



SCC-5832 - Capítulo 3 - Linguagens Sensíveis ao Contexto, Linguagens Recursivamente Enumeráveis e Máquinas de Turing

João Luís Garcia Rosa¹

¹Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
joaoluis@icmc.usp.br

2009

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Definição

- Como uma generalização das regras livres de contexto, introduz-se agora regras sensíveis ao contexto que também especificam a substituição de um símbolo não terminal, mas requer um contexto para aplicação.
- Portanto a regra $bC \rightarrow bc$ é *sensível ao contexto* porque ela diz que um não terminal C pode ser substituído pelo símbolo terminal c apenas no contexto de um b precedente.
- **Definição:** Uma gramática G é **sensível ao contexto** se cada produção ou é da forma
 - 1 $yAz \rightarrow ywz$, para $A \in V$, $y, z \in (V \cup \Sigma)^*$, $w \in (V \cup \Sigma)^+$; ou
 - 2 $S \rightarrow \lambda$, dado que S não aparece no lado direito de nenhuma produção.
- Uma linguagem é sensível ao contexto se pode ser gerada por uma gramática sensível ao contexto.

Exemplo de GSC

- **Fato:** Seja G uma gramática tal que toda produção de G é da forma $v \rightarrow w$, com $|v| \leq |w|$, exceto que pode existir uma única produção- λ $S \rightarrow \lambda$ se S não aparece do lado direito de nenhuma produção. Então, existe uma gramática sensível ao contexto G' que é equivalente a G : $L(G) = L(G')$ [1].
- **Exemplo** de Gramática Sensível ao Contexto:
 - $G = (\Sigma, V, S, P)$, $\Sigma = \{a, b\}$, $V = \{S, B, C\}$ e $P = \{$
 - 1 $S \rightarrow aSBC,$
 - 2 $S \rightarrow aBC,$
 - 3 $CB \rightarrow BC,$
 - 4 $aB \rightarrow ab,$
 - 5 $bB \rightarrow bb,$
 - 6 $bC \rightarrow bc,$
 - 7 $cC \rightarrow cc\}$

Exemplo de GSC

- A linguagem $L(G)$ contém a palavra $a^n b^n c^n$ para cada $n \geq 1$, pois pode-se usar a produção (1) $n - 1$ vezes para chegar a $S \Rightarrow^* a^{n-1} S(BC)^{n-1}$. Depois usa-se a (2) para $S \Rightarrow^* a^n (BC)^n$. A produção (3) permite arranjar os B s e C s tal que todo B preceda todos os C s. Por exemplo, se $n = 3$:

$$aaaBCBCBC \Rightarrow aaaBBCCBC \Rightarrow aaaBBCBCC \Rightarrow \\ aaaBBBCCC$$

- Portanto, $S \Rightarrow^* a^n B^n C^n$. Depois usa-se (4) uma vez para chegar a $S \Rightarrow^* a^n b B^{n-1} C^n$. Aí usa-se a produção (5) $n - 1$ vezes: $S \Rightarrow^* a^n b^n C^n$. Finalmente, usa-se a produção (6) uma vez e a produção (7) $n - 1$ vezes para chegar a $S \Rightarrow^* a^n b^n c^n$. É necessário mostrar também que as cadeias $a^n b^n c^n$, $n \geq 1$, são as únicas cadeias terminais em $L(G)$.

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Lema da Cadeia Vazia

- Note que nem toda gramática livre de contexto é sensível ao contexto, porque as produções livres de contexto podem ser da forma $A \rightarrow \lambda$. Entretanto, toda linguagem livre de contexto é uma linguagem sensível ao contexto.
- **Lema: (O Lema da Cadeia Vazia).** Seja G uma gramática livre de contexto que envolve regras da forma $A \rightarrow \lambda$ para qualquer $A \in V$. Então existe uma gramática livre de contexto G' tal que
 - 1 $L(G) = L(G')$;
 - 2 Se $\lambda \notin L(G)$ então não existem produções da forma $A \rightarrow \lambda$ em G' ; mas
 - 3 Se $\lambda \in L(G)$, então existe uma única produção- λ em G' , $S' \rightarrow \lambda$, onde S' é o símbolo inicial de G' e S' não aparece no lado direito de nenhuma produção em G' .

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Lema da Cadeia Vazia: Prova

- **PROVA:** Suponha que $\lambda \notin L(G)$. Para $C \in V$ considere todas as produções $C \rightarrow w$, onde w não é vazio. Agora suponha que w contém variáveis $A_1, \dots, A_k, B_1, \dots, B_j$ onde $A_i, 1 \leq i < k$, são aquelas variáveis em w para as quais $A_i \Rightarrow^* \lambda$. Adicione a G a produção $C \rightarrow w'$, onde w' é obtida de w apagando zero ou mais ocorrências de uma variável dos A_i 's. Então, por exemplo,

$$C \rightarrow aA_1BA_2$$

leva a quatro produções

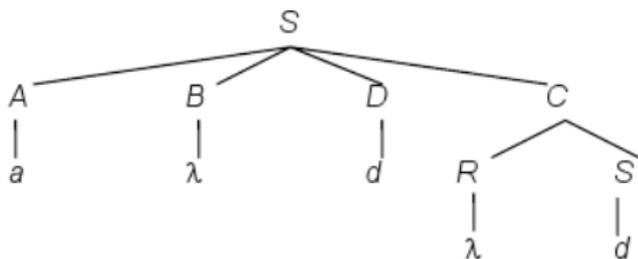
$$C \rightarrow aA_1BA_2 \mid aBA_2 \mid aA_1B \mid aB$$

Quando este processo estiver completo, remova todas as produções- λ , incluindo qualquer produção- λ nova que tenha sido incluída no processo. A gramática resultante é G' . Então $L(G) = L(G')$ como se segue.

Lema da Cadeia Vazia: Prova

- Se uma derivação de alguma cadeia $w \in L(G)$ envolveu produções- λ , então a árvore de derivação para w terá nós rotulados com λ , como no seguinte exemplo.

$S \rightarrow ABDC$
 $C \rightarrow RS$
 $B \rightarrow \lambda$
 $A \rightarrow a$
 $D \rightarrow d$
 $R \rightarrow \lambda$
 $S \rightarrow d$



- Para obter uma derivação G' para w a partir de uma derivação G para w , comece no topo da árvore G e apague qualquer sub-árvore da raiz da árvore que deriva λ . O topo da árvore deve agora ilustrar uma produção a partir de G' .

Lema da Cadeia Vazia: Prova

- Então considere os sucessores imediatos do nó raiz que não derivam λ . Aplique o mesmo princípio a cada um deles: se alguma das suas sub-árvores deriva λ , apague a sub-árvore. Continue desta forma para baixo na árvore. A árvore final representa uma derivação G' . Além disto, esta árvore deriva w , porque uma subcadeia de w é removida pelo processo de apagamento se e somente se ela for λ . Por outro lado, se $w \in L(G')$ e a derivação de w envolve uma regra recém adicionada, então simule a derivação de G' usando a regra G original apropriada e regras- λ . Isto é, faça o processo descrito acima na ordem reversa, novamente de cima para baixo.

Lema da Cadeia Vazia

- No caso onde $\lambda \in L(G)$, primeiro adicione um novo símbolo inicial a G' , por exemplo S' , e adicione duas novas produções, $S' \rightarrow S$ e $S' \rightarrow \lambda$. Depois repita o processo descrito previamente em todas as produções exceto as produções S' . Então $L(G) = L(G')$.
- **Teorema:** Uma linguagem L é *sensível ao contexto* se e somente se existe alguma gramática G tal que $L = L(G)$ onde toda produção de G da forma $u \rightarrow v$ tem a propriedade de que $0 < |u| \leq |v|$ com uma exceção: se $\lambda \in L(G)$, então a regra $S \rightarrow \lambda$ está também presente e neste caso S não pode aparecer no lado direito de nenhuma produção.

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Máquina de Turing

- Vai-se relacionar agora a teoria das linguagens formais com a teoria abstrata da computação.
- Como se pode caracterizar o conjunto de funções computadas por programas de computador?
- Esta questão - quais funções são realizáveis por algoritmos e quais não são? - tem suas raízes mais diretas no trabalho de Alan Turing nos anos 1930 [7].
- Usando, o que agora se chama de modelo de máquina de Turing, Turing mostrou que certos problemas naturais em computação não podem ser computados por nenhum algoritmo, real ou imaginado.

Máquina de Turing

- Realmente, Turing mostrou apenas que estes problemas não são calculáveis especificamente por máquinas de Turing; mais tarde as investigações de outros pesquisadores levaram a crença geral que a computabilidade de Turing é sinônimo da computabilidade em qualquer outro sistema de especificação de algoritmo suficientemente poderoso.
- Introduz-se as máquinas de Turing apresentando os principais resultados de Turing, a universalidade das máquinas de Turing e a insolubilidade do problema da parada (*halting*).
- Na sua forma mais geral, o último resultado diz que nenhum algoritmo pode corretamente decidir se um programa de computador arbitrário pára em uma entrada arbitrária.

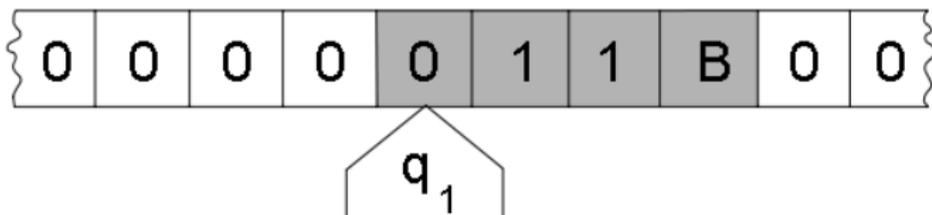
Máquina de Turing

- Vai-se mostrar como codificar este resultado na teoria da linguagem.
- Usando esta codificação prova-se vários resultados, incluindo a asserção de que nenhum algoritmo pode corretamente decidir se uma gramática livre de contexto arbitrária é ambígua.

Máquina de Turing: Definição

- Uma máquina de Turing T pode ser descrita (Figura 14) como um controle de estados finitos equipado com um dispositivo de armazenamento externo na forma de uma fita finita que pode ser estendida indefinidamente em ambas as direções.

Figure: Uma máquina de Turing: A fita pode ser estendida indefinidamente pela adição de B's (brancos) em qualquer lado [9].



Máquina de Turing: Definição

- Como mostrado na Figura 14, a fita é dividida em quadrados.
- Cada quadrado da fita pode estar em branco (B) ou pode carregar qualquer símbolo de um alfabeto de fita finito Σ especificado.
- Por conveniência, um símbolo especial (aqui, a letra B) não em Σ é reservado para denotar um quadrado em branco na fita.
- O controle de estados finitos é acoplado à fita através de uma cabeça de leitura/escrita.
- A qualquer instante dado, a cabeça percorre um quadrado da fita e o controle de estados finitos está em um estado.

Máquina de Turing: Definição

- Dependendo deste estado e do símbolo no quadrado percorrido pela cabeça da fita, a máquina irá, em um passo, fazer o seguinte:
 - 1 Entrar em um novo estado do controle de estados finitos;
 - 2 Sobrescrever um símbolo no quadrado percorrido (é possível sobrescrever o mesmo símbolo e portanto deixar o quadrado da fita sem mudar, ou sobrescrever um B e “apagar” o símbolo da fita);
 - 3 Deslocar a cabeça para esquerda ou para a direita um quadrado.
- Como o alfabeto da fita é finito e o controle de estados finitos tem finitamente muitos estados, a operação de uma máquina de Turing pode ser completamente descrita por um conjunto finito de quintuplas da forma (*estado antigo, símbolo percorrido, estado novo, símbolo escrito, direção de movimento*).

Máquina de Turing: Exemplo

- **Exemplo** : A seguinte máquina de Turing aceita o conjunto $\{a^{2n} | n \geq 0\}$ de cadeias de comprimento par de a 's: quando começa no estado “par” no a mais a esquerda, com a configuração de fita inicial $Baaa...aaaB$ ela irá terminar no estado “aceitação” apenas no caso do número de a 's ser par. Aqui está a máquina:

(par a ímpar $B R$)

(ímpar a par $B R$)

(par B aceitação $B R$)

- Quando ela alcança primeiro um B ela muda para o estado “aceitação” se ela viu um número par (incluindo 0) de a 's mas de outra forma ela irá parar porque não há quintupla para guiar seu próximo movimento.

Máquina de Turing: Exemplo

- Por outro lado, a máquina com quintuplas:
 - (par a ímpar $B R$)
 - (ímpar a par $B R$)
 - (par B aceitação $B R$)
 - (ímpar B ímpar $B R$)
- tem a propriedade de que em cadeias de comprimento par de a 's ela pára no estado de aceitação, mas em cadeias de comprimento ímpar ela “roda” para sempre.
- Isto ilustra como as máquinas de Turing são realmente diferentes dos Autômatos Finitos (AFDs).
- Em particular, as computações da máquina de Turing podem fornecer resultados indefinidos - computações que nunca terminam.
- Tal comportamento não é possível com AEFs.

Máquina de Turing

- Assim como com os AEFs, pode-se dar uma representação equivalente das quintuplas no Exemplo 8 usando uma tabela de estado, como se segue:

estado	símbolo	percorrido
antigo	a	B
par	ímpar B R	aceitação B R
ímpar	par B R	

- A fim de tornar o comportamento da máquina determinístico, necessita-se que quando duas quintuplas têm a mesma combinação (estado antigo, símbolo percorrido), então as duas quintuplas são idênticas.
- Veja agora a definição formal de uma máquina de Turing e a linguagem que ela aceita.

Máquina de Turing: Definição

- **Definição:** Uma máquina de Turing M é uma quintupla $M = (Q, \Sigma, q_0, q_a, \delta)$, onde
 - 1 Q é um conjunto finito de estados;
 - 2 Σ é um alfabeto finito da fita. Escreve-se $\Sigma' = \Sigma \cup \{B\}$ para Σ aumentado por um símbolo de fita distinto B , o símbolo branco;
 - 3 q_0 é o estado inicial distinto;
 - 4 q_a é o estado de aceitação distinto;
 - 5 δ é a função (parcial) de transição de estado,

$$\delta : Q \times \Sigma' \rightarrow Q \times \Sigma' \times \{L, R, S\}$$

sujeita a condição de que $\delta(q_a, x)$ não é definida para nenhum x em Σ' . Pode-se representar δ por uma lista de quintuplas, com $(q \ x \ q' \ x' \ D)$ na lista apenas no caso em que $\delta(q, x) = (q', x', D)$.

Máquina de Turing: Descrição Instantânea

- Pode-se descrever a configuração de uma máquina de Turing M especificando:
 - 1 a cadeia w_1 impressa na fita a esquerda da cabeça de leitura/escrita;
 - 2 o estado corrente q ; e
 - 3 a cadeia w_2 na fita, começando na cabeça de leitura/escrita.

- Pode-se resumir isto na cadeia

$$w_1 q w_2$$

que é chamada de **descrição instantânea** (DI) de M .

- Note que w_1 e w_2 não são únicos - a cadeia $w_1 w_2$ deve conter todos os quadrados não brancos da fita de M , mas ela pode ser estendida adicionando brancos em qualquer lado: $1q_01B11$ e $BB1q_01B11BBB$ são duas formas de escrever a mesma DI.

Máquina de Turing: Descrição Instantânea

- As máquinas de Turing têm fitas, cabeças de leitura/escrita, etc., e não podem obviamente ser descritas usando a maquinaria da teoria da linguagem.
- As descrições instantâneas, por outro lado, são realmente cadeias de símbolos e como tais podem ser manipuladas por uma gramática formal.
- Pode-se ligar a teoria da linguagem e a noção de computabilidade da máquina de Turing, o que será feito mais tarde nesta parte, mostrando como uma gramática reescrevendo uma descrição instantânea pode confiavelmente seguir a trilha das transições de uma computação da máquina de Turing.

Máquina de Turing: Descrição Instantânea

- **Definição:** Seja M uma máquina de Turing com conjunto de estados Q e alfabeto de fita aumentado Σ' . Então uma **descrição instantânea** (DI) de M é uma cadeia w_1qw_2 do conjunto $(\Sigma')^* \cdot Q \cdot (\Sigma')^*$. Cada DI w_1qw_2 é equivalente às DIs Bw_1qw_2 e w_1qw_2B .
- Diz-se que M pára em w_1qw_2 se nem w_1qw_2 nem w_1qw_2B for da forma w_1qxw para um par (q, x) para o qual $\delta(q, x)$ é definido.
- Se M não pára para w_1qw_2 , define-se como as DIs são transformadas pela ação de M .
- Primeiro, escreve-se w_1qw_2 como $w_{1A}yqxw_{2A}$ onde $w_{1A}y = w_1$ se $w_1 \neq \lambda$, e $w_{1A} = \lambda$, $y = B$ caso contrário; e $xw_{2A} = w_2$ se $w_2 \neq \lambda$, e $x = B$, $w_{2A} = \lambda$ caso contrário.

Máquina de Turing: Descrição Instantânea

- Define-se então a relação próxima DI

$$w_1 q w_2 \Rightarrow w'_1 q' w'_2$$

como se segue:

- se $\delta(q, x) = (q', x', L)$ então

$$w'_1 q' w'_2 = w_{1A} q' y x' w_{2A}$$

- se $\delta(q, x) = (q', x', S)$ então

$$w'_1 q' w'_2 = w_{1A} y q' x' w_{2A}$$

- se $\delta(q, x) = (q', x', R)$ então

$$w'_1 q' w'_2 = w_{1A} y x' q' w_{2A}$$

- Como de costume, estende-se \Rightarrow para formar seu fechamento reflexivo e transitivo \Rightarrow^* tal que

$$\alpha \Rightarrow^* \alpha'$$

significa que M pode transformar a DI α na DI α' através de 0 ou mais transições de DI sucessoras como descrito em (1)-(2) anteriormente.

Máquina de Turing

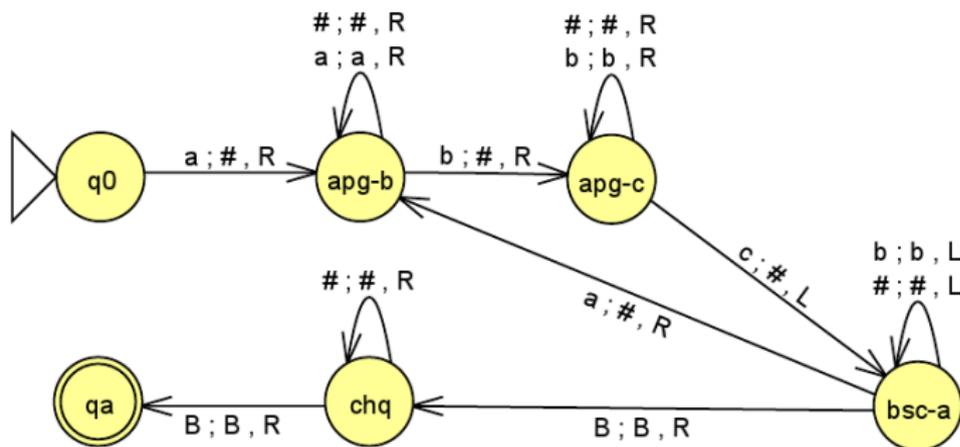
- Para uma máquina de Turing M sobre o alfabeto Σ , define-se $T(M)$, o conjunto aceitação de M , como o conjunto das cadeias w sobre Σ tal que se w é escrita em uma fita em branco e M começa processando no estado q_0 no símbolo mais a esquerda de w , então M eventualmente pára no estado de aceitação. Formalmente:
- **Definição:** O conjunto de aceitação $T(M)$ de uma máquina de Turing M sobre o alfabeto Σ é o conjunto

$$T(M) = \{w \in \Sigma^* \mid \exists w_1, w_2 \in (\Sigma')^* \mid q_0 w \Rightarrow^* w_1 q_a w_2\}.$$

Máquina de Turing: Exemplo

- **Exemplo:** A seguinte máquina de Turing aceita a linguagem $\{a^n b^n c^n \mid n > 0\}$:

Figure: Uma máquina de Turing para processar a linguagem $\{a^n b^n c^n \mid n > 0\}$.



Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - **Autômatos Limitados Linearmente**
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

ALL: Definição

- Vai-se considerar agora os **autômatos limitados linearmente**.
- Estes dispositivos são máquinas de Turing não determinísticas com a restrição de que a cabeça de leitura/escrita não tem a permissão de deixar os quadrados da fita que possuem a entrada original.
- Mostrar-se-á que estas máquinas são precisamente os aceitadores para linguagens sensíveis ao contexto.
- **Definição:** Um autômato limitado linearmente (ALL) é uma máquina de Turing não determinística que deve operar dentro do espaço definido pela colocação original da entrada na fita.

O Lema do Alfabeto

- **Lema do Alfabeto:** Seja L uma linguagem tal que para qualquer $w \in \Sigma^*$, a pertinência de w em L pode ser decidida por uma máquina de Turing M usando $k \cdot |w|$ quadrados da fita, onde k é um inteiro positivo independente de w . Então L pode ser reconhecida por alguma outra máquina de Turing M' usando apenas $|w|$ quadrados da fita.
 - Então o lema do alfabeto estabelece que se alguma máquina de Turing pode reconhecer uma linguagem L no espaço que é uma função linear do comprimento de uma cadeia de entrada, então alguma outra máquina de Turing pode de fato reconhecer L dentro do espaço exatamente igual do comprimento da entrada.
 - O relacionamento linear identificado no lema justifica o nome autômatos limitados linearmente.

Limites à Esquerda e à Direita

- Assume-se, quando conveniente, que a entrada da máquina de Turing é limitada a esquerda por um símbolo \$ e a direita por um ϕ . Assume-se que estes símbolos são não apagáveis e que a cabeça não pode mover sobre eles.
- **Teorema:** Se $L = T(M)$ para algum ALL M , então L é gerável por alguma gramática sensível ao contexto.
- **PROVA:** Vai-se assumir que $\lambda \notin L$. Para $\lambda \in L$ usa-se o teorema para $L - \{\lambda\}$ e com cuidado adiciona-se uma produção $S \rightarrow \lambda$ depois do fato. Para estabelecer o resultado assumindo $\lambda \notin L$, primeiro escreve-se uma GSC para a linguagem

$$\{w\$q_0w \mid w \in \Sigma^*\}$$

Limites à Esquerda e à Direita

- A cadeia a direita do símbolo \$ é de comprimento $|w| + 2$. Já que a máquina neste caso é um ALL, as DIs nunca serão maiores que isto. Entretanto, no término da simulação de gramática de M , não se pode apagar todos os símbolos a direita de \$, porque as regras de apagamento não são sensíveis ao contexto. Ao invés disto, se q_a aparece, faz-se o seguinte:
 - 1 Primeiro, converte-se todos os símbolos a direita de \$ para um novo símbolo, “/”, tal que fica-se com a seguinte cadeia:
 $w\$/^nq_a$
 - 2 Se $w = x_1x_2\dots x_n$, usa-se regras sensíveis ao contexto para construir a seguinte cadeia: $r/x_1/x_2/\dots/x_n\#q$ (onde r e q são novos símbolos terminais);
 - 3 Apaga-se todas as ocorrências de r , $/$, $\#$ e q .

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - **Gramáticas Irrestritas**
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Definição [4]

- **Definição de Gramáticas Irrestritas:** As produções de uma gramática têm a forma:

$$(V \cup \Sigma)^+ \rightarrow (V \cup \Sigma)^*$$

sendo que o lado esquerdo possui no mínimo uma variável (elemento de V). Os outros tipos de gramáticas consideradas (linear a direita, livre de contexto, sensível ao contexto) restringem a forma das produções. Uma gramática irrestrita, não.

- Vai-se tentar mostrar que as gramáticas irrestritas são equivalentes às Máquinas de Turing.

Definição

- Lembre-se que:
 - Uma linguagem é recursivamente enumerável se existe uma máquina de Turing que aceita toda cadeia da linguagem, e não aceita cadeias que não pertencem à linguagem.
 - “Não aceita” não é o mesmo que “rejeita” - a máquina de Turing poderia entrar num loop infinito e nunca parar para aceitar ou rejeitar a cadeia.
- Planeja-se mostrar que as linguagens geradas pelas gramáticas irrestritas são precisamente as linguagens recursivamente enumeráveis.

Das Gramáticas para as Máquinas de Turing

- **Teorema:** Qualquer linguagem gerada por uma gramática irrestrita é recursivamente enumerável.
- Isto pode ser provado da seguinte forma:
 - 1 Se existe um procedimento para enumerar as cadeias de uma linguagem, então a linguagem é recursivamente enumerável.
 - 2 Existe um procedimento para enumerar todas as cadeias em qualquer linguagem gerada por uma gramática irrestrita.
 - 3 Portanto, qualquer linguagem gerada por uma gramática irrestrita é recursivamente enumerável.

Das Gramáticas para as Máquinas de Turing

- Prova-se que a linguagem é recursivamente enumerável construindo uma Máquina de Turing para aceitar qualquer cadeia w da linguagem.
 - Construa uma máquina de Turing que “gere” as cadeias da linguagem em alguma ordem sistemática.
 - Construa uma segunda máquina de Turing que compara sua entrada a w e aceita sua entrada se as duas cadeias são idênticas.
 - Construa uma máquina de Turing composta que incorpora as duas máquinas acima, usando a saída da primeira como entrada para a segunda.

Das Gramáticas para as Máquinas de Turing

- Agora, gera-se sistematicamente todas as cadeias da linguagem. Para outros tipos de gramáticas, gera-se as cadeias menores antes; não se sabe como fazer isso com uma gramática irrestrita, porque algumas produções poderiam encurtar a forma sentencial. Pode levar um milhão de passos para derivar λ .
- Ao invés disto, ordena-se as cadeias de derivação mais curta antes. Primeiro, considera-se todas as cadeias que podem ser geradas a partir de S em um passo de derivação e verifica-se se as mesmas são compostas inteiramente de terminais (pode-se fazer isso porque há apenas um número finito de produções). Então considera-se todas as cadeias que podem ser derivadas em dois passos, e assim por diante.

Das Máquinas de Turing para as Gramáticas

- Mostrou-se que uma Máquina de Turing pode fazer qualquer coisa que uma gramática irrestrita pode fazer.
- Agora, deve-se mostrar que uma gramática irrestrita pode fazer qualquer coisa que uma Máquina de Turing pode fazer.
- Isto pode ser feito usando uma gramática irrestrita para emular uma Máquina de Turing.
- Lembre-se de que uma descrição instantânea (DI) de uma Máquina de Turing é a cadeia

$$w_1 q w_2$$

onde os $w_1, w_2 \in (\Sigma')^*$ são cadeias de símbolos de fita, q é o estado corrente e a cabeça de leitura/escrita está no quadrado que contém o símbolo mais a esquerda de w_2 .

Das Máquinas de Turing para as Gramáticas

- Faz sentido que uma gramática, que é um sistema para reescrever cadeias, possa ser usada para manipular DIs, que são cadeias de símbolos.
- Uma Máquina de Turing aceita uma cadeia w se

$$q_0 w \Rightarrow^* w_1 q_a w_2$$

para $w_1, w_2 \in (\Sigma')^*$ e estado de aceitação q_a , enquanto uma gramática produz uma cadeia se

$$S \Rightarrow^* w.$$

- Como a máquina de Turing começa com w e a derivação gramatical termina com w , a gramática construída funcionará “reversamente” quando comparada à Máquina de Turing.

Das Máquinas de Turing para as Gramáticas

- As produções da gramática construída podem ser logicamente agrupadas em três conjuntos:
 - 1 **Iniciação**: Estas produções constroem a cadeia $\dots B \& w_1 q_a w_2 B \dots$ onde B indica um branco e $\&$ é uma variável especial usada para terminação;
 - 2 **Execução**: Para cada regra de transição δ necessita-se uma produção correspondente;
 - 3 **Limpeza**: A derivação deixará alguns símbolos q_0 , B e $\&$ na cadeia (juntamente com a cadeia w), tal que são necessárias algumas produções adicionais para limpá-los.

Das Máquinas de Turing para as Gramáticas

- Para os símbolos terminais Σ da gramática, usa-se o alfabeto de fita Σ da Máquina de Turing (mesmo alfabeto).
- Para as variáveis V da gramática, usa-se:
 - $\Sigma' - \Sigma$, ou seja, o alfabeto de fita estendido menos os símbolos terminais (alfabeto de entrada).
 - Um símbolo $q_i \in Q$ para cada estado da Máquina de Turing.
 - B (branco) e $\&$ (usado para terminação).
 - S (para símbolo inicial) e A (para iniciação).

Das Máquinas de Turing para as Gramáticas

- **Iniciação:** É necessário gerar qualquer cadeia da forma

$$B...B&w_1q_a w_2B...B$$

- Para gerar um número arbitrário de “brancos” em ambos os lados, usa-se as produções

$$S \rightarrow BS \mid SB \mid \&A$$

- Agora, usa-se o A para gerar as cadeias $w_1, w_2 \in \Sigma'$, com um estado q_a em algum lugar no meio:

$$A \rightarrow xA \mid Ax \mid q_a, \text{ para todo } x \in \Sigma'.$$

Das Máquinas de Turing para as Gramáticas

- **Execução:** Para cada regra de transição δ necessita-se de uma produção correspondente. Para cada regra da forma

$$\delta(q_i, a) = (q_j, b, S)$$

usa-se uma produção

$$q_j b \rightarrow q_i a,$$

para cada regra da forma

$$\delta(q_i, a) = (q_j, b, R)$$

usa-se uma produção

$$b q_j \rightarrow q_i a$$

Das Máquinas de Turing para as Gramáticas

- e para cada regra da forma

$$\delta(q_i, a) = (q_j, b, L)$$

usa-se uma produção

$$q_jcb \rightarrow cq_ia$$

para todo $c \in \Sigma'$ (a assimetria é devido ao símbolo à direita de q ser o símbolo sob a cabeça de leitura/escrita da Máquina de Turing.)

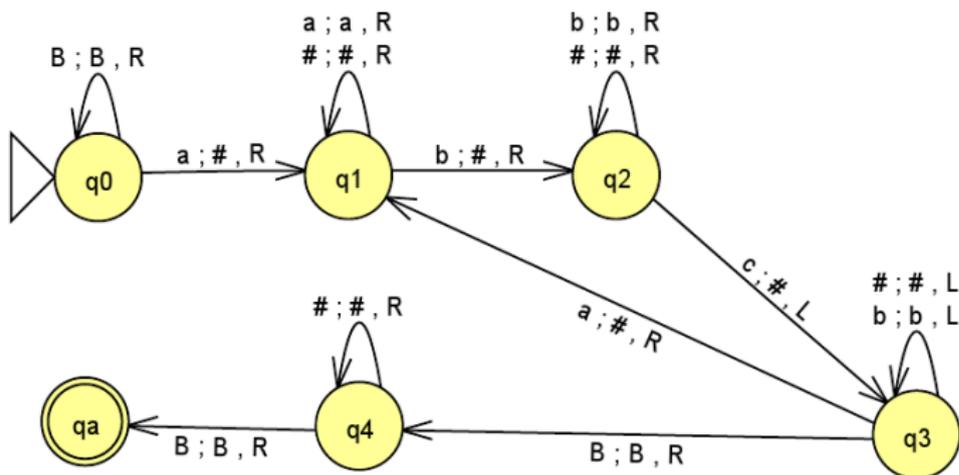
- **Limpeza:** Termina-se com uma cadeia que se parece com $B...B&q_0wB...B$, tal que são necessárias produções para se livrar de tudo menos do w :

$$B \rightarrow \lambda$$
$$&q_0 \rightarrow \lambda$$

Linguagem $\{a^n b^n c^n \mid n > 0\}$: Máquina de Turing

- Seja a seguinte Máquina de Turing:

Figure: Uma máquina de Turing para processar a linguagem $\{a^n b^n c^n \mid n > 0\}$.



Linguagem $\{a^n b^n c^n \mid n > 0\}$: Gramática

- Produções de Iniciação:

1(a) $S \rightarrow BS$

1(b) $S \rightarrow SB$

1(c) $S \rightarrow \&A$

1(d) $A \rightarrow aA$

1(e) $A \rightarrow Aa$

1(f) $A \rightarrow bA$

1(g) $A \rightarrow Ab$

1(h) $A \rightarrow cA$

1(i) $A \rightarrow Ac$

1(j) $A \rightarrow A\#$

1(k) $A \rightarrow \#A$

1(l) $A \rightarrow AB$

1(m) $A \rightarrow BA$

1(n) $A \rightarrow q_a$

Linguagem $\{a^n b^n c^n \mid n > 0\}$: Gramática

- Produções de Execução:

nro	δ	produção
2(a)	(q_0, B, q_0, B, R)	$Bq_0 \rightarrow q_0B$
2(b)	$(q_0, a, q_a, \#, R)$	$\#q_1 \rightarrow q_0a$
2(c)	(q_1, a, q_1, a, R)	$aq_1 \rightarrow q_1a$
2(d)	$(q_1, \#, q_1, \#, R)$	$\#q_1 \rightarrow q_1\#$
2(e)	$(q_1, b, q_2, \#, R)$	$\#q_2 \rightarrow q_1b$
2(f)	(q_2, b, q_2, b, R)	$bq_2 \rightarrow q_2b$
2(g)	$(q_2, \#, q_2, \#, R)$	$\#q_2 \rightarrow q_2\#$
2(h)	$(q_3, a, q_1, \#, R)$	$\#q_1 \rightarrow q_3a$
2(i)	(q_3, B, q_4, B, R)	$Bq_4 \rightarrow q_3B$
2(j)	$(q_4, \#, q_4, \#, R)$	$\#q_4 \rightarrow q_4\#$
2(k)	(q_4, B, q_a, B, R)	$Bq_a \rightarrow q_4B$

Linguagem $\{a^n b^n c^n \mid n > 0\}$: Gramática

nro	δ	produções
2(l) 2(m) 2(n) 2(o) 2(p)	$(q_2, c, q_3, \#, L)$	$q_3 a \# \rightarrow a q_2 c$ $q_3 b \# \rightarrow b q_2 c$ $q_3 c \# \rightarrow c q_2 c$ $q_3 B \# \rightarrow B q_2 c$ $q_3 \# \# \rightarrow \# q_2 c$
2(q) 2(r) 2(s) 2(t) 2(u)	$(q_3, \#, q_3, \#, L)$	$q_3 a \# \rightarrow a q_3 \#$ $q_3 b \# \rightarrow b q_3 \#$ $q_3 c \# \rightarrow c q_3 \#$ $q_3 B \# \rightarrow B q_3 \#$ $q_3 \# \# \rightarrow \# q_3 \#$
2(v) 2(w) 2(x) 2(y) 2(z)	(q_3, b, q_3, b, L)	$q_3 a b \rightarrow a q_3 b$ $q_3 b b \rightarrow b q_3 b$ $q_3 c b \rightarrow c q_3 b$ $q_3 B b \rightarrow B q_3 b$ $q_3 \# b \rightarrow \# q_3 b$

Linguagem $\{a^n b^n c^n \mid n > 0\}$: Gramática

- Produções de Limpeza:

3(a) $B \rightarrow \lambda$

3(b) $\&q_0 \rightarrow \lambda$

- Gramática:

- $\Sigma = \{a, b, c\}$

- $V = \{S, A, \&, q_0, q_1, q_2, q_3, q_4, q_a, \#, B\}$

- $S = S$

- Produções: apresentadas anteriormente

- Observe que neste caso $\Sigma' = \Sigma \cup \{B, \#\}$.

Exemplo: cadeia *aabbcc*

- $q_0 aabbcc \Rightarrow_{2b} \#q_1 abbcc \Rightarrow_{2c} \#aq_1 bbcc \Rightarrow_{2e} \#a\#q_2 bcc \Rightarrow_{2f}$
 $\#a\#bq_2 cc \Rightarrow_{2l} \#a\#q_3 b\#c \Rightarrow_{2v} \#aq_3 \#b\#c \Rightarrow_{2q}$
 $\#q_3 a\#b\#c \Rightarrow_{2h} \#\#q_1 \#b\#c \Rightarrow_{2d} \#\#\#q_1 b\#c \Rightarrow_{2e}$
 $\#\#\#\#q_2 \#c \Rightarrow_{2g} \#\#\#\#\#q_2 c \Rightarrow_{2l} \#\#\#\#\#q_3 \#\# \Rightarrow_{2q}^*$
 $q_3 B \#\#\#\#\#\# \Rightarrow_{2i} q_4 \#\#\#\#\#\# \Rightarrow_{2j}^* \#\#\#\#\#\# q_4 B \Rightarrow_{2k}$
 $\#\#\#\#\#\# Bq_a$
- $S \Rightarrow_{1b} SB \Rightarrow_{1c} \&AB \Rightarrow_{1m} \&BAB \Rightarrow_{1k}^* \&B \#\#\#\#\#\# AB \Rightarrow_{1m}$
 $\&B \#\#\#\#\#\# BAB \Rightarrow_{1n} \&B \#\#\#\#\#\# Bq_a \Rightarrow_{2k}$
 $\&B \#\#\#\#\#\# q_4 B \Rightarrow_{2j}^* \&Bq_4 \#\#\#\#\#\# \Rightarrow_{2i}$
 $\&q_3 B \#\#\#\#\#\# \Rightarrow_{2t} \&Bq_3 \#\#\#\#\#\# \Rightarrow_{2u}^*$
 $\&B \#\#\#\#\# q_3 \#\#\# B \Rightarrow_{2p} \&B \#\#\#\#\# q_2 c B \Rightarrow_{2g}$
 $\&B \#\#\#\# q_2 \#c \Rightarrow_{2e} \&B \#\#\#\# q_1 b \#c \Rightarrow_{2d} \&B \#\#\# q_1 \#b \#c \Rightarrow_{2h}$
 $\&B \#q_3 a \#b \#c \Rightarrow_{2q} \&B \#aq_3 \#b \#c \Rightarrow_{2z} \&B \#a \#q_3 b \#c \Rightarrow_{2m}$
 $\&B \#a \#bq_2 cc \Rightarrow_{2f} \&B \#a \#q_2 bcc \Rightarrow_{2e} \&B \#aq_1 bbcc \Rightarrow_{2c}$
 $\&B \#q_1 abbcc \Rightarrow_{2b} \&Bq_0 aabbcc \Rightarrow_{3a} \&q_0 aabbcc \Rightarrow_{3b} aabbcc$

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Funções Numéricas: Notação Unária

- Sempre se pensou apenas nas máquinas de Turing como aceitadores de linguagem.
- É também importante usar estas máquinas como dispositivos que computam funções numéricas, isto é, que mapeiam $\mathbb{N}^k \rightarrow \mathbb{N}$.
- Pretende-se codificar o conjunto dos números naturais na notação unária.
- Então o código para 0 é 1, o código para 1 é 11, 2 é 111, 3 é 1111, etc.
- Escreve-se n^u para simbolizar o n codificado em unário.

Funções Numéricas: Notação Unária

- Usando a notação unária, pode-se dar uma semântica teórico-numérica para as máquinas de Turing.
- Ou seja, dada uma máquina de Turing sobre um alfabeto que inclui o símbolo 1, pode-se dizer como interpretar o comportamento de uma máquina de Turing tal que ela possa ser pensada como um dispositivo que computa uma função teórico-numérica.
- Como se verá, uma única máquina de Turing de acordo com a convenção adotada realmente computa uma função teórico-numérica (diferente) $\mathbb{N}^k \rightarrow \mathbb{N}$ para toda aridade k .

Funções Numéricas: Notação Unária

- **Definição:** Uma máquina de Turing M computa uma função $\varphi_M^{(k)}$ de aridade k como se segue.
 - Na entrada (n_1, \dots, n_k) , n_1, \dots, n_k são colocados na fita de M em unário, separados por brancos simples.
 - A cabeça de M é colocada sobre o 1 mais a esquerda de n_1^u , e o controle de estados finitos de M é colocado em q_0 .
 - Em outras palavras, M tem DI inicial

$$q_0 n_1^u B n_2^u B \dots B n_k^u$$

- Se e quando M terminar o processamento, os 1's na fita são contados e seu total é o valor de $\varphi_M^{(k)}(n_1, \dots, n_k)$.
- Se M nunca pára, diz-se que $\varphi_M^{(k)}(n_1, \dots, n_k)$ é indefinido.
- Refere-se a $\varphi_M^{(k)}$ como a (k -ésima) semântica de M .

Funções Numéricas: Notação Unária

- **Exemplo:** A máquina de Turing sem nenhuma quintupla (isto é, ela pára qualquer que seja a DI) computa a função sucessora $\varphi^{(1)}(n) = s(n) = n + 1$.
 - Entretanto, deve-se checar que, como uma calculadora para uma função de duas variáveis, ela computa a função $\varphi^{(2)}(x, y) = x + y + 2$.
- **Exemplo:** Considere a seguinte máquina de Turing.

$(q_0 \ 1 \ q_1 \ 1 \ R)$

$(q_1 \ 1 \ q_0 \ 1 \ R)$

$(q_1 \ B \ q_1 \ B \ R)$

Esta máquina de Turing computa a seguinte função de uma variável

$$\begin{aligned}\varphi_M^{(1)}(n) &= n + 1, \text{ se } n \text{ é ímpar;} \\ &= \perp, \text{ caso contrário}\end{aligned}$$

Funções Numéricas: Notação Unária

- Isto é, aqui a semântica de M é uma função parcial: para algumas entradas, um valor é retornado, mas para outras - neste caso, todos os argumentos pares - a função é indefinida.
- Este fenômeno é um fato inegável da vida em ciência da computação.
- Existem programas perfeitamente legais em qualquer linguagem de programação (suficientemente rica) que falha ao retornar valores para algumas ou possivelmente todas as entradas.
- Isto é porque a noção matemática correspondente de uma função parcial é o estabelecimento apropriado para a teoria de algoritmos abstrata e as funções que eles computam.

Funções Parciais e Totais

- **Definição:** Uma **função parcial** é uma função que pode ou não ser definida para todos os seus argumentos.
 - Especificamente, onde uma função “ordinária” $g : A \rightarrow B$ atribui um valor $g(x)$ em B para cada x em A , uma função parcial $\varphi : A \rightarrow B$ atribui um valor $\varphi(x)$ apenas para os x 's em algum subconjunto $dom(\varphi)$ de A , chamado de **domínio** da definição de φ .
 - Se $x \notin dom(\varphi)$, diz-se que φ é **indefinido** ou não especificado para aquele valor.
 - Note que deve-se referir a A como o domínio de $\varphi : A \rightarrow B$, e a B como o **contra-domínio**.
 - O conjunto $\{\varphi(x) | x \in dom(\varphi)\}$ é chamado de **faixa** de φ e é denotado por $\varphi(A)$.
 - Se o $dom(\varphi) = A$, isto é, φ atribui um valor em B para todo $x \in A$, então φ é chamado de **função total**.

Funções Parciais e Totais

- A mínima função parcial definida é a função vazia.
- É escrita \perp , com o $dom(\perp) = \emptyset$, tal que $\perp(n) = \perp$ para todo n em \mathbb{N} .
- As máximas funções parciais definidas são as funções totais, tal como a função sucessora, $s(n) = n + 1$, para todo n em \mathbb{N} .
- Entre elas há funções parciais definidas parcialmente, tal como a função computada no Exemplo 5.
- **Definição:** Uma função parcial $\varphi : \mathbb{N} \rightarrow \mathbb{N}$ é Turing-computável se ela for $\varphi_M^{(1)}$ para alguma máquina de Turing.

Funções Turing-computáveis

- Agora pode-se falar sobre as funções teórico-numéricas PASCAL-computáveis, as funções teórico-numéricas C-computáveis, etc.
- Qual é o relacionamento entre estas classes?
- Conclui-se que estas classes de funções coincidem com as funções Turing-computáveis.
- Depois de meio século de estudos detalhados de vários sistemas de computação e as funções que eles computam, achou-se que as funções computáveis são invariantes ao longo de uma grande faixa de diferentes mecanismos de definição - cada sistema formal estudado foi mostrado computar ou todas as funções Turing-computáveis ou algum subconjunto delas.

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - **A Tese de Church-Turing**
 - A Máquina Universal

A Tese de Church

- Isto levou o lógico matemático americano Alonzo Church a formular a **tese de Church**, que diz que **todos os mecanismos de computação suficientemente poderosos definem a mesma classe de funções computáveis**.
- Um conceito familiar em ciência da computação é que quando um computador, M , for suficientemente de “propósito geral”, um programa escrito em qualquer outra máquina pode ser recodificado para fornecer um programa para M que computará a mesma função.

O Resultado de Turing

- Apresenta-se um resultado de 1936 devido ao matemático inglês A. M. Turing que antecipa o computador digital por quase uma década e ainda carrega a idéia inicial da sentença anterior: ou seja, que existe uma máquina de Turing U que é universal, no sentido de que o comportamento de qualquer outra máquina M pode ser codificado como uma cadeia $e(M)$ tal que U processará qualquer cadeia da forma $(e(M), w)$ da forma como w seria processado por M ; diagramaticamente, significa

$$\text{se } w \Rightarrow_M^* w' \text{ então } (e(M), w) \Rightarrow_U^* w'$$

Sumário

- 1 Gramáticas e Linguagens Sensíveis ao Contexto
 - Gramáticas e Linguagens Sensíveis ao Contexto
 - O Lema da Cadeia Vazia
 - Prova do Lema da Cadeia Vazia
- 2 Máquinas de Turing
 - Máquinas de Turing e a Computabilidade
 - Autômatos Limitados Linearmente
 - Gramáticas Irrestritas
- 3 A Máquina de Turing Universal
 - A Máquina de Turing e Funções Numéricas [5]
 - A Tese de Church-Turing
 - A Máquina Universal

Enumeração das Máquinas de Turing

- Antes de construir a máquina universal U , é necessário mostrar como enumerar todas as descrições da máquina de Turing.
- Por quê? Porque uma máquina de Turing é apresentada como uma lista de quintuplas, a enumeração deve ser uma listagem de listas de quintuplas.
- Além disso, a enumeração será feita de uma forma efetiva, tal que dado um inteiro k pode-se algoritmicamente achar a k -ésima máquina de Turing, e dado uma máquina de Turing, pode-se algoritmicamente achar k , sua posição na enumeração.

Enumeração das Máquinas de Turing

- Começa-se descrevendo uma versão especial do “programa vazio,” uma máquina de Turing que é indefinida para todas as aridades e todas as entradas.

$$\begin{aligned} & (q_0 \ 1 \ q_0 \ 1 \ R) \\ & (q_0 \ B \ q_0 \ B \ R) \end{aligned}$$

- Claramente, para qualquer entrada envolvendo apenas 1's e brancos, este programa “roda” para sempre.
- Denota-se esta máquina como M_R , já que ela sempre se move para a direita.

Enumeração das Máquinas de Turing

- Agora, fornece-se a listagem sistemática de todas as máquinas de Turing:

$$M_0, M_1, \dots, M_k, \dots$$

assumindo que os símbolos da fita são apenas B e 1, e todos os estados são codificados na forma q_k onde k é um número natural escrito na notação decimal.

- Aqui, a máquina M_k é determinada como se segue.
- Associa-se um código parecido com ASCII com todo símbolo que pode aparecer em uma quintupla.

Enumeração das Máquinas de Turing

- O seguinte quadro dá uma destas combinações.

100000	0
100001	1
100010	2
...	
101001	9
110000	q
110001	1 (o símbolo da fita)
110010	B
111000	R
111001	L
111010	S

Enumeração das Máquinas de Turing

- Usando este código pode-se associar uma cadeia binária com qualquer quintupla simplesmente concatenando as cadeias de 6 bits associadas com cada símbolo. Por exemplo:

$$(q_2 \ 1 \ q_{11} \ B \ L)$$

110000:100010:110001:110000:100001:100001:110010:111001
q : 2 : 1 : q : 1 : 1 : B : L

(Os dois-pontos “:” não são parte da cadeia - estão aqui apenas para ajudar a leitura.)

Enumeração das Máquinas de Turing

- Uma vez estabelecida uma codificação para as quintuplas, pode-se codificar uma máquina de Turing completa sem ambigüidade através da concatenação de cadeias de bits de suas quintuplas individuais.
- A cadeia concatenada resultante, interpretada como um número binário, é o código da máquina de Turing.
- O número natural n é uma descrição de máquina legal se ele corresponde a um conjunto de quintuplas que são determinísticas no sentido descrito na seção anterior.

Enumeração das Máquinas de Turing

- Portanto, tem-se a seguinte listagem de máquinas de Turing,

$$M_0, M_1, \dots, M_n, \dots$$

onde M_n é a máquina com código binário n , se n for uma descrição de máquina legal, e a máquina fixa M_R para a função vazia, caso contrário. Chama-se n o índice da máquina M_n . (M_R aparece muito freqüentemente na lista.)

- Além da listagem das máquinas de Turing pode-se falar sobre uma listagem das funções Turing-computáveis.

Enumeração das Máquinas de Turing

- As funções Turing-computáveis de uma variável são enumeradas como se segue:

$$\varphi_0, \varphi_1, \dots, \varphi_n, \dots$$

onde φ_n é a função de uma variável $\varphi_{M_n}^{(1)}$ computada pela máquina de Turing M_n .

- Antes de apresentar o resultado principal, considere várias observações sobre a listagem da máquina de Turing 8 e a listagem de funções computáveis 9.
- Primeiro, a listagem demonstra que há muitas máquinas de Turing e funções associadas.
- Segundo, toda função Turing-computável aparece freqüentemente na listagem φ_n .

Enumeração das Máquinas de Turing

- Isto acontece porque, dada qualquer máquina M_n , considere o conjunto de estados de M_n , Q .
- Para qualquer estado $p \notin Q$ (e há infinitamente muitos destes), considere a máquina consistindo de M_n e da quintupla adicional $(p B p B R)$.
- Esta nova máquina, M' , faz exatamente o que M_n faz pois o estado p nunca pode ser alcançado.
- Mas para cada escolha do estado p , M' tem um índice diferente.
- Finalmente, note que listou-se apenas as funções de uma variável computadas pelas máquinas de Turing.
- Pode-se dar uma listagem também para as funções de k variáveis para qualquer $k > 1$. Escreve-se como $\varphi_n^{(k)}$.

A Máquina Universal

- Considere novamente a máquina de Turing universal U .
- Primeiro, mostra-se que não há perda de generalidade na simulação de máquinas tendo apenas dois símbolos, B e 1 , em seus alfabetos de entrada.
- Segundo, desenha-se uma máquina universal U^Δ que usa três cabeças de fita para acompanhar sua tarefa.
- Finalmente, mostra-se que U^Δ pode ser substituída por uma U de cabeça única, ainda que sofrendo um grande aumento no tempo de computação.
- O primeiro estágio é simples.
- Se cada símbolo de um alfabeto de fita grande for substituído por uma cadeia distinta de m símbolos de $\{1, B\}^m$, pode-se simular a máquina original M por uma nova máquina M^Δ que simula um único movimento de M em no máximo $3m$ movimentos.

A Máquina Universal

- Pense na fita da máquina como dividida em blocos de m quadrados consecutivos, tal que cada bloco contenha o código para um único símbolo do alfabeto original Σ' (e com B^m codificando o B de Σ').
- Então a nova máquina M^Δ começa seu ciclo de simulação no quadrado mais a esquerda de um bloco.
- Ela então lê à direita m passos para inferir o símbolo de Σ' codificado.
- Depois, armazena o estado novo desejado de M e completa seu ciclo não movendo sua cabeça (para simular um R) ou movendo $2m$ quadrados a esquerda (para simular um L).
- Como ela codifica diretamente este processo na forma de quintuplas, tem-se o seguinte:

A Máquina Universal

- **Lema:** Dada qualquer máquina de Turing $M = (Q, \Sigma, q_0, q_a, \delta)$ e qualquer mapa um por um $h : \Sigma \rightarrow \{1, B\}^m$ que envia B para B^m , existe uma máquina de Turing

$$M^\Delta = \{Q^\Delta, \{1\}, q'_0, q'_a, \delta^\Delta\}$$

que simula M de uma forma precisa: existe uma codificação $h_q : Q \rightarrow Q^\Delta$ de estados de M tal que para toda DI $\alpha q \beta$ de M pode-se obter a codificação de seu sucessor em $3m$ passos pelo uso de M^Δ ; isto é

$$\delta_M(\alpha q \beta) = \alpha' q' \beta' \Rightarrow \delta_{M^\Delta}^{3m}(h(\alpha)h_q(q)h(\beta)) = h(\alpha')h_q(q')h(\beta')$$

- **PROVA:** Por uniformidade, faz-se o “ciclo” de M^Δ para $2m$ passos quando simula-se um movimento a direita tal que a simulação de um passo- M sempre leva $3m$ passos- M^Δ . ♦

A Máquina Universal

- Vai-se mostrar agora como qualquer máquina de Turing M com um alfabeto de 2 símbolos pode ser simulada por uma máquina de Turing U^Δ que tem três cabeças (H_1 , H_2 e H_3) com as quais ela pode percorrer as fitas (que convenientemente representa-se como três trilhas de uma única fita).
- A idéia é que H_1 seja situada na primeira trilha da fita de tal forma que a cabeça de M se situe na sua fita binária.
- U^Δ então consulta as quintuplas de M , usando H_2 para ler sua codificação na trilha 2, para comandar o comportamento de H_1 na trilha 1; usa H_3 na trilha 3 para computações subsidiárias requeridas para reposicionar H_2 na codificação de quintupla correta para o próximo ciclo de simulação.

A Máquina Universal

- Suponha uma máquina com p cabeças percorrendo uma única fita na qual são impressos símbolos do alfabeto Σ' .
- A qualquer tempo a caixa de controle estará no estado q de Q e receberá como entrada os p símbolos percorridos pelas suas cabeças, isto é, um elemento de $(\Sigma')^p$.
- A saída da unidade de controle é um elemento de $((\Sigma')^p \times M) \cup \{pare\}$, onde $M = I \mid I$ é uma instrução possível às cabeças para mover ao máximo a distância 1.
- Então, a máquina de Turing é especificada pelas quintuplas

$$q_i \ x_j \ q_l \ x_k \ I$$

A Máquina Universal

- com a única diferença de que os x 's e os l 's são “vetores,” e que se deve empregar uma convenção para resolver conflitos se duas cabeças tentam imprimir símbolos diferentes num único “quadrado.”
- Uma computação de tal máquina começa com a atribuição de um estado à unidade de controle e a atribuição das posições iniciais para as cabeças.
- Como de costume, é assumido que a fita tem no máximo finitamente muitos quadrados não brancos.
- Então a computação prossegue normalmente, parando quando e apenas quando nenhuma quintupla começando com $q_i x_j$ é aplicável.

A Máquina Universal

- A tarefa é mostrar que tal computação pode ser simulada, de forma adaptável, em uma máquina de Turing “ordinária” (isto é, com $p = 1$) tal que o número de passos necessários para tal simulação é limitado.
- **Lema:** Suponha que uma máquina de Turing generalizada M^Δ tenha (a) p rastreadores em uma única fita de uma dimensão ou (b) um rastreador em cada uma das p fitas. Então M^Δ pode ser simulada por uma máquina de Turing ordinária M de tal forma que um único passo de M^Δ , quando a porção ativa de sua fita for de n quadrados, pode ser simulada em no máximo $2n + 2p$ passos.

A Máquina Universal

- Note que este resultado suporta a tese de Church de que qualquer algoritmo codificado em qualquer linguagem de computador, real ou imaginária, é em princípio programável por máquinas de Turing.
- Imagine que uma fita da máquina de Turing é uma palavra de memória: de fato, uma palavra de memória não limitada.
- Agora suponha uma máquina de fita de 512K/cabeça.
- Se a fita 1 é usada como um armazenamento de massa infinito e as outras fitas são meramente consideradas como palavras de memória, então haverá uma máquina com uma linguagem de máquina esquisita mas útil e uma memória infinita.
- Tal linguagem é adequada para tornar real um algoritmo escrito em qualquer linguagem de computador.

A Máquina Universal

- **Teorema: (O Teorema da Máquina Universal).** Existe uma máquina de Turing universal U tal que

$$\varphi_U^{(2)}(x, y) = \varphi_x^{(1)}(y)$$

- O teorema da máquina universal estabelece que existe uma única máquina de Turing que, quando considerada como um dispositivo que calcula uma função de duas variáveis, age como um interpretador: ela trata seu primeiro argumento como um código de programa e aplica este código ao seu segundo argumento.
- Note que Turing publicou este resultado em 1936 (quando ele tinha 24 anos), cerca de 8 anos antes de o primeiro computador eletrônico programado ser construído, e bem antes de um interpretador real ter sido projetado.

Linguagens do tipo 0

- **Teorema:** Uma linguagem é do tipo 0 se e somente se ela for o conjunto de aceitação de alguma máquina de Turing.

Bibliografia I



[1] Chomsky, N.

On Certain Formal Properties of Grammars.
Information and Control, 2, 1959, pp. 137–167.



[2] Hopcroft, J. E., Ullman, J. D.

Formal Languages and Their Relation to Automata.
Addison-Wesley Publishing Company, 1969.



[3] Hopcroft, J. E., Ullman, J. D. e Motwani, R.

*Introdução à Teoria de Autômatos, Linguagens e
Computação*.

Tradução da segunda edição americana. Editora Campus,
2003.

Bibliografia II



[4] Matuszek, D.

Definition of Unrestricted Grammars, 1996.

<http://www.seas.upenn.edu/~cit596/notes/dave/ungram1.html>



[5] Moll, R. N., Arbib, M. A., and Kfoury, A. J.

An Introduction to Formal Language Theory.

Springer-Verlag, 1988.



[6] Rosa, J. L. G.

SCE-0185 - Teoria da Computação e Linguagens Formais.

Slides. Ciências de Computação. Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, 2008.

Bibliografia III



[7] Turing, A.M.

On Computable Numbers, with an Application to the Entscheidungsproblem.

Proceedings of the London Mathematical Society, 2 42: 230-65, 1937.



[8] Página do artista Jin Wicked.

<http://www.jinwicked.com/>



[9] Wikipedia - The Free Encyclopedia.

http://en.wikipedia.org/wiki/Turing_machine