

FACIN-PPGCC

Teoria da Computabilidade Parte II - Autômatos de Pilha e Máquinas de Turing Ney Laert Vilar Calazans calazans@inf.pucrs.br

Última atualização – 25/11/2008

2

Sumário

12. GRAMÁTICAS LIVRES DO CONTEXTO

- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

3

12. Gramáticas

- Mais um método para definir linguagens
- Exemplo:
 1. Uma sentença é um sujeito seguido de um predicado.
 2. Um sujeito pode ser uma expressão nominal.
 3. Uma expressão nominal pode ser um adjetivo seguido por uma expressão nominal.
 4. Uma expressão nominal pode ser um artigo seguido por uma expressão nominal.
 5. Uma expressão nominal pode ser um nome.
 6. Um predicado pode ser um verbo seguido por uma expressão nominal.
 7. Um nome pode ser *apple bear cat dog*.
 8. Um verbo pode ser *eats follows gets hugs*.
 9. Um adjetivo pode ser *itchy jumpy*.
 10. Um artigo pode ser *a an the*.
- Frases válidas na linguagem:
 - The itchy bear hugs the jumpy dog.
 - Itchy the apple eats a jumpy jumpy jumpy bear.

4

12. Gramáticas

- Na sintaxe dada:
 - Palavras grifadas/em itálico – símbolos
 - Grifadas - símbolos não-terminais
 - Em itálico – símbolos terminais
- Gramática para expressões aritméticas:
 1. START → (AE)
 2. AE → (AE + AE)
 3. AE → (AE - AE)
 4. AE → (AE * AE)
 5. AE → (AE / AE)
 6. AE → (AE ** AE)
 7. AE → (AE)
 8. AE → - (AE)
 9. AE → ANY-NUMBER
 - Gramática geradora de números:
 1. ANY-NUMBER → FIRST-DIGIT
 2. FIRST-DIGIT → FIRST-DIGIT OTHER-DIGIT
 3. FIRST-DIGIT → 1 2 3 4 5 6 7 8 9
 4. OTHER-DIGIT → 0 1 2 3 4 5 6 7 8 9
 - Cada regra acima é uma PRODUÇÃO.
 - O conjunto de regras para gerar uma palavra é uma DERIVAÇÃO ou GERAÇÃO.

5

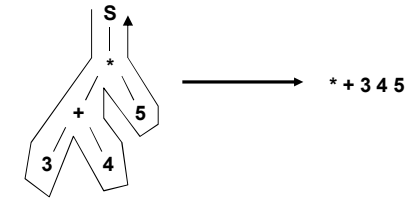
12. Gramáticas Livres do Contexto

- **Definição (Context Free Grammar - CFG)**
 - Uma gramática livre do contexto possui 3 itens:
 - 1) Um alfabeto Σ de letras terminais, do qual se formam cadeias de caracteres que serão palavras da linguagem associada à CFG.
 - 2) Um conjunto de símbolos Γ não-terminais, um dos quais é denominado símbolo inicial (S).
 - 3) Um conjunto finito de produções com a forma
 - Símbolo não-terminal \rightarrow cadeia finita de terminais e não-terminais (Pelo menos uma das produções possui o não-terminal S do lado esquerdo.)
- **Definição (Context Free Language - CFL)**
 - A linguagem gerada por um CFG é o conjunto de todas as cadeias de terminais que podem ser produzidas a partir de S usando produções.

6

12. Gramáticas

- **Notação de Lukasiewicz (ou notação polonesa)**
 - Usada em expressões aritméticas
 - Evita parênteses
 - É produzida a partir de Árvores de Derivação
 - Ex: $S \rightarrow S+S \mid S * S \mid \text{number}$



7

12. Gramáticas

- **Definição (Ambigüidade)**
 - Uma CFG é dita ambígua se para pelo menos uma palavra da linguagem existem duas possíveis árvores de derivação da palavra, correspondentes a diferentes árvores sintáticas.
- **Problemas: 1/2/3/5/10/11/16/17**

8

Sumário

- 12. GRAMÁTICAS LIVRES DO CONTEXTO
- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

9

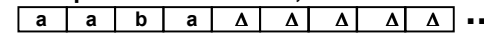
14. Autômatos de Pilha

- Cap 13 mostra que CFGs são mais poderosas que ERs/FAs/TGs, pois:
 - Todo ER/FA/TG pode ser descrito por uma CFG
 - Várias linguagens não-regulares podem ser descritas por CFGs, tais como $\{a^n b^n\}$, PALINDROME, EQUAL-EQUAL, etc.
- Deseja-se então máquinas reconhecedoras / aceitadoras de CFLs
- Parte-se de FAs e acrescenta-se “estruturas” para definir novos tipos de máquinas
- Para começar, nova representação de FAs

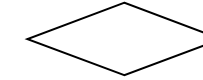
10

14. Autômatos de Pilha

- Representação da entrada do FAs --> Fita de entrada
 - possui início
 - não possui fim
 - após ler letra da Fita, letra é eliminada



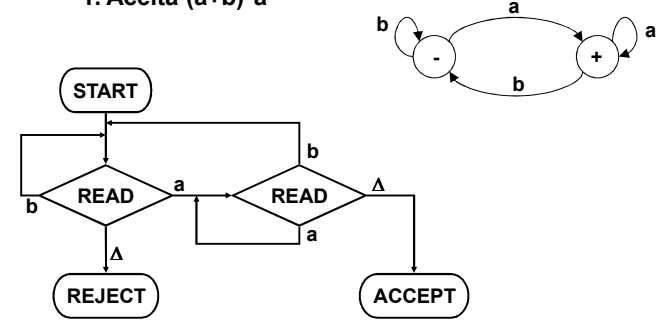
- Símbolos delimitadores
 - início, aceitação e não-aceitação
- Símbolo consumidor de caracteres da fita de entrada



11

14. Autômatos de Pilha

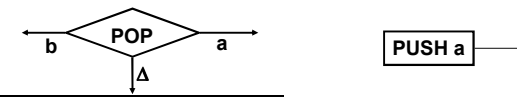
- Exemplo:
 - 1. Aceita $(a+b)^*a$



12

14. Autômatos de Pilha

- Acrescentando uma pilha a FAs com esta nova representação (equivalente à anterior)
 - Nada mais é que um autômato finito com uma pilha auxiliar (o que faz com que o todo não seja mais um FA!)
 - A pilha é infinita, possui um topo e pode conter letras do alfabeto ou Δ
 - duas novas operações são definidas, PUSH e POP, e acrescentadas à operação READ
 - Nova máquina - Autômato de Pilha



13

14. Autômatos de Pilha

- Exemplo:
 - 1. Aceita $a^n b^n$

14

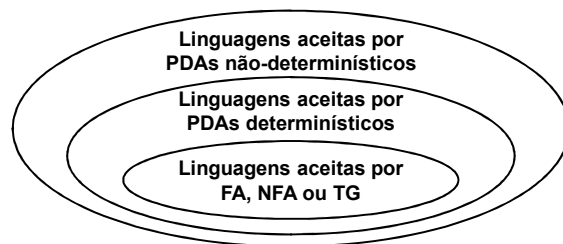
14. Autômatos de Pilha

- Autômatos de Pilha são mais poderosos que FAs, por quê?
- Motivo principal é a capacidade de memória ilimitada
- Autômatos de pilha representados pela sigla PDA (do inglês PushDown Automaton)
- PDAs determinísticos - cada cadeia de entrada percorre o mesmo caminho no PDA, sempre
- PDAs não-determinísticos - existem escolhas em tempo de execução

15

14. Autômatos de Pilha

- PDAs não-determinísticos são mais poderosos que PDAs determinísticos!! Isto não acontecia com FAs e FAS não-determinísticos, que têm o mesmo poder de reconhecer linguagens



16

14. Autômato de Pilha

- Definição de Autômato de Pilha - Um autômato de pilha é um conjunto de 8 elementos:
 - 1. Um alfabeto Σ de possíveis letras de entrada;
 - 2. Uma fita de entrada (infinita em uma direção). A cadeia a ser reconhecida está colocada a partir do início da fita e o resto desta está inicialmente vazio;
 - 3. Um alfabeto Γ de caracteres de pilha;
 - 4. Uma pilha infinita em uma direção inicialmente vazia;
 - 5. Um símbolo de partida, START;

14. Autômato de Pilha

- Definição de Autômato de Pilha (cont):

- 6. Dois símbolos finais, ACCEPT e REJECT.
- 7. Um número finito de estados PUSH x onde $x \in \Gamma$;
- 8. Um número finito de estados READ que lêem da fita e trocam de estado baseado no valor lido, avançando na fita, e um número finito de estados POP, que retiram um caractere do topo da pilha e trocam de estado baseado no valor retirado da pilha.

14. Autômatos de Pilha

- Problemas: 1/2/4/5/6/8

Sumário

12. GRAMÁTICAS LIVRES DO CONTEXTO

14. AUTÔMATOS DE PILHA

19. MÁQUINAS DE TURING

20. MÁQUINAS DE POST

21. TEOREMA DE MINSKY

23. LINGUAGENS DE MÁQUINAS DE TURING

24. A HIERARQUIA DE CHOMSKY

25. COMPUTADORES

19. Máquinas de Turing

Quadro geral de resultados de Teoria da Computabilidade

Linguagem definida por	Aceitador Correspondente	Não-determinismo=determinismo?	Linguagem fechada sob	O que pode ser decidido	Exemplo de aplicação
Expressão regular	FA, TG	Sim	união, produto, Kleene star, interseção, complemento	equivalência, "vazitude", finitude, pertencimento	editores de texto, circuitos seqüenciais
CFG	PDA	Não	união, produto, Kleene star	"vazitude", finitude, pertencimento	linguagens de programação, compiladores
Gramática tipo 0	Máquina de Turing, máquina de Post, 2PDA, nPDA	Sim	união, produto, Kleene star, interseção	Não muito	computadores

21

19. Máquinas de Turing

- **Definição de Máquina de Turing - Uma Máquina de Turing é um conjunto de 6 elementos:**
 - 1. Um alfabeto Σ de possíveis letras de entrada, que não contém o símbolo Δ (branco);
 - 2. Uma fita dividida em células numeradas, cada uma contendo uma letra de Σ ou um branco;
 - 3. Um cabeçote de leitura da fita, que pode ler uma posição desta, escrever numa posição desta ou se reposicionar exatamente uma posição à direita ou à esquerda;
 - 4. Um alfabeto Γ de possíveis caracteres da fita, que pode incluir Σ ;

22

19. Máquinas de Turing

- **Definição de Máquina de Turing (cont):**
 - 5. Um conjunto finito de estados, incluindo exatamente um estado inicial e um conjunto de estados de parada (possivelmente vazio);
 - 6. Um programa, formado por um conjunto de regras que diz, com base no estado atual e na letra que acabou de ser lida pelo cabeçote de leitura, como mudar de estado e o que imprimir na fita. Formado por uma seqüência de triplas (letra, letra, direção), onde a primeira letra é o caracter lido da fita, a segunda letra é o caracter a escrever na fita e direção move o cabeçote à esquerda ou à direita.

23

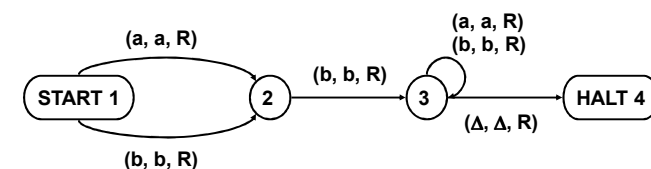
19. Máquinas de Turing

- Toda TM é determinística
- Não existe necessidade de uma TM ser completa (de um estado ter transições associadas a cada possível caracter lido da fita). Uma transição não especificada quebra (*crashes*) a TM
- Tentar ir à esquerda da primeira posição da fita quebra a máquina
- Uma TM aceita uma cadeia se o processamento desta leva a um estado de parada (HALT)

24

19. Máquinas de Turing

- **Exemplo de TM (aceita $(a+b)b(a+b)^*$):**



a	b	a	Δ	Δ	Δ	Δ	Δ	...
---	---	---	---	---	---	---	---	-----



START 1 2 3 3 HALT 4

Traço de execução: aba → aba → aba → abaΔ → abaΔΔ

25

19. Máquinas de Turing

- **Outros assuntos do Capítulo**
 - Prova de que toda linguagem regular possui uma TM que a aceita (Teorema 46)
 - Subprogramas em uma TM(e.g. INSERT e DELETE)

26

19. Máquinas de Turing

- **Problemas: 1/2/3/4/6/8**

27

Sumário

- 12. GRAMÁTICAS LIVRES DO CONTEXTO
- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST**
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

28

20. Máquinas de Post

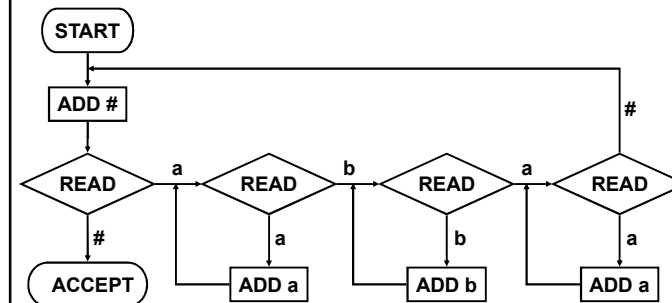
- **Definição de Máquina de Post - Uma Máquina de Post é um conjunto de 5 elementos:**
 - 1. Um alfabeto Σ de possíveis letras de entrada, mais o símbolo especial #;
 - 2. Uma fila linear (STORE ou QUEUE) que inicialmente contém a cadeia de entrada. Letras são removidas à esquerda e inseridas à direita da fila. O alfabeto de caracteres da fila é Γ que pode ser o mesmo de Σ ou ter mais caracteres;
 - 3. Estados de leitura (READ), que consomem caracteres da fila, e tomam decisão baseado neste. Usa-se o mesmo símbolo gráfico de PDAs para representar tais estados;

20. Máquinas de Post

- **Definição de Máquina de Post (cont):**
 - 4. Estados de adição de caracteres (ADD) à fila, representados por retângulos;
 - 5. Símbolos START (único) e um conjunto de símbolos ACCEPT e REJECT.
- **Notar que não existe a fita de entrada, a fila única é usada para entrada e saída;**
- **Notar similitude com PDAs;**
- **PMs têm poder equivalente a TMs (Teorema 49)!**
- **Exercícios do Capítulo: Nenhum (ilustração é objetivo)**

20. Máquinas de Post

- **Exemplo:**
 - 1. Aceita $a^n b^n a^n$



Sumário

- 12. GRAMÁTICAS LIVRES DO CONTEXTO
- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

21. Teorema de Minsky

- **Uma pilha acrescentou poder significativo a FAs;**
- **TMs não são FAs, TG ou PDAs mais algumas estruturas;**
- **O que acontece quando se acrescenta uma pilha a mais, ou duas, ou setenta?**
- **Definição de 2PDA**
 - É uma máquina similar a um PDA, com duas diferenças:
 - Possui duas pilhas com operações modificadas de acordo (dois tipos de PUSH, $PUSH_1$ e $PUSH_2$, dois tipos de POP, POP_1 e POP_2);
 - É determinística.

33

21. Teorema de Minsky

- **Exemplo:**
 - 1. 2PDA que aceita $a^n b^n a^n$, uma linguagem que não pode ser aceita por nenhum PDA!
- **Marvin Minsky provou, em 1961 o Teorema 50**
- **Teorema 50: 2PDA=TM, ou seja, qualquer linguagem aceita por um 2PDA pode ser aceita por alguma TM e vice e versa.**
- **Prova:**
 - Primeira parte - Se L pode ser aceita por algum 2PDA, é possível construir uma TM que aceita L.

34

21. Teorema de Minsky

- Segunda parte - Se L pode ser aceita por alguma TM, é possível construir um 2PDA que aceita L.
- **Teorema 51: nPDA=TM, ou seja, qualquer linguagem aceita por um nPDA pode ser aceita por alguma TM e vice e versa. ($n \geq 2$)**

35

Sumário

- 12. GRAMÁTICAS LIVRES DO CONTEXTO
- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

36

23. Linguagens de Máquinas de Turing

- **CAP 22 define uma série de variações de TM e prova que todas são igualmente poderosas, com exceção da última:**
 - Move-in state machine TM- onde a movimentação L/R é característica do estado, e não da transição;
 - Stay-option TM - onde não necessariamente se move o cabeçote de leitura a cada transição;
 - k-track TM - onde existem k fitas ao invés de uma;
 - Two-way infinite TM - onde a fita é infinita para ambos os lados;
 - Non-deterministic TM;
 - Read-only TM - onde não se pode escrever nada na fita - aceita apenas linguagens regulares!!

37

23. Linguagens de Máquinas de Turing

- Linguagens aceitas por FAs/TGs
 - regulares
- Linguagens aceitas por PDAs
 - livres do contexto
- Linguagens aceitas por TMs
 - recursivamente enumeráveis

38

23. Linguagens de Máquinas de Turing

- **Definição: Linguagens Recursivamente Enumeráveis (linguagens r.e.)**
 - Uma linguagem L sobre um alfabeto Σ é dita recursivamente enumerável se existe uma TM que aceita toda palavra em L e, ou rejeita (quebra), ou entra em laço infinito para toda palavra na linguagem L' , o complemento de L .
- **Definição: Linguagens Recursivas**
 - Uma linguagem L sobre um alfabeto Σ é dita recursiva se existe uma TM que aceita toda palavra em L e rejeita (quebra para) toda palavra na linguagem L' , o complemento de L .

39

23. Linguagens de Máquinas de Turing

- **Teorema 60**
 - Se a linguagem L é recursiva então seu complemento, L' , é uma linguagem recursiva. Em outras palavras, o conjunto de linguagens recursivas é fechado sob a operação de complemento.
- **Prova no livro, usando PMs e não TMs. Como o conjunto de linguagens r.e. contém o conjunto das linguagens recursivas:**
- **Teorema 61**
 - Se a linguagem L é recursivamente enumerável e seu complemento L' é uma linguagem recursivamente enumerável, então L é recursiva.

40

23. Linguagens de Máquinas de Turing

- **Teorema 62**
 - Se $T1$ e $T2$ são TMs, então existe uma TM $T3$ tal que $\text{accept}(T3) = \text{accept}(T1) + \text{accept}(T2)$.
- **Em palavras, o Teorema 62 mostra que a união de duas linguagens r.e. é recursivamente enumerável, ou seja, o conjunto de linguagens r.e. é fechado sob a operação de união.**
- **Teorema 63**
 - A intersecção de duas linguagens r.e. também é recursivamente enumerável.

41

23. Linguagens de Máquinas de Turing

- Perguntas ainda por responder sobre TMs:
 - 1. O produto de duas TMs é uma TM?
 - 2. O fechamento Kleene de uma TM é uma TM?
 - 3. O complemento de uma TM é uma TM?
 - 4. Existem linguagens fora do conjunto r.e.?
 - 5. É possível decidir para TMs
 - 5.1. “vaziitude”?
 - 5.2. finitude?
 - 5.3. “pertencimento”?
- Inicia-se com a questão 4
- Introdução de uma codificação para representar TMs

42

23. Linguagens de Máquinas de Turing

- A linguagem ALAN (definição):
 - ALAN = {todas as palavras na linguagem de codificação de palavras (CWL) que não são aceitas pelas TMs que elas representam, ou que não representam qualquer TM}
- ALAN não é r.e.
 - prova por contradição
- Conclusão: nem todas linguagens são r.e. (Teorema 64)

43

23. Linguagens de Máquinas de Turing

- Outros resultados
 - Teorema 67 - O complemento de uma linguagem r.e. pode não ser r.e.
 - Teorema 68 - Existem linguagens r.e. que não são recursivas
 - Teorema 69 - Não existe nenhuma TM que pode aceitar qualquer cadeia w e qualquer TM T codificada e sempre decidir corretamente se T pára com w . Em outras palavras, o problema da parada não pode ser decidido por uma TM.
 - Teorema 70 - Não existe nenhuma TM que possa decidir, para toda TM T , alimentada na primeira sob forma codificada, se T aceita a palavra vazia Λ .

44

23. Linguagens de Máquinas de Turing

- Outros resultados (cont.)
 - Teorema 71 - Não existe nenhuma TM que, quando alimentada com a palavra de código arbitrária de uma TM arbitrária, possa sempre decidir se a TM codificada aceita pelo menos uma palavra. Em outras palavras, a questão de “vaziitude” de linguagens r.e. não pode ser decidida por TMs.
 - Teorema 72 - Não existe uma TM que possa decidir, para qualquer TM T codificada alimentada na primeira, se a linguagem de T é finita ou infinita.
- Problemas: 1/2/3/4

Sumário

- 12. GRAMÁTICAS LIVRES DO CONTEXTO
- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

24. Hierarquia de Chomsky

Quadro geral de resultados de Teoria da Computabilidade (repetido do Cap 19)

Linguagem definida por	Aceitador Correspondente	Não-determinismo=determinismo?	Linguagem fechada sob	O que pode ser decidido	Exemplo de aplicação
Expressão regular	FA, TG	Sim	união, produto, Kleene star, interseção, complemento	equivalência, "vazitude", finitude, pertencimento	editores de texto, circuitos seqüenciais
CFG	PDA	Não	união, produto, Kleene star	"vazitude", finitude, pertencimento	linguagens de programação, compiladores
Gramática tipo 0	Máquina de Turing, máquina de Post, 2PDA, nPDA	Sim	união, produto, Kleene star, interseção	Não muito	Computadores

- **Falta:**
 - Justificativa para última linha
 - Def. de conjuntos recursivamente enumeráveis s/ TMs

24. Hierarquia de Chomsky

- **Definição:** Uma Gramática de Estrutura Frasal (Phrase-Structure Grammar - PSG) é um conjunto de 3 elementos:
 - 1. Um alfabeto finito Σ de letras terminais;
 - 2. Um alfabeto finito de símbolos denominados não-terminais, que inclui o símbolo inicial S;
 - 3. Uma lista finita de produções da forma
 - Cadeia1 \rightarrow Cadeia2,
 onde Cadeia1 pode ser qualquer cadeia de terminais e não-terminais que contém no mínimo 1 não-terminal e Cadeia2 pode ser qualquer cadeia de terminais e não-terminais.

24. Hierarquia de Chomsky

- **Derivação** (em PSGs) - série de cadeias intermediárias que começam com S, modificadas pela aplicação de produções, chegando a uma cadeia com apenas símbolos terminais, onde o processo de geração pára.
- **Linguagem Gerada** (por uma PSG) - conjunto de todas as cadeias de terminais que podem ser derivadas de S.

24. Hierarquia de Chomsky

- Exemplo de PSG:

- $\Sigma = \{a,b\}$, não-terminais= $\{S, X\}$

- Produções

- Prod1 $S \rightarrow XS \mid \Lambda$
- Prod2 $X \rightarrow aX \mid a$
- Prod3 $aaX \rightarrow ba$

- Exemplo de derivação:

- $S \Rightarrow XXXXXX$ (Aplicando 7 vezes a Prod1 – 6 Esq/1 Dir)
- $\Rightarrow aaaaaXXXXX$ (Aplicando 5 vezes a Prod2 – 4 Esq/ 1 Dir)
- $\Rightarrow aabaXXXX$ (Aplicando 1 vez a Prod3)
- $\Rightarrow aabaaaXXX$ (Aplicando 2 vezes a Prod2 – 1 Esq/ 1 Dir)
- $\Rightarrow aabbaXX$ (Aplicando 1 vez a Prod3)
- $\Rightarrow aabbaaaX$ (Aplicando 2 vezes a Prod2 – 1 Esq/ 1 Dir)
- $\Rightarrow aabbba$ (Aplicando 1 vez a Prod3)

24. Hierarquia de Chomsky

- Teorema 73

- Pelo menos uma linguagem que não pode ser gerada por uma CFG pode ser gerada por uma gramática de estrutura frasal.

- Prova por método construtivo ($\{a^n b^n a^n\}$):

- Prod1 $S \Rightarrow aSBA$
- Prod2 $S \Rightarrow abA$
- Prod3 $AB \Rightarrow BA$
- Prod4 $bB \Rightarrow bb$
- Prod5 $bA \Rightarrow ba$
- Prod6 $aA \Rightarrow aa$

- Linguagens de estrutura frasal são uma classe mais ampla que linguagens livres do contexto!

24. Hierarquia de Chomsky

- Teorema 74

- Se uma PSG gera uma linguagem L, existe outra PSG que gera L que possui o mesmo alfabeto de terminais e na qual as produções são da forma

- cadeia de não-terminais \rightarrow cadeia de terminais e não-terminais

- Definição - Uma PSG é dita do tipo 0 se cada produção possui a forma

- cadeia de não-terminais \rightarrow cadeia de terminais e não-terminais

24. Hierarquia de Chomsky

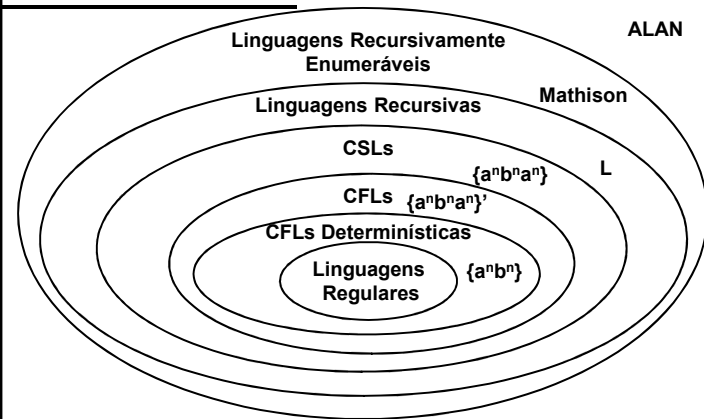
A Hierarquia de Chomsky para gramáticas

Tipo	Nome das linguagens geradas	Restrições sobre as produções $X \rightarrow Y$	Aceitador
0	Estrutura frasal = recursivamente enumerável	$X =$ qualquer string de não-terminais $Y =$ qualquer string	TM
1	Sensível ao contexto	$X =$ qualquer string de não-terminais $Y =$ qualquer string, desde que mais longo que X	TMs com fitas finitas
2	Livre do contexto	$X =$ 1 não-terminal $Y =$ qualquer string	PDA
3	Regular	$X =$ 1 não-terminal $Y = tN$ ou $Y = t$, onde t é terminal e N é não-terminal	FA/TG

- Linguagens não incluídas: CFLs determinísticas e linguagens recursivas

53

24. Hierarquia de Chomsky..



54

24. Hierarquia de Chomsky

- Outros resultados: Teoremas 75 a 81
- Problemas: 1/2/3/4/5/12/13

55

Sumário

- 12. GRAMÁTICAS LIVRES DO CONTEXTO
- 14. AUTÔMATOS DE PILHA
- 19. MÁQUINAS DE TURING
- 20. MÁQUINAS DE POST
- 21. TEOREMA DE MINSKY
- 23. LINGUAGENS DE MÁQUINAS DE TURING
- 24. A HIERARQUIA DE CHOMSKY
- 25. COMPUTADORES

56

25. Computadores

- **Codificação unária**
 - Usando a's para representar números separados por b's
 - Exemplo de somador unário
- **Codificação binária**
 - incrementador
 - decrementador
 - somador de dois números

57

25. Computadores

- **Definição:** Se uma TM tem a propriedade de que, para toda palavra que esta aceita após parar a TM deixa uma cadeia de a's e b's na sua fita a partir da célula 1 desta, denomina-se esta TM um computador. A cadeia de entrada é chamada de entrada da TM e é identificada como uma seqüência de números inteiros não-negativos. A cadeia que resta na fita após o processamento é a saída da TM e é também identificada como uma seqüência de inteiros não-negativos.

58

25. Computadores

- **Definição:** Se uma TM toma uma seqüência de inteiros não-negativos como entrada e deixa apenas um número como saída, diz-se que o computador agiu como uma função matemática. Qualquer operação que é definida sobre uma seqüência de K números ($K \geq 1$), e que pode ser executada por uma TM é dita ser Turing-computável ou computável.

59

25. Computadores

- **Exemplos de funções computáveis:**
 - Adição e subtração (Teorema 82)
 - MAX (x,y) (Teorema 83)
 - Identidade e Sucessor (Teorema 84)
 - Seletor de l-ésimo elemento (Teorema 85)
 - Multiplicação (Teorema 86)

60

25. Computadores

- **Tese de Church (Alonzo Church – 1936)**
 - “Acredita-se que não existam funções que possam ser definidas por seres humanos, cujas computações possam ser descritas por algum algoritmo matemático bem definido, e que pessoas possam ser ensinadas a aplicar que não possam ser computadas por uma TM.”
 - Não é teorema
 - Há razões para acreditar que esta tese é válida
 - Foi definida antes de existirem TMs
 - Enunciado original um pouco diferente

25. Computadores

- **TMs como geradores de linguagens**
 - Toda linguagem gerável por uma TM é aceitável por alguma TM e vice-versa (Teoremas 87, 88 e 89)
 - **Problemas: 1/2/3/4**
-