

Máquinas que Transforman . . .
 Outros Modelos Computacionais
 Máquinas de Markov e Post
 Claudio Cesar de Sá - claudio@joinville.udesc.br
 Versão - 1.3 (16 de outubro de 2006)

Sumário

| | | |
|----------|--|-----------|
| 1 | Introdução | 1 |
| 2 | Algoritmo de Markov (Máquina de Markov - 1954) | 2 |
| 2.1 | Exemplos | 3 |
| 3 | Algoritmo de Post (Máquina ou Sistema de Post - 1943) | 7 |
| 3.1 | Exemplos | 9 |
| 3.2 | Sistema de Post | 11 |
| 3.3 | Exemplos | 13 |
| 3.4 | Post-Computável | 14 |
| 3.4.1 | Mais um Exemplo de Post-Computável | 15 |
| 4 | Equivalência de Modelos | 16 |
| 5 | Exercícios Propostos | 16 |

1 Introdução

Neste capítulo, as Máquinas de **Markov** e de **Post** são apresentadas como dois outros modelos computacionais equivalentes a TM (Máquinas de Turing), λ -Calculus e Funções Recursivas Parciais. Estes dois modelos computacionais também apresentam um formalismo e axiomático, ao conceito de computação.

Estes dois modelos, a seguir, utilizam o conceito de *Memória de Trabalho* (MT) e de *Regras de Produção* (RP). A MT (não confundir com TM, de Máquinas de Turing) é uma área ativa destinada a escrita e leitura de uma sequência de símbolos. As RPs representam a descrição da *função*, *programa*, ou *algoritmo* a ser computado.

Estes modelos são conhecidos como Máquinas de **Markov** e **Post**, e sua finalidade é transformarem palavras (“*strings*”), reescrevendo símbolos dessas palavras. Essas são sequências de símbolos de um alfabeto.

Porquê “*Strings*” (cadeias de símbolos) ?

Resposta: Como já frisado anteriormente, consegue-se descrever muitos problemas (ou quase todos) com esta representação simbólica. Em um exemplo já visto, grafos podem ser representados por uma sequência de números, virgulas e parenteses. E finalmente, o

processamento de Strings são mais fáceis que números, sob o ponto de vista do entendimento do problema.

2 Algoritmo de Markov (Máquina de Markov - 1954)

Um Algoritmo ou Máquina de Markov (MM) sobre um alfabeto A é uma sequência ordenada de produções do tipo $x \rightarrow y$ onde $x, y \in A^*$. As palavras x e y são geradas a partir de uma alfabeto A , ou pré-existentes na definição do algoritmo, como um programa de computador em sua Memória de Trabalho (memória tipo RAM). Uma sequência ordenada é o conceito convencional de programas usualmente disponíveis nos computadores atuais. Exemplo:

| | | | |
|-----|-----|---------------|-----|
| 1 | x | \rightarrow | y |
| 2 | w | \rightarrow | x |
| ... | ... | ... | ... |
| n | z | \rightarrow | k |

O funcionamento das MM é ir realizando substituições *sequenciais* segundo a ordenação do programa. Estas substituições ocorrem por um **casamento parcial** ou **total de símbolos** nas palavras entre o conteúdo da memória, e a parte condicional das regras que descrevem o algoritmo. Logo, a análise da varredura do algoritmo ocorre segundo uma *ordem*:

$$1 \succ 2 \succ 3 \succ \dots \succ n$$

Permanecendo num ciclo sobre o valor corrente do símbolo ativo na memória, retornando ao início na instrução 1 após atingir a instrução n . Este ciclo de varredura sobre as regras de produção se repete até que:

1. Caso em que nenhuma nova substituição ocorra. Isto é, ao se percorrer toda sequência de regras, nenhuma nova mudança sobre w corrente foi avaliado. Obviamente se observa uma exemplo de indecidibilidade aqui, como determinar que o valor de w não está modificando?
2. Caso uma parada seja determinada. Assim, podem existir algumas palavras com o rótulo “*pare*” ou “*halt*” para forçar uma parada do algoritmo. Contudo, esta condição não é necessária a função implementada.

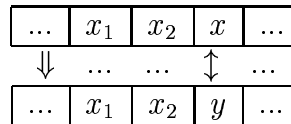
O procedimento de substituição ocorre na existência de uma string w ativa, e uma regra corrente do tipo $x \rightarrow y$, sendo que x é uma sub-sequência (“*sub-string*”) de w . Então a ocorrência mais a esquerda de x em w é sobreescrita por y , para obter um w transformado por $x \rightarrow y$. Em resumo, o casamento de w com a parte antecedente da regra $x \rightarrow y$, é parcial tal que $x \subseteq w$.

Matematicamente tem-se:

$str_1 \xrightarrow{*} str_2$, onde as MM computam sobre $A^* \rightarrow A^*$

Procedimento:

Como já descrito, há um casamento convencional da esquerda para direita em w , onde x é uma sub-palavra de w , substituindo x por y de acordo com $x \rightarrow y$. Veja o exemplo:



A varredura é retomada continuamente, até que não houver novas strings ou uma parada explícita.

Uma produção especial:

Seja uma produção do tipo $\lambda \rightarrow y$ ¹ então qualquer string w se transforma em yw .

2.1 Exemplos

1. Seja $\Sigma = \{a\}$, construir um MM que transforme strings do tipo a^i para a^{i+1} . Uma solução é dado por:

$$\boxed{\lambda \rightarrow a(\text{pare})}$$

A letra a será anexada a w , pois λ concatena com vazio. Seja o exemplo onde $w = aaa$ aplicando $\lambda \rightarrow a(\text{pare})$ em w tem-se:

$$a(\text{pare})aaa$$

2. Seja $Z = \{a, *\}$ e as seguintes sequências de regras de produção:
 1. $a * a \rightarrow *$
 2. $* a \rightarrow *$
 3. $* \rightarrow \lambda$

Sendo uma entrada $w = aa * aaa$

| <u>Entrada</u> | <u>Regra aplicada</u> |
|----------------|-------------------------|
| 1. $aa * aaa$ | w de input |
| 2. $a * aa$ | $a * a \rightarrow *$ |
| 3. $* a$ | $a * a \rightarrow *$ |
| 4. $*$ | $* a \rightarrow *$ |
| 5. λ | $* \rightarrow \lambda$ |

¹ λ ou Λ

Observa-se que MM computa o λ para todas strings do formato $a^i * a^j$ onde $i \leq j$.

Comentários

Seja uma árvore de derivação conforme a figura 2, e algumas seqüências possíveis:

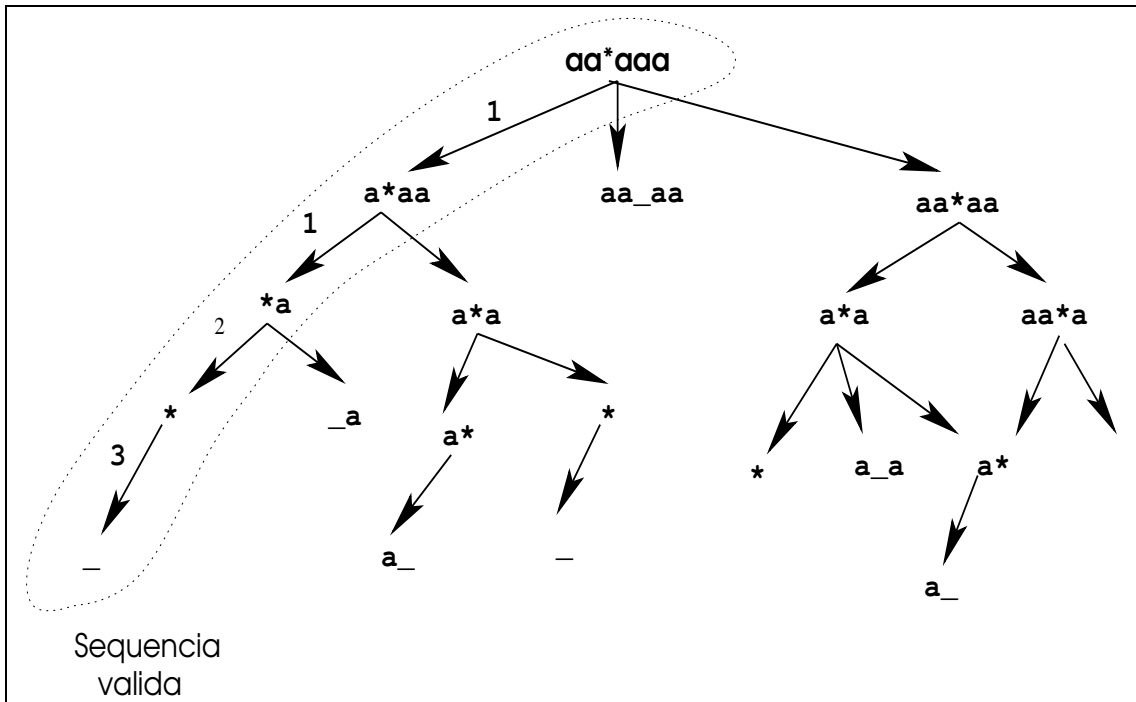


Figura 1: Árvore de derivação desse exemplo

Contudo o detalhe é: uma **seqüência ordenada de produções**. Isto é, sem variáveis (repetindo: são *símbolos*) e funcionam como constantes para servirem como marcadores ²! Na figura 2 o símbolo “_” deve ser interpretado como Λ ou λ .

Reforçando o que já foi escrito:

- (a) As produções do tipo $\lambda \rightarrow x$, acrescentam x à esquerda (e sempre à esquerda) da entrada. Assim:
 $abc \rightarrow xabc$ estaria correto e
 $abc \rightarrow abxc$ estaria incorreto de se alcançar diretamente com a produção $\lambda \rightarrow x$;
- (b) Pode-se dispor de símbolos **auxiliares** para contornar a restrição do item anterior. Estes símbolos não são *variáveis* num sentido amplo e usual, mas sim outros símbolos, que podem ser temporariamente substituídos por outros símbolos!

Também é fácil ver que M calcula a^{i-j} para todas entradas de strings na forma $a^i * a^j$, onde $i \geq j$.

²Relembre esta idéia de programação das MTs.

Logo, a^i pode ser pensado como uma representação natural de um número i . Adicionalmente, $a^i * a^j$ se transforma na função *monus*³ aplicados a i e j , tal que:

$$a^i * a^j \Rightarrow x - y = \begin{cases} a^{i-j} & \text{se } i \geq j \\ 0 & \text{caso contrário } j > i \end{cases}$$

Obs: O cálculo $x - y$ numa representação unária. Para confirmar a generalização dessa subtração unária (positiva), compute as seguintes entradas: “aaaa * a” (equivalente a: 4 – 1), e ainda “aa * aaaa”, equivalente a: 2 – 4 que para este caso retorna 0.

Como exercício faça a computação de:

- (a) “aaaa * a”;
- (b) “aaaa * aaaa”;
- (c) “*a”;
- (d) “a*”;

3. Inversão de String (ver no livro pág. 411 da referência [Hein, 1999])

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

$$\Sigma^* \rightarrow a\dots$$

$$abc \xrightarrow{*} cba$$

Deixamos a cargo do estudante o estudo deste clássico exemplo.

4. Soma Unária. Exemplificando $3 + 2 \Leftrightarrow 111 + 11$

Há várias soluções, o que reforça o conceito intuitivo de algoritmo, para um computação que use um deslocamento do sinal + para esquerda é dado por:

1. $1+ \rightarrow +1$
2. $+ \rightarrow \wedge$

Ou com deslocamento para direita do sinal +:

1. $+1 \rightarrow 1+$
2. $+ \rightarrow \wedge$

Uma regra particular para suporta a soma $1 + 0$, pode ser dado por: $1+ \rightarrow 1$, contudo, esta regra pode ser obtida a partir de uma combinação entre as regras 1 e 2.

Uma terceira solução, mas questionável sob a visão de computar símbolo a símbolo, considerando um hipotético cabeçote de leitura no símbolo mais a esquerda ou a direita, tem-se uma solução dada por em apenas uma regra de produção, do tipo:

$$1 + 1 \rightarrow 11$$

Realmente, esta regra também realiza uma soma unária, contudo não há deslocamento de “um cabeçote” de um eventual leitor na sub-string.

³Uma função de subtração em N^+ .

Para as duas definições acima, compute as entradas: $11 + 111$, $111 + 1$, $+1$. Adicionalmente, verifique que para entradas genéricas de somas, tais como: $11 + 111 + 111 + 1$ os programas acima também computam

5. Exemplo enviado pelo Fábio Teodoro, e descubra o que o programa faz:

1. $xa \rightarrow ax$
2. $axb \rightarrow aabx$
3. $xb \rightarrow bx$
4. $bx c \rightarrow bbcx$
5. $xc \rightarrow cx$
6. $cx\lambda \rightarrow cc(halt)$
7. $\lambda \rightarrow x$

Seja a computação de: $a^2b^3c^4 \xrightarrow{\#7} xaab^3c^4 \xrightarrow{\#1} \dots a^3b^4c^5(halt)$

Qual sua conclusão? É possível de ser melhorado?

6. Remoção do b mais a esquerda de uma string $\{a, b, c\}^*$

1. $xa \rightarrow ax$
2. $xc \rightarrow cx$
3. $xb \rightarrow (halt)$
4. $x\lambda \rightarrow (halt)$
5. $\lambda \rightarrow x$ (introduziu um x a string de entrada)

ou melhorando a solução com a introdução de um marcador de símbolo lido:

1. $xa \rightarrow ax$
2. $xc \rightarrow cx$
3. $xb \rightarrow \#$ (removeu o b e escreveu $\#$)
4. $\#a \rightarrow a\#$
5. $\#b \rightarrow b\#$
6. $\#c \rightarrow c\#$
7. $\#\lambda \rightarrow (halt)$
8. $\lambda \rightarrow x$ (introduziu um x a string de entrada)
9. $x\lambda \rightarrow \lambda(halt)$ (caso a palavra não contenha nenhum b)

onde $\#$ funciona como um marcador de cabeçote.

Para as versões acima, compute as entradas: $cacabbac$, ca , bbb , etc. Veja a notação utilizada em sala de aula, e alguns comentários dados por:

- Soluções simplificadas, mas sem o rigor destacado, é dado por: $b \rightarrow (halt)$;
- Para as versões de MM do exemplo acima, compute a entrada: $acabbcac$;
- Há ainda um rigor que poderia ser implementado. Considere que cada símbolo está numa célula de uma fita a exemplo de uma MT. Assim, qualquer operação sobre esta palavra, a mesma deve estar escrita em células, com seu respectivo símbolo. Logo, para o caso anterior, o marcador $\#$ deve ser substituído por $\#\#$.

Este último exemplo, reforça alguns elementos comuns entre as MM e as MT, tais como:

1. Substituição de símbolos por outros símbolos⁴;
2. Marcador de cabeçote com a introdução de novos símbolos a entrada dada;
3. Leitura e escrita sobre a palavra da entrada⁵;
4. Algum rigor extra sobre a MM foi considerada, afim de reforçar a analogia com MT.

Essas idéias evidenciam a equivalência entre os diversos modelos computacionais e seu reforço sobre a Tese de Church-Turing.

3 Algoritmo de Post (Máquina ou Sistema de Post - 1943)

O Algoritmo de Post sobre um alfabeto finito A é um conjunto de *regras de produção* que são usadas para transformar palavras. Emil Post foi pioneiro em introduzir sistemas para reescrita de cadeias de símbolos. Tal que a função computável neste algoritmo é dada por:

$$A^* \rightarrow A^*$$

As produções tem formato:

$$s \rightarrow t$$

Onde s e t tem símbolos a partir de A e a possibilidade de *variáveis*. Contudo, a restrição das ocorrências da variáveis nas regras é dada por: **uma variável X está em s , se e somente se, ela estiver em t** . Isto é, uma variável que estiver em s terá que estar em t . Subentende-se que contrário não é necessário. Exemplos:

$$\begin{aligned} sW &\rightarrow tW \\ u &\rightarrow vX \\ k \cap X &\rightarrow k \cap X \\ \neq \emptyset &\rightarrow \neq \emptyset \end{aligned}$$

Diferentemente das produções de Markov, aqui não há nenhum ordenamento em particular da aplicação das regras de produção. Igualmente, algumas produções podem ter a palavra *pare* ou “*halt*” em seu lado direito da produção.

Funcionamento:

⁴Análogo ao conceito de atribuição das linguagens de programação modernas.

⁵Literalmente esta é alterada segundo uma computação dada uma MM.

1. A computação de Post ocorre por casamento *completo*⁶ de padrões no lado esquerdo, sendo gerada uma nova string do lado direito;
2. Se a produção for uma do tipo de parada, a computação pára e a string produzida é a saída;
3. Caso contrário, tentativas são feitas até que não existam mais produções a serem computadas;
4. Caso não hajam mais casamentos $\dots \Rightarrow$ a computação pára!

Exemplo

Seja o Algoritmo de Post $A = \{a, *\}$, com uma única regra de produção⁷:

$$aX* \rightarrow X$$

Seja a entrada: “aa*”, logo $X = a$ é possível casar, tal que:

$$\begin{array}{ccc} a & a & * \\ \downarrow & \downarrow & \downarrow \\ a & X & * \end{array}$$

ou seja $aa* \rightarrow X$, como X era a (X/a) então:

$$aa* \rightarrow a$$

Logo a saída passou a ser o “a”, o qual não tem mais casamentos com a produção disponível, então a computação pára!

A idéia é que sob as entradas deriva-se uma palavra **computada** e esta é via como um *como novo teorema* do sistema. Onde algumas regras podem estar implícitas para um conjunto particular de entradas. Por exemplo, uma substituição entre X e λ (Λ) é possível, e com isto se demonstra que para alguns casos é como se o sistema tivesse uma regra do tipo $X \rightarrow \lambda$.

Para isto considere uma entrada do tipo “a*”, assim:

$$\begin{array}{l} a* \iff aX* \quad \text{assim deduz-se que: } X \equiv \lambda \\ \text{isto é:} \\ X \text{ casa com } \lambda, \text{ leia-se que } X \text{ é substituível por } \lambda \end{array}$$

Então:

$$a* \rightarrow \lambda$$

A qual foi derivada da seguinte entrada:

$$a* \xrightarrow{aX* \rightarrow X} \lambda$$

⁶Na Máquina de Markov tal casamento era parcial, ou por símbolo.

⁷Ao estudante, como exercício, faça a interpretação deste algoritmo a partir desta regra.

Logo, é permitido incluir o λ na string de entrada . Do caso anterior tem-se:

$$\lambda a^*, a \lambda^*, a^* \lambda$$

Assim os casamentos permitidos com essas entradas são dados por:

$$Xa^*, aX^*, a^* X$$

Em resumo, um casamento entre X e λ é possível, sem que isto aumente o poder de computação do sistema original. Adicionalmente, outras regras podem ser incorporadas a esse sistema, sem que a máquina original seja alterada. Apenas novas instâncias a partir da regra $aX^* \rightarrow X$, foram obtidas, tais como:

$$\begin{aligned} aa^* &\iff a \\ a^* &\iff \lambda \\ a^* \lambda^* &\iff \lambda^* \\ &\dots \end{aligned}$$

3.1 Exemplos

1. Substituir todas ocorrências de a por b em uma nova string $\{a, b, c\}^*$
Contudo, duas variações para esta Máquina de Post:

- (a) **Não-determinística** (sem uma única solução):

$$XaY \rightarrow XbY$$

Exemplificando com uma entrada dada por, “ $acab$ ”, tem-se duas alternativas de computação:

$$\begin{aligned} \text{i. } acab &\rightarrow \underbrace{ac}_X a \underbrace{b}_Y \rightarrow acbb \rightarrow \underbrace{\lambda}_X a \underbrace{cbb}_Y \rightarrow bcbb \\ \text{ii. } acab &\rightarrow \underbrace{\lambda}_X a \underbrace{cab}_Y \rightarrow bcab \rightarrow \underbrace{bc}_X a \underbrace{b}_Y \rightarrow bcbb \end{aligned}$$

Percebe-se que uma regra é o suficiente para resolver de modo não-determinístico. Contudo, tal fato pode levar há uma exploração exponencial até um resultado desejado seja encontrado, e outros descartados. Ao estudante, amplie o tamanho da entrada e constate essa afirmação.

- (b) **Determinística:**

Uma estratégia na construção das regras:

- i. Usar o símbolo # para marcar a posição à direita, quando a varredura segue da esquerda para a direita;
- ii. Usar o símbolo @ como marcador de instrução à esquerda para indicar que apenas uma instrução por vez é para ser usada.

As regras de produção neste caso são dadas por:

1. $aX \rightarrow @b\#X$
2. $bX \rightarrow @b\#X$
3. $cX \rightarrow @c\#X$
4. $@X\#aY \rightarrow @Xb\#Y$
5. $@X\#bY \rightarrow @Xb\#Y$
6. $@X\#cY \rightarrow @Xc\#Y$
7. $@X\# \rightarrow X(halt)$

Exemplificando:

$$acab \xrightarrow{r^1} @b\#cab \xrightarrow{r^6} @bc\#ab \xrightarrow{r^4} @bcb\#b \xrightarrow{r^5} @bcb\# \xrightarrow{r^7} bcb$$

Ao estudante: realize algumas outras computações para assegurar o entendimento.

2. Remoção do b mais a esquerda de uma string $\{a, b, c\}^*$. Este exemplo foi corrigido para 2005-2:

1. $bX \rightarrow X(halt)$
2. $aX \rightarrow @a\#X$
3. $cX \rightarrow @c\#X$
4. $@X\#bY \rightarrow XY(halt)$
5. $@X\#aY \rightarrow @Xa\#Y$
6. $@X\#cY \rightarrow @Xc\#Y$
7. $@X\# \rightarrow X(halt)$

Finalmente, avalie a entrada $acacbbbab$, e compare o resultado da computação com o exemplo da seção anterior.

3. Subtração Unária. Uma solução da subtração unária pode ser vista como:

1. $X1 - 1Y \rightarrow X - Y$
2. $\lambda - 1Y \rightarrow -1Y(halt)$
3. $X1 - \lambda \rightarrow X1(halt)$
4. $\lambda - \lambda \rightarrow \lambda(halt)$

Relembrando uma solução alternativa para Máquina de Markov da subtração unária:

1. $1 - 1 \rightarrow \# - \lambda$ (vazio)
2. $\lambda 1 \rightarrow 1$
3. $1 - \rightarrow 1(halt)$
4. $-1 \rightarrow -1(halt)$

Para o MP, identifique e enumere as regras utilizadas nas computações abaixo:

Ex1: $11 - 1 \implies 1 - \lambda \implies 1(halt)$

Ex2: $1 - 111 \implies \lambda - 11 \implies -11$

A exemplo do que já foi concluído anteriormente, algumas novas regras foram derivadas⁸:

$$\begin{aligned} x\lambda &\rightarrow x \\ x\lambda y &\rightarrow xy \\ \text{logo:} & \\ -\lambda &\rightarrow - \end{aligned}$$

é uma produção necessária. Finalmente, o exemplo completo da MM proposta inicialmente reescrita como:

1. $1 - 1 \rightarrow -\lambda$
2. $-\lambda \rightarrow -$
3. $\lambda 1 \rightarrow 1$
4. $1 - \rightarrow 1 \text{ (halt)}$
5. $-1 \rightarrow -1 \text{ (halt)}$

Novamente, vale uma reflexão sobre os conceitos visto sobre TM e esses dois modelos equivalentes, releia as observações feitas na sub-seção 2.1.

3.2 Sistema de Post

Algumas particularizações podem ser feitas sobre a Máquina de Post. Uma destas formalizações são dadas pelos “*Sistema Canônicos de Post*” ou simplesmente “*Sistema de Post*”. O que diferencia de uma MP, é que não há entradas para serem computadas, mas sim **axiomas**. Os axiomas são os objetos/termos/teoremas verdadeiros a priori, disponíveis para se iniciar uma computação. Podem existir um ou mais axiomas, e a cada nova palavra deduzida, esta é dita ser um **teorema**. Logo, aqui novos conjuntos de objetos/axiomas válidos são gerados e incorporados ao sistema por definição. Como é um conceito de teorema, este vem de uma *teoria* a ser estruturada sob algum *sistema formal*. Este sistema formal pensado por Post, é definido por uma quádrupla dada por:

$$SP = (V, \Sigma, Ax, RP) \tag{1}$$

A proposição inicial de Post era forte no sentido de dar um outro encaminhamento na busca da definição de um *procedimento mecânico* ou *efetivo* para problemas, proposto por Hilbert em 1900. Assim, se necessitava formalizar uma teoria, e desenvolver o seu próprio conjunto de regras, para que o mesmo desse consistência as suas computações. Embora tal demonstração de completude tenha se mostrado ineficaz pelo teorema da incompletude de Gödel, a proposta de computar números/símbolos, tornaram-a equivalente há uma Máquina

⁸Além de serem válidas sob este sistema, não aumentam a computação do mesmo.

de Turing. Esta proposta de formalização foi reusada em parte por Noam Chomsky na definição na hierarquia das gramáticas formais. Os termos da equação 1 são definidos por:

- V : Conjunto de variáveis;
- Σ : Conjunto de símbolos (a atomicidade diferencia estes das variáveis);
- Ax : Conjunto de Axiomas;
- RP : Conjunto de Regras de Produção.

A representação do formato das regras possíveis são dadas por:

$$s_1, s_2, s_3 \rightarrow t$$

ou

$$\begin{aligned} s_1 &\rightarrow t \\ s_2 &\rightarrow t \\ s_3 &\rightarrow t \end{aligned}$$

Ou seja, o termo “*Canônico*” t^9 , vem da formatação n-ária onde a aplicação das regras terminam num termo t , que é incorporado ao conjunto de novos axiomas derivados. Os termos s_i^{10} vem da especificação de uma simples variável ao lado esquerdo. Lembrar de uma GLC, que é um exemplo clássico de uma notação canônica.

Procedimento: Os axiomas são adicionados como uma ou mais strings e se combinam com as regras, sem ordenação. Tal procedimento é sumarizado pela figura 3.2.

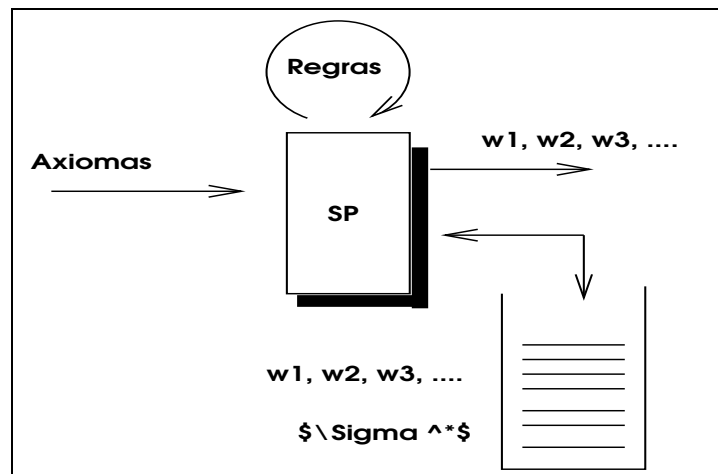


Figura 2: Esquema do Sistema Canônico de Post

⁹A notação da letra t , vem da idéia de gerar um *teorema*, que passa a ser um novo axioma do sistema.

¹⁰O uso da letra s possivelmente venha da denominação “*single*” ou “*statement*” do inglês.

3.3 Exemplos

1. Seja um Sistema de Post definido pelos seguintes elementos:

Axioma = $\{\lambda\}$, $\Sigma = \{(,)\}$, $V = \{X, Y\}$

Regras de Produção:

1. $X \rightarrow XY$
2. $Y \rightarrow XY$
3. $X \rightarrow ()$

Demonstre que o conjunto de palavras válidas neste sistema canônico é dado por:

$$\Sigma_{\text{SP}}^* = \{\lambda, (), ()(), ()()(), ()()()(), \dots\}$$

Observações:

- ✓ Acompanhe as derivações de novos teoremas deste sistema (atividade de sala de aula);
 - ✓ O axioma definido para este sistema, pode ser demonstrado a partir desse mesmo sistema? Discuta profundamente esta questão. Essa resposta traz pontos pertinentes ao teorema da Incompletude de Gödel;
 - ✓ Reavalie as questões sobre: *o que é uma teoria, o que é um axioma, o que é um teorema, o que é um sistema formal, etc*, e como tudo isto se consolida com a as máquinas vista até então;
 - ✓ Siga essas observações nos exemplos subsequentes.
2. Seja um Sistema de Post definido pelos seguintes elementos:
 Axioma = $\{\lambda\}$, $\Sigma = \{(,)\}$, $V = \{X, Y\}$
 Regras de Produção:

1. $X \rightarrow XY$
2. $Y \rightarrow XY$
3. $X \rightarrow (X)$

Demonstre que o conjunto de palavras válidas neste sistema canônico é dado por:

$$\Sigma_{\text{SP}}^* = \{\lambda, (), (()), ()()(), ()()()(), \dots\}$$

Obs: Acompanhe as derivações feitas em sala de aula.

3. Construir um Sistema de Post que gere palíndromos (pares e ímpares).

Axiomas = $\{a, b, \lambda\}$, $\Sigma = \{a, b\}$, $V = \{X\}$

Regras de Produção:

1. $X \rightarrow aXa$
2. $X \rightarrow bXb$

Exemplifique com alguns resultados, e derive o conjunto A^* gerado por este sistema, é dado por: $\{a, b, \lambda, aa, bb, aaa, aba, bab, abba, \dots\}$.

4. Do item anterior, altere o exercício no conjunto de Axiomas = $\{\lambda\}$, e compare com os conjunto acima obtido. Com esta alteração, teremos os palíndromos pares, incluindo o λ , em Σ^* . Exemplo desenvolvido em sala de aula em: 13/06/2005.

Avalie que algumas regras podem estar “*embutidas*” no sistema. Para o último exemplo, a regra $X \rightarrow \lambda$ é derivável do sistema, haja visto que λ é um axioma. Veja o casamento *completo* com a parte da esquerda das regras, e siga avaliando os novos axiomas produzidos a direita das regras.

3.4 Post-Computável

Se A é um alfabeto e $f : A^* \rightarrow A^*$ é uma função, então f é dita ser *Post-computável*, se existir um conjunto de pares que defina a função:

$$\{\langle x, f(x) \rangle \mid x \in A^*\}$$

Para simplificar, o par $\langle x, f(x) \rangle$ é representado por uma string como tipo: $x\#f(x)$, onde $\#$ é um novo símbolo que não está em A . Exemplificando estes conceitos, seja um conjunto dado pelos pares: $\{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 4 \rangle, \langle 3, 9 \rangle, \dots\}$, logo, $f(x) = x^2$. Literalmente, aqui é o conceito de **função-computável** em uma máquina formal, no caso Post.

Exemplo:

Seja uma $f(x) = xa^{11}$ sendo Post-computável sobre o alfabeto $\{a\}$ para o axioma $\#a$, com a seguinte regra de produção:

$$X\#Xa \rightarrow Xa\#Xaa$$

Este sistema de Post computa $f(x) = xa$ resultando as seguintes saídas em $f(x)$:

$$\{\#a, a\#aa, aa\#aaa, \dots\} = \Sigma^*$$

Estes pares foram gerados a partir de:

$$f(x) \Rightarrow xa \Rightarrow (\#a)a \Rightarrow (\#aa)a \Rightarrow (\#aaa)a \dots$$

¹¹Como $A = \{a\}$, então $f(a) = aa$, onde aa é o respectivo par de a . Para uma entrada aaa tem-se $aaaa$. Assim os pares deste exemplo são: $\{\langle \lambda, a \rangle, \langle a, aa \rangle, \langle aa, aaa \rangle, \langle aaa, aaaa \rangle, \dots\}$.

Logo, se percebe que o símbolo # foi usado no lugar da “,” (vírgula) do conjunto encontrado na nota: 11. Enfatizando assim, o conceito equivalente de Post-computável com regras de produção, e uma função qualquer, no caso uma que adiciona um “a” a direita de sua entrada “x”.

Exercício: encontre o conjunto acima a partir do axioma e da regra dada. Isto é, componha o conjunto de pares: $\{\langle x, f(x) \rangle | x \in A^*\}$. A $f(x)$ já foi calculada. Resta avaliar a regra $X\#Xa \rightarrow Xa\#Xaa$, gerando novos “teoremas” (palavras), a partir dos já existentes. Verifique que X/λ é derivável a partir do axioma existente inicialmente.

3.4.1 Mais um Exemplo de Post-Computável

Montar o conjunto Post-Computável para o conjunto dos palíndromos.

Axiomas: $\{\lambda\}$

Regras de produção: = {

1. $X\#X \rightarrow aXa\#aXa$
2. $X\#X \rightarrow bXb\#bXb$
3. $X\#X \rightarrow cXc\#cXc$ }

Conjunto Post-Computável:

$\Sigma_{PC}^* = \{aa\#aa, bb\#bb, bbbb\#bbbb, cc\#cc, baab\#baab, \dots\}$

Algumas computações:

1. “ $aa\#aa$ ” $\overset{*}{\rightsquigarrow} a\lambda a\#a\lambda a \equiv aa\#aa$
2. “ $baab\#baab$ ” $\overset{*}{\rightsquigarrow} aa\#aa \xrightarrow{2} baab\#baab$
3. “ $cbaabc\#cbaabc$ ” $\overset{*}{\rightsquigarrow} baab\#baab \xrightarrow{3} cbaabc\#cbaabc$

Para palíndromos ímpares deve-se trocar um dos X por “Y”, tal como:

Axiomas: $\{\lambda, a, b, c\}$

Regras de produção: = {

1. $X\#Y \rightarrow aXa\#aYa$
2. $X\#Y \rightarrow bXb\#bYb$
3. $X\#Y \rightarrow cXc\#cYc$ }

Algumas computações:

1. “ $bb\#bab$ ” $\overset{*}{\rightsquigarrow} b\lambda b\#bab \equiv bb\#bab$
2. “ $abba\#ababa$ ” $\overset{*}{\rightsquigarrow} bb\#bab \xrightarrow{1} abba\#ababa$

Logo, o conjunto $\Sigma_{PC}^* = \{ \dots \}$.

4 Equivalência de Modelos

Seriam os modelos de Markov ou Post mais poderosos que as Máquinas de Turing? Onde tanto as MT, como MM e MP que também computam palavras e funções matemáticas, há alguma mais poderosa que a outra? Não, são todas equivalentes entre si, com o mesmo poder computacional, e apresentando as mesmas deficiências como o *problema da parada!*

5 Exercícios Propostos

Máquinas de Markov: Construir algoritmos que realizem as seguintes ações:

1. Proceda um laço infinito a cada entrada de uma letra a numa string $\{a, b, c\}$. Simplifique este alfabeto se for o caso;
2. Mover um símbolo para o final da string;
3. Troque todos os a 's por b 's em qualquer string $\{a, b\}^*$;
4. A multiplicação unária. Exemplo: $111 \times 111 = 111111111$;
5. Realize a transformação de palavras do tipo: $a^i b^j c^k \xrightarrow{*} a^{i+1} b^{j+1} c^{k+1}$;

Máquinas de Post: Construir algoritmos que realizem as seguintes ações:

1. Proceda um laço infinito a cada entrada de uma letra a numa string $\{a, b, c\}$. Simplifique este alfabeto se for o caso;
2. Mover um símbolo para o final da string;
3. Sejam entradas da classe $a^n b^n$ com $n \geq 0$ e que esta entrada seja reduzida a saída em λ (λ). Outros tipos de entradas devem ser recusados/rejeitados;
4. A multiplicação unária. Exemplo: $111 \times 111 = 111111111$;
5. Realize a transformação de palavras do tipo: $a^i b^j c^k \xrightarrow{*} a^{i+1} b^{j+1} c^{k+1}$;

Sistemas de Post: Construir SP's que gerem os seguintes conjuntos de palavras:

1. Os palíndromos pares com $\Sigma = \{a, b, c\}$;
2. Palavras do tipo a^{3n} para $n \geq 0$ com $\Sigma = \{a\}$. Se mudarmos Σ para $\Sigma = \{a, b, c\}$, obtem-se generalizações bem interessantes. Pense sobre.

Referências

[Hein, 1999] Hein, J. L. (1999). *Theory of Computation: An Introduction*. Jones and Barlett Publishers, United States – USA.