UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br
Departamento de Ciências de Computação

SCC-205 - Capítulo 4
Linguagens Recursivamente Enumeráveis e Máquinas de Turing

João Luís Garcia Rosa[1]

[1]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
http://www.icmc.usp.br/~joaoluis

2009

Sumário

Sumário

Definição [3]

- **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.

Sumário

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.

Sumário

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 ...$B\&w_1 q_a w_2 B$... 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

$$B...B\&w_1 q_a w_2 B...B$$

- 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

$$q_j cb \rightarrow cq_i a$$

  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:

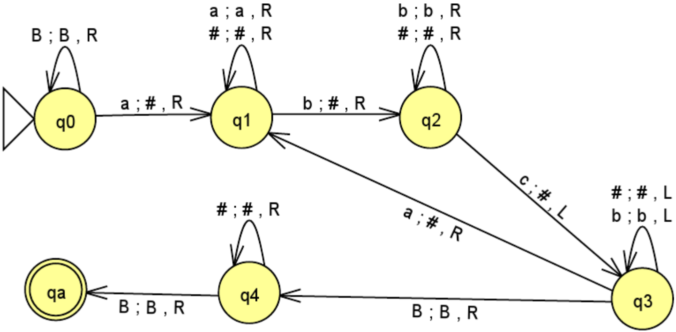

Figure: Uma máquina de Turing para processar a linguagem $\{a^n b^n c^n \mid n > 0\}$.

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

- Produções de Iniciação:

1(a) $S \to BS$
1(b) $S \to SB$
1(c) $S \to \&A$
1(d) $A \to aA$
1(e) $A \to Aa$
1(f) $A \to bA$
1(g) $A \to Ab$
1(h) $A \to cA$
1(i) $A \to Ac$
1(j) $A \to A\#$
1(k) $A \to \#A$
1(l) $A \to AB$
1(m) $A \to BA$
1(n) $A \to q_a$

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)	$Bq_0 \to q_0 B$
2(b)	$(q_0, a, q_a, \#, R)$	$\#q_1 \to q_0 a$
2(c)	(q_1, a, q_1, a, R)	$aq_1 \to q_1 a$
2(d)	$(q_1, \#, q_1, \#, R)$	$\#q_1 \to q_1 \#$
2(e)	$(q_1, b, q_2, \#, R)$	$\#q_2 \to q_1 b$
2(f)	(q_2, b, q_2, b, R)	$bq_2 \to q_2 b$
2(g)	$(q_2, \#, q_2, \#, R)$	$\#q_2 \to q_2 \#$
2(h)	$(q_3, a, q_1, \#, R)$	$\#q_1 \to q_3 a$
2(i)	(q_3, B, q_4, B, R)	$Bq_4 \to q_3 B$
2(j)	$(q_4, \#, q_4, \#, R)$	$\#q_4 \to q_4 \#$
2(k)	(q_4, B, q_a, B, R)	$Bq_a \to q_4 B$

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

nro	δ	produções
2(l)	$(q_2, c, q_3, \#, L)$	$q_3 a\# \rightarrow aq_2 c$
2(m)		$q_3 b\# \rightarrow bq_2 c$
2(n)		$q_3 c\# \rightarrow cq_2 c$
2(o)		$q_3 B\# \rightarrow Bq_2 c$
2(p)		$q_3 \#\# \rightarrow \#q_2 c$
2(q)	$(q_3, \#, q_3, \#, L)$	$q_3 a\# \rightarrow aq_3\#$
2(r)		$q_3 b\# \rightarrow bq_3\#$
2(s)		$q_3 c\# \rightarrow cq_3\#$
2(t)		$q_3 B\# \rightarrow Bq_3\#$
2(u)		$q_3 \#\# \rightarrow \#q_3\#$
2(v)	(q_3, b, q_3, b, L)	$q_3 ab \rightarrow aq_3 b$
2(w)		$q_3 bb \rightarrow bq_3 b$
2(x)		$q_3 cb \rightarrow cq_3 b$
2(y)		$q_3 Bb \rightarrow Bq_3 b$
2(z)		$q_3 \#b \rightarrow \#q_3 b$

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

- Produções de Limpeza:

  3(a) $B \to \lambda$
  3(b) $\&q_0 \to \lambda$

- 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*

- $q_0aabbcc \Rightarrow_{2b} \#q_1abbcc \Rightarrow_{2c} \#aq_1bbcc \Rightarrow_{2e} \#a\#q_2bcc \Rightarrow_{2f} \#a\#bq_2cc \Rightarrow_{2l} \#a\#q_3b\#c \Rightarrow_{2v} \#aq_3\#b\#c \Rightarrow_{2q} \#q_3a\#b\#c \Rightarrow_{2h} \#\#q_1\#b\#c \Rightarrow_{2d} \#\#\#q_1b\#c \Rightarrow_{2e} \#\#\#\#q_2\#c \Rightarrow_{2g} \#\#\#\#\#q_2c \Rightarrow_{2l} \#\#\#\#q_3\#\# \Rightarrow^*_{2q} q_3B\#\#\#\#\#\# \Rightarrow_{2i} q_4\#\#\#\#\#\# \Rightarrow^*_{2j} \#\#\#\#\#\#q_4B \Rightarrow_{2k} \#\#\#\#\#\#Bq_a$
- $S \Rightarrow_{1b} SB \Rightarrow_{1c} \&AB \Rightarrow_{1m} \&BAB \Rightarrow^*_{1k} \&B\#\#\#\#\#\#AB \Rightarrow_{1m} \&B\#\#\#\#\#\#BAB \Rightarrow_{1n} \&B\#\#\#\#\#\#Bq_a \Rightarrow_{2k} \&B\#\#\#\#\#\#q_4B \Rightarrow^*_{2j} \&Bq_4\#\#\#\#\#\# \Rightarrow_{2i} \&q_3B\#\#\#\#\#\# \Rightarrow_{2t} \&Bq_3\#\#\#\#\#\# \Rightarrow^*_{2u} \&B\#\#\#\#q_3\#\#B \Rightarrow_{2p} \&B\#\#\#\#\#q_2cB \Rightarrow_{2g} \&B\#\#\#\#q_2\#c \Rightarrow_{2e} \&B\#\#\#q_1b\#c \Rightarrow_{2d} \&B\#\#q_1\#b\#c \Rightarrow_{2h} \&B\#q_3a\#b\#c \Rightarrow_{2q} \&B\#aq_3\#b\#c \Rightarrow_{2z} \&B\#a\#q_3b\#c \Rightarrow_{2m} \&B\#a\#bq_2cc \Rightarrow_{2f} \&B\#a\#q_2bcc \Rightarrow_{2e} \&B\#aq_1bbcc \Rightarrow_{2c} \&B\#q_1abbcc \Rightarrow_{2b} \&Bq_0aabbcc \Rightarrow_{3a} \&q_0aabbcc \Rightarrow_{3b} aabbcc$

Sumário

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 $\mathbb{N}^k \to \mathbb{N}$.
- 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 teorético-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 teorético-numérica.
- Como se verá, uma única máquina de Turing de acordo com a convenção adotada realmente computa uma função teorético-numérica (diferente) $\mathbb{N}^k \to \mathbb{N}$ 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 $(n_1, ..., n_k)$, $n_1, ..., n_k$ 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
$$q_0 n_1^u B n_2^u B ... B n_k^u$$
  - Se e quando M terminar o processamento, os 1's na fita são contados e seu total é o valor de $\varphi_M^{(k)}(n_1, ..., n_k)$.
  - Se M nunca pára, diz-se que $\varphi_M^{(k)}(n_1, ..., n_k)$ 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 quíntupla (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.

$$\begin{array}{c} (q_0 \; 1 \; q_1 \; 1 \; R) \\ (q_1 \; 1 \; q_0 \; 1 \; R) \\ (q_1 \; B \; q_1 \; B \; R) \end{array}$$

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 $\varphi : \mathbb{N} \to \mathbb{N}$ é Turing-computável se ela for $\varphi_M^{(1)}$ para alguma máquina de Turing.

Funções Turing-computáveis

- Agora pode-se falar sobre as funções teorético-numéricas PASCAL-computáveis, as funções teorético-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

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

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 quíntuplas, a enumeração deve ser uma listagem de listas de quíntuplas.
- 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.

$$(q_0\ 1\ q_0\ 1\ R)$$
$$(q_0\ B\ q_0\ B\ R)$$

- 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:

  $$M_0, M_1, ..., M_k, ...$$

  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 quíntupla.

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 quíntupla 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 quíntuplas, pode-se codificar uma máquina de Turing completa sem ambigüidade através da concatenação de cadeias de bits de suas quíntuplas 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 quíntuplas que são determinísticas no sentido descrito na seção anterior.

Enumeração das Máquinas de Turing

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

$$M_0, M_1, ..., M_n, ...$$

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:

$$\varphi_0, \varphi_1, \ldots, \varphi_n, \cdots$$

  onde φ_n é a função de uma variável $\varphi^{(1)}_{M_n}$ 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 quíntupla 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

Figure: “Universal Turing Machine”, ©2003, Jin Wicked [7].

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 $U^{\triangle}$ 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.
- $U^{\triangle}$ então consulta as quíntuplas 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 quíntupla 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 quíntuplas

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

A Máquina Universal

- com a única diferença de que os x's e os I'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 quíntupla 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 $M^{\triangle}$ tenha (a) p rastreadores em uma única fita de uma dimensão ou (b) um rastreador em cada uma das p fitas. Então $M^{\triangle}$ pode ser simulada por uma máquina de Turing ordinária M de tal forma que um único passo de $M^{\triangle}$, 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.

Bibliografia I

[1] Hopcroft, J. E., Ullman, J. D.
Formal Languages and Their Relation to Automata.
Addison-Wesley Publishing Company, 1969.

[2] 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.

[3] Matuszek, D.
Definition of Unrestricted Grammars, 1996.
http://www.seas.upenn.edu/~cit596/notes/dave/ungram1.html

Bibliografia II

[4] Moll, R. N., Arbib, M. A., and Kfoury, A. J.
An Introduction to Formal Language Theory.
Springer-Verlag, 1988.

[5] Rosa, J. L. G.
Linguagens Formais e Autômatos.
Notas de Aula. Engenharia de Computação. Pontifícia Universidade Católica de Campinas, 2007.

[6] Turing, A.M.
On Computable Numbers, with an Application to the Entscheidungsproblem.
Proceedings of the London Mathematical Society, 2 42: 230-65, 1937.

Bibliografia III

[7] Página do artista Jin Wicked.
http://www.jinwicked.com/