Máquinas Universais

Prof. Marcus Vinícius Midena Ramos

Universidade Federal do Vale do São Francisco

9 de dezembro de 2022

marcus.ramos@univasf.edu.br
`www.univasf.edu.br/~marcus.ramos`

Bibliografia

1. *Teoria da Computação (capítulos 4, 5, 6 e 7)*
   T. A. Diverio e P. B. Menezes
   Bookman, 2011, 3ª edição
2. *Introduction to Automata Theory, Languages and Computation (capítulo 8)*
   J. E. Hopcroft, R. Motwani e J. D. Ullman
   Addison-Wesley, 2007, 3ª edição

Roteiro

1. Introdução
2. Hipótese de Church
3. Codificação de dados estruturados
4. Máquina Norma
5. Máquina de Turing
6. Máquina de Post
7. Máquina com Pilhas
8. Autômato com Duas Pilhas
9. Variações das Máquinas de Turing

Algoritmo

De uma forma geral:

- Definição informal;
- Descrição finita e não-ambígua;
- Passos discretos, executáveis mecanicamente;
- Tempo finito;
- Restrições de ordem prática: tempo e espaço;
- Restrições de ordem teórica: tanto quanto necessário.

Principalmente:

- Um algoritmo sempre pára com qualquer entrada e dá uma resposta;
- Um algoritmo nunca entra em "loop".

Algoritmo

- Realização na forma de programa;
- Programa demanda uma máquina para sua execução;
- Características desejáveis das máquinas:
  - *Simplicidade*: Apenas características essenciais, com omissão de características não-relevantes. Permitir conclusões generalizadas sobre a classe das funções computáveis.
  - *Poder*: Representação de qualquer função computável. Simulação de qualquer outra máquina real ou teórica.

Máquina universal

Conceito

- Aquela que permite a representação de qualquer algoritmo na forma de um programa para a mesma;
- Evidências que permitem caracterizar uma máquina como sendo universal:
  - *Interna*: Quaisquer extensões ou variações não aumentam o seu poder computacional (o conjunto de funções computáveis permanece inalterado).
  - *Externa*: Equivalência com outros modelos (máquinas ou não) que representam a noção de algoritmo.

Máquina universal
Modelos estudados

- Máquina Norma;
  - Interna;
- Máquina de Turing;
- Máquina Norma;
  - Externa;
- Máquina de Post;
  - Externa;
- Máquina com Pilhas;
- Autômato com Duas Pilhas.
  - Externa;

Hipótese de Church

- Alonzo Church, 1903-1995, matemático norte-americano;
- Também conhecida como Hipótese de Church-Turing, 1936;
- Mesmo ano em que foi apresentada a Máquina de Turing;
- Estabelece a equivalência entre a noção de algoritmo e Máquina de Turing;
- Como a noção de algoritmo é informal, a hipótese não pode ser provada;
- A necessidade por uma definição formal de algoritmo é grande, pois apenas a partir dela é que é possível investigar a existência de algoritmos que resolvem (ou não) certos problemas e calculam (ou não) certas funções, além de poder demonstrar certas propriedades dos mesmos.

Hipótese de Church

- “Qualquer função computável pode ser processada por alguma Máquina de Turing”;
- “A Máquina de Turing é o dispositivo de computação mais genérico que existe”;
- “Tudo que é computável é computável por uma Máquina de Turing”;
- “ A capacidade de computação representada pela Máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação”;
- “Qualquer outra forma de expressar algoritmos terá, no máximo, a mesma capacidade computacional da Máquina de Turing”.

Hipótese de Church

- Ao longo das décadas, evidências internas e externas apenas reforçam a Hipótese de Church, que é aceita como verdadeira de forma praticamente generalizada e não questionada;
- A Máquina de Turing (entre outros modelos), pela sua simplicidade, passa a ser usada como definição formal de algoritmo, atendendo aos propósitos citados anteriormente.

Algoritmos e tipos de dados

- Algoritmos manipulam, normalmente, diversos tipos de dados (inteiros positivos, negativos, racionais, reais, lógicos, cadeias de caracteres, vetores, estruturas etc);
- Com o objetivo de evitar que os modelos matemáticos abstratos se tornem (desnecessariamente) complexos, o escopo de manipulação de dados dos algoritmos que serão estudados é restrito aos números inteiros positivos;
- Essa restrição não traz maiores conseqüências, uma vez que esses e vários outros tipos de dados podem ser representados através de codificações apropriadas dos mesmos no espaço dos números inteiros não negativos.

Função de codificação

Seja X um conjunto de dados estruturados. A função *injetora*:

$$c : X \to \mathbb{N}$$

é tal que, $\forall x \in X, c(x)$ representa a codificação do dado estruturado x. Como c é injetora,

$$(c(x) = c(y)) \Rightarrow (x = y)$$

portanto a codificação representa de forma unívoca o dado estruturado x na forma de um número natural $c(x)$[1].

[1] Não é necessário que a função c seja *total* e *sobrejetora*. Mas, se este for o caso, então o conjunto X deve ser um conjunto enumerável (caso contrário não há bijeção entre X e $\mathbb{N}$).

Teorema fundamental da aritmética

Enunciado

Seja $a > 1$. Então:

$$a = p_1^{n_1} p_2^{n_2} ... p_k^{n_k}$$

onde:

- $p_1 < p_2 < ... < p_k$ são números primos (não necessariamente os primeiros, não necessariamente consecutivos);
- $n_1, n_2, ..., n_k$ são números inteiros positivos maiores ou iguais a 1;
- Essa decomposição é única, a menos de permutações.

$\Rightarrow$ Qualquer número inteiro maior que 1 pode ser decomposto, de forma unívoca, no produto de potências de números primos.
$\Rightarrow$ Números primos são a base para a definição dos demais números (números compostos).

Teorema fundamental da aritmética

Exemplos

- $2 = 2^1$;
- $17 = 17^1$;
- $256 = 2^8$;
- $143 = 11^1.13^1$;
- $42706587 = 3^1.7^6.11^2$;
- $132187055 = 5^1.7^5.11^2.13^1$.

Exemplo

n-uplas de números naturais

- Deseja-se obter $c : \mathbb{N}^n \rightarrow \mathbb{N}$
- Teorema fundamental da aritmética;
- Considere os n primeiros números primos, $p_1, p_2, ..., p_n$;
- Então $c(x_1, x_2, ..., x_n) = p_1^{x_1}.p_2^{x_2}.....p_n^{x_n}$
- Todo número natural decomponível nos n primeiros números primos corresponde a uma (única) n-upla;
- Representação unívoca de n-uplas como números naturais.

Exemplo:

- $c(1, 2, 3) = 2^1.3^2.5^3 = 2.9.125 = 2250$;
- 2250 representa, de forma unívoca, a tripla $(1, 2, 3)$.

Exemplo

Programas monolíticos

- Deseja-se obter $c : \mathbb{P} \to \mathbb{N}$, onde $\mathbb{P}$ é o conjunto dos programas monolíticos;
- Considere que o programa P possui as operações $O_1, O_2, ..., O_m$ e os testes $T_1, T_2, ..., T_n$;
- Considere rótulos numéricos sequenciais, com rótulo inicial 1 e rótulo final (único) 0;
- Quádruplas representam as instruções: o primeiro elemento indica o tipo de instrução (0 para desvio incondicional e 1 para desvio condicional);
- Considere que $(0, k, r_2, r_2)$ representa a instrução
  r_1 : faça O_k vá_para r_2
- Considere que $(1, k, r_2, r_3)$ representa a instrução
  r_1 : se T_k então vá_para r_2 senão vá_para r_3

Exemplo

Programas monolíticos

- Cada instrução de P é codificada na forma de uma quádrupla;
- Cada quádrupla é codificada na forma de um número inteiro;
- Se P contém t instruções, serão geradas t quádruplas e, consequentemente, t números inteiros;
- Considere a t-upla formada por esses t números inteiros;
- Codifique a t-upla como um número inteiro.

Exemplo

Programas monolíticos

Considere o programa monolítico P:
1: se T_1 vá_para 2 senão vá_para 0
2: faça O_1 vá_para 1

- $(1, 1, 2, 0)$ representa a instrução associada ao rótulo 1;
- $(0, 1, 1, 1)$ representa a instrução associada ao rótulo 2;
- $c(1, 1, 2, 0) = 2^1.3^1.5^2.7^0 = 150$;
- $c(0, 1, 1, 1) = 2^0.3^1.5^1.7^1 = 105$;
- Considere $(150, 105)$ como a representação de P;
- $c(150, 105) = 2^{150}.3^{105}$
- O número $2^{150}.3^{105}$ representa P.

Exemplo

Programas monolíticos

Genericamente, se w representa um programa monolítico P com t instruções, então:

- $w = 2^{i_1}.3^{i_2}.5^{i_3}.....p_t^{i_t}$
- $\forall j, 1 \leq j \leq t$:
  - $i_j = 2^a.3^b.5^c.7^d$
  - Se $a = 0$, i_j representa a instrução:
    r_j: faça O_b vá_para r_c
  - Se $a = 1$, i_j representa a instrução:
    r_j: se T_b então vá_para r_c senão vá_para r_d

Generalidades

- Definida por Richard Bird em 1976;
- Number Theoretic Register MAchine (e, também, o nome da esposa dele...);
- É uma máquina de registradores (possui uma quantidade ilimitada deles);
- Arquitetura semelhante à dos computadores modernos;
- Cada registrador armazena um único número natural (sem limitação de tamanho);
- Operações e testes (para cada registrador):
  - Adicionar o valor 1;
  - Subtrair o valor 1 (se 0, continua com 0);
  - Testar se o conteúdo é 0.
- Máquina Universal.

Definição

$$\text{Norma} = (\mathbb{N}^\infty, \mathbb{N}, \mathbb{N}, ent, sai, \{add_k, sub_k \mid k \geq 0\}, \{zero_k \mid k \geq 0\})$$

- Os registradores são denotados $A, B, ..., X, Y$;
- $A(k = 0), B(k = 1), ...$;
- $ent : \mathbb{N} \rightarrow \mathbb{N}^\infty$, transfere o valor da entrada para X e zera os demais registradores;
- $sai : \mathbb{N}^\infty \rightarrow \mathbb{N}$, transfere o valor de Y para a saída;

Definição

$$\text{Norma} = (\mathbb{N}^{\infty}, \mathbb{N}, \mathbb{N}, ent, sai, \{add_k, sub_k \mid k \geq 0\}, \{zero_k \mid k \geq 0\})$$

- $add_k : \mathbb{N}^{\infty} \to \mathbb{N}^{\infty}$, adiciona 1 ao k-ésimo registrador, mantendo os demais inalterados;
- $sub_k : \mathbb{N}^{\infty} \to \mathbb{N}^{\infty}$, subtrai 1 do k-ésimo registrador, mantendo os demais inalterados; se 0, mantém 0;
- $zero_k : \mathbb{N}^{\infty} \to \{verdadeiro, falso\}$, retorna $verdadeiro$ se o conteúdo do k-ésimo registrador é 0, $falso$ caso contrário;
- Notação: $K := K + 1, K := K - 1, K = 0$

Evidências internas

- Operações e testes;
- Tipos de dados;
- Agregados;
- Endereçamento indireto;
- Recursão.

Operações e testes

Definições incrementais, através da expansão sucessiva do repertório de operações e testes da Máquina Norma.

- Atribuição do valor 0 a um registrador;
- Atribuição de um valor qualquer a um registrador;
- Adição de dois registradores;
- Atribuição de registrador à registrador;
- Multiplicação de dois registradores;
- Operador relacional menor;
- Teste de divisibilidade;
- Teste se o valor de um registrador é primo;
- Atribuição do n-ésimo número primo a um registrador.

Operações e testes

Atribuição do valor 0 a um registrador

Denotado:

$$A := 0$$

para o registrador A.

- Decrementar A até chegar em zero;
- Operação implementada através do programa iterativo:
  até $A = 0$
  faça $A := A - 1$
- Considerada como *macro*, $A := 0$ representa uma nova operação.

Operações e testes

Atribuição de um valor qualquer a um registrador

Denotado:

$$A := n$$

para o registrador A.

- Zerar A e depois incrementar até chegar em n;
- Operação implementada através do programa iterativo, com n repetições da operação $A := A + 1$:
  $A := 0$
  $A := A + 1$
  $A := A + 1$
  ...
  $A := A + 1$
- Considerada como *macro*, $A := n$ representa uma nova operação.

Operações e testes
Adição de dois registradores

Denotado:

$$A := A + B$$

para os registradores A e B.

- Decrementar B e incrementar A até zerar B;
- Operação implementada através do programa iterativo:
  até $B = 0$
  faça $(A := A + 1; B := B - 1)$
- O registrador B é zerado;
- Para preservar o valor de B, deve-se usar um registrador auxiliar;
- Considerada como *macro*, $A := A + B$ representa uma nova operação.

Operações e testes

Adição de dois registradores

Denotado:

$$A := A + B \text{ usando } C$$

para os registradores A e B, empregando C como auxiliar.

- Operação implementada através do programa iterativo:
  $C := 0$
  até $B = 0$
  faça $(A := A + 1; C := C + 1; B := B - 1)$;
  até $C = 0$
  faça $(B := B + 1; C := C - 1)$
- O registrador C é zerado;
- O identificação explícita do registrador C serve para evitar conflitos no uso do mesmo;
- Considerada como *macro*, "$A := A + B$ usando C" representa uma nova operação.

Operações e testes

Atribuição de registrador à registrador

Denotado:

$$A := B \text{ ou } A := B \text{ usando } C$$

para os registradores A e B, empregando C como auxiliar.

- "$A := B$ usando C" denota:
  $A := 0$
  $A := A + B$ usando C
  ou seja, B permanece inalterado após a atribuição.
- "$A := B$" denota:
  $A := 0$
  $A := A + B$
  ou seja, B é zerado após a atribuição.
- Consideradas como *macros*, "$A := B$" e "$A := B$ usando C" representam novas operações;

Operações e testes

Multiplicação de dois registradores

Denotado:

$$A := A * B \text{ usando } C, D$$

para os registradores A e B, empregando C e D como auxiliares.

- Somar B com ele mesmo $A - 1$ vezes;
- Operação implementada através do programa iterativo:
  $C := A$
  até $(C = 0)$
  faça $(A := A + B$ usando $D; C := C - 1)$
- Considerada como *macro*, "$A := A * B$ usando C, D" representa uma nova operação.

Operações e testes

Operador relacional menor

Denotado:

$$A < B$$

- Decrementar simultâneamente A e B até que um dos dois (ou os dois) se torne(m) zero;
- Se $B = 0$ então FALSO senão VERDADEIRO.

ou ainda:

- Até que $B = 0$ faça
  Se $A = 0$ então VERDADEIRO senão $(A := A - 1; B := B - 1)$
  FALSO

Operações e testes

Operador relacional menor

Denotado:

$$A < B \text{ usando } C, D, E$$

- Operação implementada através do programa iterativo (pelo segundo algoritmo):
  $C := A$ usando E;
  $D := B$ usando E;
  até $D = 0$
  faça (se $C = 0$
  então VERDADEIRO
  senão $(C := C - 1; D := D - 1)$);
  FALSO

Operações e testes

Teste de divisibilidade

Denotado:

$$\text{teste\_mod}(A, B) \text{ usando } C, D, E, C', D', E'$$

- Determina se A é divisível por B;
- Ou seja, se o resto da divisão inteira de A por B é zero;
- Denominador não pode ser 0;
- Numerador 0 é divisível por qualquer número diferente de 0;
- Subtrai B sucessivamente de A até o resto ser menor do que B;
- Se o resto for 0, então VERDADEIRO;
- Senão, FALSO.

Operações e testes

Teste de divisibilidade

Denotado:

$$\text{teste\_mod}(A, B) \text{ usando } C, D, E, C', D', E'$$

- Operação implementada através do programa iterativo:
  $C := A$ usando E;
  $D := B$ usando E;
  se $B = 0$
  então FALSO
  senão (até $C < D$ usando C', D', E' faça ($C := C - D$ usando E);
  se $C = 0$ então VERDADEIRO senão FALSO)

Operações e testes

Exercício

Obtenha um programa iterativo que implemente a operação:

$$A := A - B \text{ usando } C$$

Operações e testes

Teste se o valor de um registrador é primo

Denotado:

$$\text{teste\_primo}(A) \text{ usando } C, D, E, F, G, H, I$$

para o registrador A, empregando $C, ..., I$ como auxiliares.

- 0 não é primo;
- 1 não é primo;
- Testa a divisibilidade de A por todos os números entre $A - 1$ e 1, nesta ordem, parando quando acontecer o primeiro caso;
- Se este caso corresponder ao 1, então VERDADEIRO;
- Senão, FALSO.

Operações e testes

Teste se o valor de um registrador é primo

Denotado:

$$\text{teste\_primo}(A) \text{ usando } C, D, E, F, G, H, I$$

para o registrador A, empregando $C, ..., I$ como auxiliares.

- Operação implementada através do programa iterativo:
  se $A = 0$ então FALSO senão
    $C := A$ usando D;
    $C := C - 1$;
    se $C = 0$ então FALSO senão
      até teste_mod (A, C)usando D, E, F, G, H, I
      faça $C := C - 1$;
      $C := C - 1$;
      se $C = 0$ então VERDADEIRO senão FALSO

Operações e testes

Atribuição do n-ésimo número primo a um registrador

Denotado:

$$A := \text{ primo } (B) \text{ usando } C, D, E, F, G, H, I$$

para o registrador A, supondo que B contém $n \geq 1$ e empregando $C, ..., I$ como auxiliares.

- $A := 1$;
- Incrementar A até chegar em um número primo;
- Neste ponto, decrementar B;
- Repetir os dois passos anteriores até que $B = 0$.

Operações e testes

Atribuição do n-ésimo número primo a um registrador

Denotado:

$$A := \text{ primo } (B) \text{ usando } C, D, E, F, G, H, I$$

para o registrador A, supondo que B contém $n \geq 1$ e empregando $C, ..., I$ como auxiliares.

- Operação implementada através do programa iterativo:
  $A := 1;$
  até $B = 0$ faça
  $\quad B := B - 1;$
  $\quad A := A + 1;$
  $\quad$até teste_primo (A) usando C, D, E, F, G, H, I faça $A := A + 1$

Tipos de dados

Números inteiros positivos e negativos

Números inteiros com sinal m podem ser representados pela dupla:

$$(s, |m|)$$

onde

- $|m|$ representa o valor absoluto de m;
- se $m < 0$ então $s = 1$ senão $s = 0$.

A representação em Norma pode ser feita:

- Codificação de duplas, ou
- Par de registradores.

Tipos de dados

Números inteiros positivos e negativos

Denotado:

$$A := A + 1$$

supondo que A representa o par de registradores A_1 (s) e A_2 (m).

- Operação implementada através do programa iterativo:
  se $A_1 = 0$ então $A_2 := A_2 + 1$ senão
  $(A_2 := A_2 - 1;$
  se $A_2 = 0$ então $A_1 := 0$ senão $\checkmark)$
- $(0,0) + 1 = (0,1); (0,1) + 1 = (0,2); (1,1) + 1 = (0,0); (1,2) + 1 = (1,1)$
- Outras operações podem ser implementadas sem dificuldade.

Tipos de dados

Números racionais

Números racionais $r = \frac{a}{b}$ podem ser representados pela dupla:

$$(a, b)$$

com $b > 0$. Algumas operações e testes sobre os números racionais:

- Soma: $(a, b) + (c, d) = (a * d + b * c, b * d)$
- Subtração: $(a, b) - (c, d) = (a * d - b * c, b * d)$
- Multiplicação: $(a, b) * (c, d) = (a * c, b * d)$
- Divisão: $(a, b) \div (c, d) = (a * d, b * c)$, para $c \neq 0$
- Igualdade: $(a, b) = (c, d)$ se e somente se $a * d = b * c$

Agregados
Vetores

- Vetores com n elementos (inclusive com n variável) podem ser representados em um único registrador, usando codificação de n-uplas;
- Suponha que o registrador A representa o vetor com os elementos $A[1], A[2], ...,$
- Indexação direta (com número natural) ou indireta (com registrador).

Algumas operações e testes sobre vetores:

- Adiciona 1 à uma posição indexada;
- Subtrai 1 de uma posição indexada;
- Testa se uma posição indexada contém o valor 0.

Agregados

Vetores

Observações sobre a codificação proposta:

- A representação produz o mesmo valor numérico para vetores idênticos porém de tamanhos diferentes preenchidos com 0 nos elementos finais: $[9, 2, 6, 0, 0]$ e $[9, 2, 6, 0]$ são ambos codificados como $2^9 * 3^2 * 5^6$
- Logo, os zeros dos elementos finais, se existirem, são desconsiderados e o vetor é codificado apenas até o último elemento diferente de zero;
- Assim, a codificação proposta não permite recuperar o tamanho do vetor codificado se este possui zeros no final;
- Algumas soluções, no entanto, podem ser consideradas.

Agregados

Vetores

- Uma alternativa é considerar todos os vetores como sendo compostos por uma quantidade ilimitada de elementos (com zeros no final): $[9, 2, 6, 0, 0, 0, 0, 0, 0, ...]$
- Outra alternativa é considerar um elemento extra no final, representando a quantidade de elementos no vetor originalmente codificado:
  $[9, 2, 6, 0, 0, 5]$, que neste caso seria codificado pelo número $2^9 * 3^2 * 5^6 * 13^5$
- Elementos intermediários que contenham zero podem ser recuperados pela simples inspeção dos números primos produzidos pela decomposição, e considerando toda a seqüência de primos desde o início:
  $[4, 0, 1]$ é codificado como $2^4 * 5^1$. Na decodificação fica claro que o expoente do número primo intermediário (3) deve ser zero.

Agregados

Vetores

Suposições:

- p_n representa o n-ésimo número primo;
- A macro teste_mod (A, C), previamente definida, que retorna VERDADEIRO se C é divisor de A e FALSO caso contrário;
- A macro $A := A/C$, que retorna o resultado da divisão inteira de A por C, é dada;
- Será omitido o termo "usando" das macros já definidas.

Agregados

Vetores

Definição da macro:

$$add_{A[n]} \text{ usando } C$$

Adição de uma unidade ao elemento n do vetor A, usando indexação <u>direta</u>.

- $C := p_n$;
  $A := A * C$
- Considere o vetor $[4, 2, 3]$ e seja $A = c(4, 2, 3) = 2^4 * 3^2 * 5^3 = 18000$:
  - Para executar $add_{A[2]}$, basta fazer $A = A * 3$; o valor resultante (54000) representa o vetor $[4, 3, 3]$;
  - Para representar $[4, 2, 3, 5]$ (acréscimo de elemento), basta fazer $A = A * 7^5$.

Agregados

Vetores

Definição da macro:

$$sub_{A[n]} \text{ usando } C$$

Subtração de uma unidade do elemento n do vetor A, usando indexação direta.

- $C := p_n$;
  se teste_mod (A, C)
  então $A := A/C$
  senão ✓
- Considere o vetor $[4, 2, 3]$ e seja $A = c(4, 2, 3) = 2^4 * 3^2 * 5^3 = 18000$:
  - Para executar $sub_{A[2]}$, basta fazer $A = A/3$; o valor resultante (6000) representa o vetor $[4, 1, 3]$;
  - Para representar $[4, 2]$ (eliminação de elemento), basta fazer $A = A/5^3$.

Agregados
Vetores

Definição da macro:

$$zero_{A[n]} \text{ usando } C$$

Testa se o elemento n do vetor A contém o valor 0, usando indexação direta.

- $C := p_n$;
  se teste_mod (A, C)
  então FALSO
  senão VERDADEIRO
- Considere o vetor $[4, 2, 3]$ e seja $A = c(4, 2, 3) = 2^4 * 3^2 * 5^3 = 18000$:
  - Como teste_mod $(18000, 3) = 0$, segue que $A[2] \neq 0$;
  - Como teste_mod $(18000, 7) \neq 0$, segue que $A[4] = 0$.

Agregados
Vetores

Definição da macro:

$$add_{A[B]} \text{ usando } C$$

Adição de uma unidade ao elemento do vetor A, usando indexação indireta através do registrador B.

- $C :=$ primo (B);
  $A := A * C$

Agregados

Vetores

Definição da macro:

$$sub_{A[B]} \text{ usando } C$$

Subtração de uma unidade do elemento do vetor A, usando indexação indireta através do registrador B.

- $C :=$ primo (B);
  se teste_mod (A, C)
  então $A := A/C$
  senão ✓

Agregados

Vetores

Definição da macro:

$$zero_{A[B]} \text{ usando } C$$

Testa se o elemento do vetor A contém o valor 0, usando indexação indireta através do registrador B.

- $C :=$ primo (B);
  se teste_mod (A, C)
    então FALSO
    senão VERDADEIRO

Agregados
Máquina Norma com apenas 2 registradores

- Os registradores $A, B, ...$ da Máquina Norma podem ser simulados numa máquina equivalente, com apenas dois registradores, usando a representação de vetores na forma de n-uplas;
- Suponha que a máquina tenha apenas os registradores X e Y;
- Todo o processamento de uma Máquina Norma pode ser simulado na nova máquina com apenas esses dois registradores;
- Convenciona-se que $X[1]$ representa o registrador A, $X[2]$ o registrador B e assim por diante;
- As seguintes operações são definidas:
  - $add_{X[k]}$ usando Y
  - $sub_{X[k]}$ usando Y
  - $zero_{X[k]}$ usando Y

Agregados

Pilhas

- Estruturas do tipo *last-in-first-out*;
- Podem ser simuladas em Máquinas Norma através de dois registradores;
- O primeiro representa o conteúdo da pilha, considerado como um vetor e conforme visto anteriormente;
- O segundo contém o número do elemento que corresponde ao topo da pilha;
- As operações abaixo podem ser definidas facilmente:
  - $empilha$
  - $desempilha$

Endereçamento indireto

Desviar para a instrução cujo rótulo corresponde ao conteúdo de um registrador.

- r: faça F vá_para A
- r: se T vá_para A senão vá_para B
- "A" e "B" são registradores;
- Desvia para o endereço contido em "A" ("B");
- A macro "End_A" para calcula o endereço correspondente;
- "r: faça F vá_para End_A"
- "r: se T vá_para End_A senão vá_para End_B"

Endereçamento indireto

Suponha que A contém valores $\leq k$. O valor de A permanece inalterado.
Macro "End_A":

i	:	se $zero_A$ então vá_para 0 senão vá_para $i+1$
$i+1$	:	faça sub_A vá_para $i+2$
$i+2$	:	se $zero_A$ então vá_para $i+3$ senão vá_para $i+4$
$i+3$	:	faça $A := 1$ vá_para 1
$i+4$	:	faça sub_A vá_para $i+5$
$i+5$	:	se $zero_A$ então vá_para $i+6$ senão vá_para $i+7$
$i+6$	:	faça $A := 2$ vá_para 2
$i+7$	:	faça sub_A vá_para $i+8$
...		...
$i+k*3-1$	:	se $zero_A$ então vá_para $i+k*3$ senão vá_para $i+k*3+1$
$i+k*3$	:	faça $A := k$ vá_para k
$i+k*3+1$	:	faça sub_A vá_para $i+k*3+2$

Recursão

- Chamada de subprogramas e recursão podem ser simuladas em programas monolíticos com o uso do endereçamento indireto;
- Demonstração em Bird76.

Generalidades

- Definida por Alan Turing em 1936;
- Formulada antes da construção do primeiro computador digital;
- Aceita como formalização da noção informal de algoritmo;
- Possui, no mínimo, o mesmo poder computacional de qualquer computador moderno ou outro modelo de computação;
- Incorpora o programa na sua definição.

Conceito

Procura reproduzir uma pessoa trabalhando na solução de um problema:

- Instrumento para escrever, outro para apagar;
- Folha de papel dividida em regiões;
- Dados iniciais na folha de papel.

Durante o trabalho:

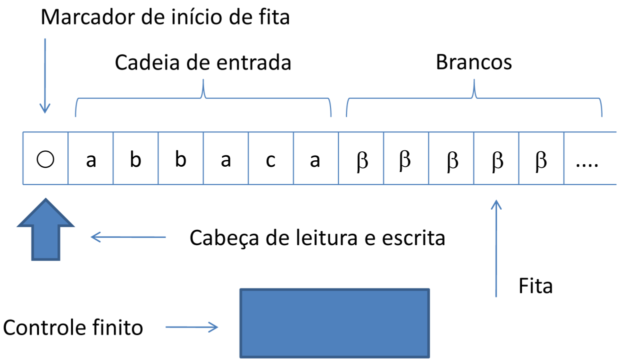
- Novo símbolo pode ser lido;
- Símbolo existente pode ser alterado;
- Olhos podem ser deslocar de região;
- Ação a ser executada depende do símbolo lido e do “estado mental” do trabalhador;
- Estados inicial e finais indicam começo e término das atividades.

Conceito

Algumas simplificações:

- A folha de papel tem dimensões tão grandes quanto necessárias;
- Ela é organizada de forma unidimensional e dividida em células;
- O conjunto de símbolos é finito;
- O conjunto de estados mentais é finito;
- Apenas um símbolo é lido de cada vez;
- A atenção se desloca apenas para as células adjacentes.

Componentes



Formalização

Uma Máquina de Turing é uma 8-upla:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

onde:

- Σ é o alfabeto de entrada;
- Q é o conjunto de estados;
- Π é a função (parcial) de transição:

$$\Pi : Q \times (\Sigma \cup V \cup \{\beta, \circ\}) \rightarrow Q \times (\Sigma \cup V \cup \{\beta, \circ\}) \times \{E, D\}$$

Formalização

Uma Máquina de Turing é uma 8-upla:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

onde:

- $q_0 \in Q$ é o estado inicial;
- $F \subseteq Q$ é o conjunto de estados finais;
- V é o alfabeto auxiliar, $V \cap \Sigma = \emptyset$;
- $\beta \notin (\Sigma \cup V)$ é o símbolo especial “branco”;
- $\circ \notin (\Sigma \cup V)$ é o marcador de início de fita.

Configuração

A configuração de uma Máquina de Turing deve representar:

- O estado corrente;
- O conteúdo corrente da fita;
- A posição do cursor sobre a fita.

Isso é feito considerando-se a configuração como um elemento (α, q, β) do conjunto:

$$(\Sigma \cup V \cup \{\beta, \circ\})^* \times Q \times (\Sigma \cup V \cup \{\beta, \circ\})^*$$

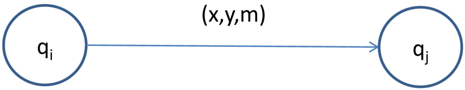
- $\alpha \in (\Sigma \cup V \cup \{\beta, \circ\})^*$ representa a parte da fita que está situada à esquerda da posição corrente do cursor;
- $q \in Q$ representa o estado corrente;
- $\beta \in (\Sigma \cup V \cup \{\beta, \circ\})^*$ representa a parte da fita que está situada à direita da posição corrente do cursor, incluindo a mesma.

Diagrama de estados

Se:

$$\Pi(q_i, x) = (q_j, y, m)$$

então:



Nesse caso,

- $(\alpha, q_i, x\beta) \vdash (\alpha y, q_j, \beta)$, se $m = D$, ou
- $(\alpha z, q_i, x\beta) \vdash (\alpha, q_j, zy\beta)$, se $m = E$,

representam possíveis movimentações a partir de uma mesma configuração pela aplicação da transição $\Pi(q_i, x) = (q_j, y, m)$.

Critérios de aceitação

Existem várias maneiras de formular a aceitação de uma cadeia w por uma Máquina de Turing M. Todas elas são equivalentes entre si:

1. "Estado final": w é aceita se, após a parada, M se encontra em um estado final; uma cadeia é rejeitada se, após a parada, M se encontra em um estado não-final;
2. "Entrada": w é aceita imediatamente após a entrada de M em um estado final, mesmo que existam outras possibilidades de movimentação nesse estado; uma cadeia é rejeitada se, após a parada, M se encontra em estado não-final;
3. "Parada": w é aceita se M pára; uma cadeia é rejeitada se M entra em loop infinito;

Em todos os casos, w é rejeitada se a cabeça de leitura/escrita se deslocar à esquerda da primeira célula da fita de entrada.

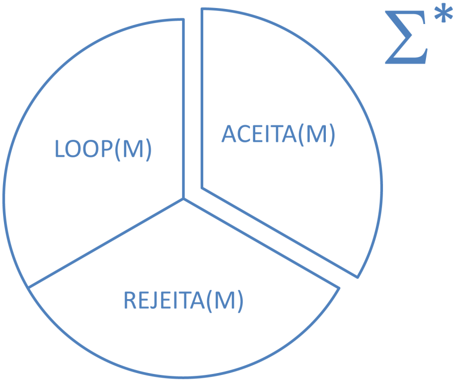
MT e linguagens

Considere-se o critério de aceitação por "Entrada" e a Máquina de Turing:

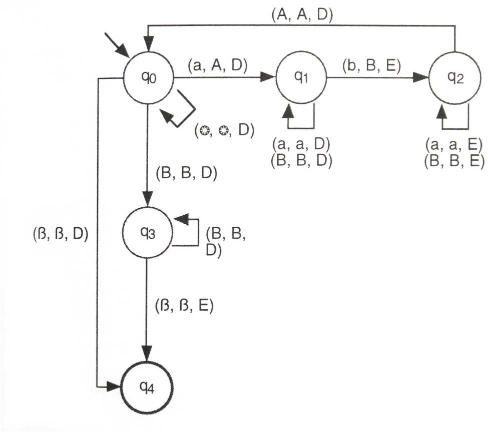
$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

- A linguagem aceita por M, denotada $ACEITA(M)$ ou $L(M)$ é: $\{w \in \Sigma^* \,|\, M$ assume algum estado $q_f \in F$ ao processar a entrada $w\}$
- A linguagem rejeitada por M, denotada $REJEITA(M)$ é: $\{w \in \Sigma^* \,|\, M$ pára em um estado $q \notin F$ ao processar a entrada w ou a cabeça de leitura/escrita se desloca para a esquerda da primeira posição$\}$
- A linguagem para a qual M entra em loop, denotada $LOOP(M)$ é: $\{w \in \Sigma^* \,|\, M$ processa a entrada indefinidamente$\}$

Particionamento



Exemplo



Exemplo

- $ACEITA(M) = \{a^n b^n | n \geq 0\}$
- $REJEITA(M) = \Sigma^* - ACEITA(M)$
- $LOOP(M) = \{\}$

Computação de M com a entrada $aabb$:

- $(\epsilon, q_0, \circ aabb)$, $(\circ, q_0, aabb)$,
  $(\circ A, q_1, abb)$, $(\circ Aa, q_1, bb)$,
  $(\circ A, q_2, aBb)$, $(\circ, q_2, AaBb)$,
  $(\circ A, q_0, aBb)$, $(\circ AA, q_1, Bb)$,
  $(\circ AAB, q_1, b)$, $(\circ AA, q_2, BB)$,
  $(\circ A, q_2, BB)$, $(\circ AA, q_0, BB)$,
  $(\circ AAB, q_3, B)$, $(\circ AABB, q_3, \epsilon)$,
  $(\circ AAB, q_4, B)$

Linguagem geral

Definição

- $L \subseteq \Sigma^*$ é dita geral;
- Corresponde à maior classe de linguagens que pode ser definida sobre um alfabeto, sem garantias de que possa ser reconhecida (ou apenas aceita) mecanicamente.

Linguagem recursivamente enumerável

Definição

$L \subseteq \Sigma^*$ é dita recursivamente enumerável se existe uma Máquina de Turing M tal que:

- Se $w \in L$, M pára e aceita a entrada;
- Se $w \notin L$, M:
  - Pára e rejeita a entrada, ou
  - Entra em processamento indefinido e não pára ("loop infinito").

Corresponde à maior classe de linguagens que pode ser aceita mecanicamente, porém sem garantia de que o processamento pára quando a cadeia de entrada não pertence à linguagem definida.

Linguagem recursiva
Definição

$L \subseteq \Sigma^*$ é dita recursiva se existe uma Máquina de Turing M tal que:

- Se $w \in L$, M pára e aceita a entrada;
- Se $w \notin L$, M: pára e rejeita a entrada.

Corresponde à maior classe de linguagens que pode ser reconhecida mecanicamente, <u>com</u> garantia de que o processamento pára para toda e qualquer cadeia de entrada.

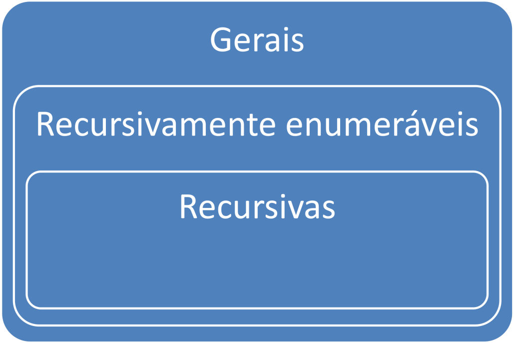
Linguagem recursivamente enumerável $\times$ linguagem recursiva

- Toda linguagem recursiva é também recursivamente enumerável;
- Existem linguagens que são recursivamente enumeráveis porém não são recursivas;
- $C_{LR} \subset C_{LRE}$;
- Se L é uma dessas linguagens, então toda e qualquer Máquina de Turing M que aceita L é tal que:
  - $ACEITA(M) = L$;
  - $LOOP(M) \neq \{\}$.
- Ou seja, existe pelo menos uma cadeia de entrada (não pertencente à linguagem) que faz M entrar em loop infinito, qualquer que seja M;
- L é dita recursivamente enumerável e não-recursiva.

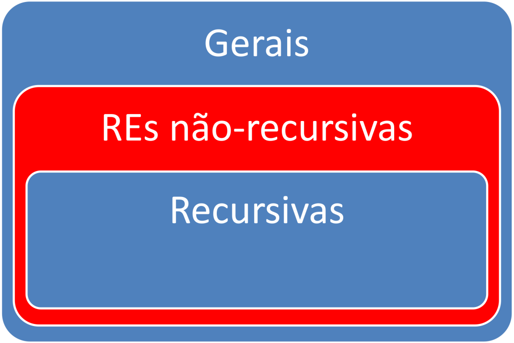
Linguagem geral × recursivamente enumerável

- Toda linguagem recursivamente enumerável é também geral;
- Existem linguagens que são gerais porém não são recursivamente enumeráveis;
- $C_{LRE} \subset C_G$;
- Se L é uma dessas linguagens, então toda e qualquer Máquina de Turing M que "aceita" L é tal que:
  - $ACEITA(M) \neq L$;
  - $LOOP(M) \neq \{\}$.
- Ou seja, existe pelo menos uma cadeia de entrada (pertencente à linguagem) que faz M entrar em loop infinito, qualquer que seja M;
- L é dita geral e não-recursivamente enumerável, ou simplesmente não-recursivamente enumerável.

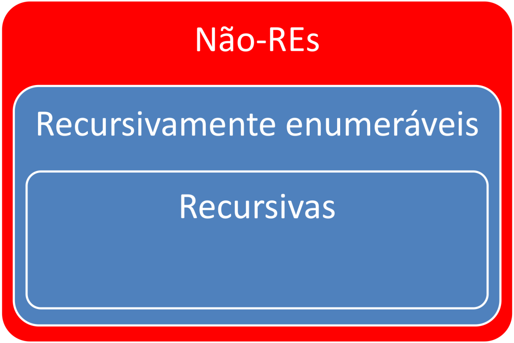
Hierarquia de linguagens



Linguagem recursivamente enumerável não-recursiva



Linguagem não-recursivamente enumerável



Propriedades

Serão demonstradas mais adiante:

- O complemento de uma linguagem recursiva é uma linguagem recursiva;
- Se uma linguagem e o seu complemento são recursivamente enumeráveis, então a linguagem é recursiva.

MT funções

- Máquinas de Turing pode ser vistas e estudadas como dispositivos que definem linguagens;
- Máquinas de Turing podem, também, ser vistas como dispositivos que computam funções:
  - O argumento é posicionado na fita de entrada;
  - Ao término da computação o conteúdo da fita representa o resultado da aplicação da função ao argumento fornecido.
- Definição de linguagens $\Leftrightarrow$ Computação de funções.

Função computável

Uma função parcial:

$$f : (\Sigma^*)^n \to \Sigma^*$$

é dita *Função Turing-Computável*, ou simplesmente *Função Computável* se existe uma Máquina de Turing $M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$ que computa f, ou seja:

- Considere $(w_1, w_2, ..., w_n) \in (\Sigma^*)^n$, representada na fita de entrada como $\circ w_1 w_2 ... w_n$
- Se $f(w_1, w_2, ..., w_n) = w$, então o processamento de M com a entrada $\circ w_1 w_2 ... w_n$:
  - Pára (não importa se aceitando ou rejeitando);
  - O conteúdo da fita de entrada é $\circ w$.
- Se f não é definida para o argumento $(w_1, w_2, ..., w_n)$, então o processamento de M com a entrada $\circ w_1 w_2 ... w_n$:
  - Entra em loop infinito.

Função computável total

Uma função total:

$$f : (\Sigma^*)^n \rightarrow \Sigma^*$$

é dita *Função Turing-Computável Total*, ou simplesmente *Função Computável Total* se existe uma Máquina de Turing:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

que computa f e que sempre pára para qualquer entrada.

Função computável total

Exemplo 1

Considere a função total:

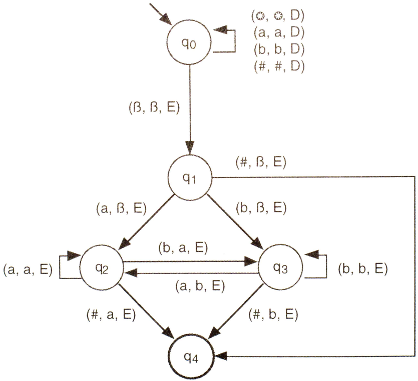
$$f : (\{a, b\}^*)^2 \rightarrow \{a, b\}^*$$

f devolve a concatenação de duas cadeias quaisquer fornecidas como entrada, ou seja $f(w_1, w_2) = w_1 w_2$. O símbolo $\#$ será usado para delimitar w_1 e w_2 na cadeia de entrada. Exemplos:

- $f(b, a) = ba$. A fita inicia com $\circ b\#a$ e termina com $\circ ba$
- $f(abb, abab) = abbabab$. A fita inicia com $\circ abb\#abab$ e termina com $\circ abbabab$

Função computável total

Exemplo 1



Função computável total

Exemplo 1

Algoritmo: a segunda cadeia é deslocada uma posição para a esquerda, símbolo por símbolo; o símbolo # desaparece.

- Desloca a cabeça para a direita até encontrar o primeiro branco;
- Desloca uma posição para a esquerda, memoriza o símbolo lido no estado e desloca novamente para a esquerda;
- q_2 representa que o último símbolo lido foi a e q_3 representa b;
- Conforme o símbolo corrente, grava um novo símbolo no lugar dele correspondente ao estado em que a máquina se encontra;
- Se houver necessidade, mudar de q_2 para q_3 e vice-versa para manter a coerência no significado atribuído aos estados;
- Fazer isso sucessivas vezes, até encontrar o símbolo #.

Função computável total

Exemplo 2

Considere a função total:

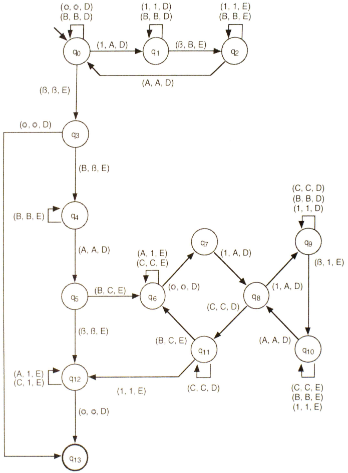
$$g : \{1\}^* \to \{1\}^*$$

f devolve o quadrado do número de entrada (ambos representados em unário), ou seja $g(n) = n^2$. Exemplos:

- $f(1) = 1$. A fita inicia com $\circ 1$ e termina com $\circ 1$
- $f(111) = 111111111$. A fita inicia com $\circ 111$ e termina com $\circ 111111111$

Função computável total

Exemplo 2



Teorema 1

Máquina de Turing $\leq$ Máquina Norma

- Toda Máquina de Turing pode ser simulada por alguma Máquina Norma.
- Se $M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$ é uma Máquina de Turing, então existe um programa monolítico P que simula M na Máquina Norma.

Teorema 1

Pré-requisitos

- O critério de aceitação deve ser por “Entrada”;
- A Máquina de Turing é determinística;
- A função de transição Π deve ser total:
  $Q' \leftarrow Q \cup \{q_e\}$
  $\Pi' \leftarrow \Pi$
  $\forall q \in Q', \tau \in (\Sigma \cup V)$, se Π não é definida para (q, τ)
  então $\Pi' \leftarrow \Pi' \cup \{(q, \tau) \rightarrow (q_e, \tau, E)\}$
- As cadeias serão rejeitadas por tentativa de movimentação da cabeça de leitura/escrita à esquerda da primeira posição da fita.

Teorema 1

Convenções

- Símbolos de $\Sigma \cup V$, $|\Sigma \cup V| = m$:
  τ_j, considerando $1 \leq j \leq m$, é representado pelo valor j;
  β é representado por 0;
  $\circ$ é representado por $m + 1$.
- A fita de entrada é representada como um vetor armazenado no registrador X;
- Observar que a escolha da representação de β por 0 faz com que existam infinitos símbolos β à direita do último símbolo da cadeia de entrada. Qualquer elemento do vetor que não contenha um elemento de $\Sigma \cup V$ retorna, na codificação de ênuplas, o valor 0 (de 2^0), que representa o símbolo β.

Teorema 1

Convenções

- A posição referenciada pela cabeça de leitura/escrita corresponde ao conteúdo do registrador C:
  valor inicial 1 aponta para o símbolo $\circ$;
- O estado corrente é representado pelo conteúdo do registrador Q: q_i, $i \geq 0$, é representado pelo valor i;
- Ao término do processamento, $Y = 0$ indica rejeição da cadeia de entrada; $Y \neq 0$ indica aceitação da cadeia de entrada, e o valor de Y representa o conteúdo da fita nessa situação;
- Observar que o conteúdo da fita é representado por um valor sempre maior ou igual a 1 (será 1 se ela contiver apenas brancos).

Teorema 1

Algoritmo

Instruções iniciais de P para a Máquina Norma:

r_0 : se $zero_C$ então vá_para r_{0_1} senão vá_para r_{0_2}
r_{0_1} : faça $Y := 0$ vá_para r_{0_5}
r_{0_2} : faça $A := 2^Q * 3^{X[C]}$ vá_para End_A
r_{0_3} : se $zero_C$ então vá_para r_{0_1} senão vá_para r_{0_4}
r_{0_4} : faça $Y := X$ vá_para r_{0_5}

- r_0 é o rótulo inicial;
- O controle retorna para r_0 sempre que o próximo estado é não-final;
- O controle retorna para r_{0_3} sempre que o próximo estado é final;
- r_{0_5} é o rótulo final.

Teorema 1

Algoritmo

Para cada transição $\Pi(q_i, \tau_m) = (q_j, \tau_n, D)$, acrescentar à P o seguinte conjunto de instruções:

$$\begin{array}{rl} r_{2^i*3^m}: & \text{faça } X[C] := n \text{ vá\_para } r_{2^i*3^m{}_1} \\ r_{2^i*3^m{}_1}: & \text{faça } add_C \text{ vá\_para } r_{2^i*3^m{}_2} \\ r_{2^i*3^m{}_2}: & \text{faça } Q := j \text{ vá\_para } r_0 \end{array}$$

- Grava τ_n na posição corrente da fita;
- Desloca a cabeça de leitura/escrita para a direita;
- Atualiza o estado corrente para q_j;
- Se o movimento for para a esquerda, usar sub_C no lugar de add_C;
- Se $q_j \in F$, então substituir r_0 por r_{0_3}.

Teorema 1
Algoritmo

Se $|Q| = m$ e $|\Sigma \cup V| = n$, então o programa monolítico correspondente possuirá:

$$5 + m * (n + 2) * 3$$

instruções rotuladas.
Detalhamento do cálculo:

- 5: quantidade de instruções rotuladas iniciais;
- $n + 2$: n símbolos, mais β e $\circ$;
- $m * (n + 2)$: função de transição total, quantidade total de transições;
- 3: quantidade de instruções rotuladas por transição.

Exemplo

Função de transição total

Considere a Máquina de Turing M que aceita a linguagem $\{a^n b^n \mid n \geq 0\}$. Então, para obter M' com função de transição total, devemos acrescentar o estado q_e e as seguintes transições:

$$
\begin{array}{lll}
(q_0, b) \rightarrow (q_e, b, E), & (q_0, A) \rightarrow (q_e, A, E), & \\
(q_1, A) \rightarrow (q_e, A, E), & (q_1, \circ) \rightarrow (q_e, \circ, E), & (q_1, \beta) \rightarrow (q_e, \beta, E), \\
(q_2, b) \rightarrow (q_e, b, E), & (q_2, \circ) \rightarrow (q_e, \circ, E), & (q_2, \beta) \rightarrow (q_e, \beta, E), \\
(q_3, a) \rightarrow (q_e, a, E), & (q_3, b) \rightarrow (q_e, b, E), & (q_3, A) \rightarrow (q_e, \circ, E), \\
(q_3, \circ) \rightarrow (q_e, \circ, E), & & \\
(q_4, a) \rightarrow (q_e, a, E), & (q_4, b) \rightarrow (q_e, b, E), & (q_4, A) \rightarrow (q_e, A, E), \\
(q_4, B) \rightarrow (q_e, B, E), & (q_4, \circ) \rightarrow (q_e, \circ, E), & (q_4, \beta) \rightarrow (q_e, \beta, E), \\
(q_e, a) \rightarrow (q_e, a, E), & (q_e, b) \rightarrow (q_e, b, E), & (q_e, A) \rightarrow (q_e, A, E), \\
(q_e, B) \rightarrow (q_e, B, E), & (q_e, \circ) \rightarrow (q_e, \circ, E), & (q_e, \beta) \rightarrow (q_e, \beta, E)
\end{array}
$$

Exemplo

Representação dos estados

Considere a Máquina de Turing M' que aceita a linguagem $\{a^n b^n \,|\, n \geq 0\}$. Então:

- $Q' = \{q_0, q_1, q_2, q_3, q_4, q_e\}$
  Representação no registador Q:
  - q_0 por 0
  - q_1 por 1
  - q_2 por 2
  - q_3 por 3
  - q_4 por 4
  - q_e por 5

Exemplo

Representação dos símbolos

Considere a Máquina de Turing M' que aceita a linguagem $\{a^n b^n \mid n \geq 0\}$. Então:

- $\Sigma \cup V = \{a, b\} \cup \{A, B\} = \{a, b, A, B\}$
  Representação no registrador X:
  - a por 1,
  - b por 2,
  - A por 3,
  - B por 4,

  Adicionalmente:
  - β por 0
  - $\circ$ como 5

Exemplo
Configuração inicial

Situação inicial dos registradores na Máquina Norma para a cadeia de entrada $\circ aabb$:

- $\circ aabb$ é representada pela seqüência 51122
- $X = 2^5 * 3^1 * 5^1 * 7^2 * 11^2 = 2.845.920$
- $Q = 0$
- $C = 1$

Exemplo

Programa para Máquina Norma

$$
\begin{array}{rl}
r_0: & \text{se } zero_C \text{ então vá\_para } r_{0_1} \text{ senão vá\_para } r_{0_2} \\
r_{0_1}: & \text{faça } Y := 0 \text{ vá\_para } r_{0_5} \\
r_{0_2}: & \text{faça } A := 2^Q * 3^{X[C]} \text{ vá\_para } End\_A \\
r_{0_3}: & \text{se } zero_C \text{ então vá\_para } r_{0_1} \text{ senão vá\_para } r_{0_4} \\
r_{0_4}: & \text{faça } Y := X \text{ vá\_para } r_{0_5}
\end{array}
$$

Exemplo

Programa para Máquina Norma

$$
\begin{array}{rl}
 & \Pi(q_0, \circ) = (q_0, \circ, D) \\
r_{243}: & \text{faça } X[C] := 5 \text{ vá\_para } r_{243_1} \\
r_{243_1}: & \text{faça } add_C \text{ vá\_para } r_{243_2} \\
r_{243_2}: & \text{faça } Q := 0 \text{ vá\_para } r_0 \\
 & \Pi(q_0, a) = (q_1, A, D) \\
r_3: & \text{faça } X[C] := 3 \text{ vá\_para } r_{3_1} \\
r_{3_1}: & \text{faça } add_C \text{ vá\_para } r_{3_2} \\
r_{3_2}: & \text{faça } Q := 1 \text{ vá\_para } r_0 \\
 & \ldots
\end{array}
$$

Exemplo

Programa para Máquina Norma

$$
\begin{array}{rl}
 & \ldots \\
 & \Pi(q_0, B) = (q_3, B, D) \\
r_{81}: & \text{faça } X[C] := 4 \text{ vá\_para } r_{81_1} \\
r_{81_1}: & \text{faça } add_C \text{ vá\_para } r_{82_2} \\
r_{81_2}: & \text{faça } Q := 3 \text{ vá\_para } r_0 \\
 & \ldots \\
 & \Pi(q_3, \beta) = (q_4, \beta, E) \\
r_8: & \text{faça } X[C] := 0 \text{ vá\_para } r_{8_1} \\
r_{8_1}: & \text{faça } sub_C \text{ vá\_para } r_{8_2} \\
r_{8_2}: & \text{faça } Q := 4 \text{ vá\_para } r_{0_3} \\
 & \ldots
\end{array}
$$

Exemplo

Endereços das transições

Q	τ	X[C]	r
0	○	5	243
0	a	1	3
0	b	2	9
0	A	3	27
0	B	4	81
0	β	0	1

Q	τ	X[C]	r
1	○	5	486
1	a	1	6
1	b	2	18
1	A	3	54
1	B	4	162
1	β	0	2

Q	τ	X[C]	r
2	○	5	972
2	a	1	12
2	b	2	36
2	A	3	108
2	B	4	324
2	β	0	4

Q	τ	X[C]	r
3	○	5	1944
3	a	1	24
3	b	2	72
3	A	3	216
3	B	4	648
3	β	0	8

Q	τ	X[C]	r
4	○	5	3888
4	a	1	48
4	b	2	144
4	A	3	432
4	B	4	1296
4	β	0	16

Q	τ	X[C]	r
5	○	5	7776
5	a	1	96
5	b	2	288
5	A	3	864
5	B	4	2592
5	β	0	32

Exemplo
Configurações

Cadeia $ab \in L(M')$:

Turing	rótulo	Q	C	X	A
$\Pi(q_0, \circ) = (q_0, \circ, D)$	r_0	0	1	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
	r_{243}	0	1	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
	r_{243_1}	0	1	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
	r_{243_2}	0	2	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
$\Pi(q_0, a) = (q_1, A, D)$	r_0	0	2	$2^5 * 3^1 * 5^2$	$2^0 * 3^1 = 3$
	r_3	0	2	$2^5 * 3^1 * 5^2$	$2^0 * 3^1 = 3$
	r_{3_1}	0	2	$2^5 * 3^3 * 5^2$	$2^0 * 3^1 = 3$
	r_{3_2}	0	3	$2^5 * 3^3 * 5^2$	$2^0 * 3^1 = 3$

Exemplo

Configurações

Turing	rótulo	Q	C	X	A
$\Pi(q_1, b) = (q_2, B, E)$	r_0	1	3	$2^5 * 3^3 * 5^2$	$2^1 * 3^2 = 18$
	r_{18}	1	3	$2^5 * 3^3 * 5^2$	$2^1 * 3^2 = 18$
	r_{18_1}	1	3	$2^5 * 3^3 * 5^4$	$2^1 * 3^2 = 18$
	r_{18_2}	1	2	$2^5 * 3^3 * 5^4$	$2^1 * 3^2 = 18$
$\Pi(q_2, A) = (q_0, A, D)$	r_0	2	2	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$
	r_{108}	2	2	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$
	r_{108_1}	2	2	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$
	r_{108_2}	2	3	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$

Exemplo

Configurações

Turing	rótulo	Q	C	X	A
$\Pi(q_0, B) = (q_3, B, D)$	r_0	0	3	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
	r_{81}	0	3	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
	r_{81_1}	0	3	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
	r_{81_2}	0	4	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
$\Pi(q_3, \beta) = (q_4, \beta, E)$	r_0	3	4	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	r_8	3	4	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	r_{8_1}	3	4	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	r_{8_2}	3	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$

Exemplo
Configurações

Turing	rótulo	Q	C	X	A
	r_{0_3}	4	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	r_{0_4}	4	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	r_{0_5}	4	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$

Exemplo
Configurações

Cadeia $ba \notin L(M')$:

Turing	rótulo	Q	C	X	A
$\Pi(q_0, \circ) = (q_0, \circ, D)$	r_0	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
	r_{243}	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
	r_{243_1}	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
	r_{243_2}	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
$\Pi(q_0, b) = (q_e, b, E)$	r_0	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$
	r_9	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$
	r_{9_1}	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$
	r_{9_2}	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$

Exemplo
Configurações

Turing	rótulo	Q	C	X	A
$\Pi(q_e, \circ) = (q_e, \circ, E)$	r_0	5	1	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	r_{7776}	5	1	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	r_{7776_1}	5	1	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	r_{7776_2}	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	r_0	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	r_{0_1}	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	r_{0_5}	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$

Teorema 2

Máquina Norma $\leq$ Máquina de Turing

- Conforme demonstrado em Bird76, programas recursivos (com definição e chamada de subprogramas e recursão) podem ser simulados em Máquinas Norma através de programas monolíticos, com o uso de endereçamento indireto.
- Portanto, é suficiente considerar programas monolíticos e as Máquinas de Turing que computam as mesmas funções;
- Também é suficiente considerar Máquina Norma com apenas dois registradores (X e Y). Ela será denotada $Norma_2$.

Teorema 2

Máquina Norma $\leq$ Máquina de Turing

- Todo Máquina Norma pode ser simulada por alguma Máquina de Turing.
- Se $P = (I, r_0)$ para $Norma_2$, então existe $M : (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$ que simula P.
- Máquina de Turing simula $Norma_2$, que por sua vez simula Norma.

Teorema 2

Convenções

- Registrador X: Seu conteúdo é representado em unário e armazenado nas células pares da fita de M;
  A fita de entrada $|\circ|\underbrace{1}|1|\underbrace{1}|1|\underbrace{1}|1|\underbrace{1}|\beta|\beta|\beta|...|$ representa $X = 4$
- Registrador Y: Seu conteúdo é representado em unário e armazenado nas células ímpares (exceto a primeira) da fita de M; A fita de entrada $|\circ|1|\underbrace{1}|1|\underbrace{1}|1|\underbrace{1}|1|\beta|\beta|\beta|...|$ representa $Y = 3$
- Rótulos das instruções: Cada rótulo r de P está em correspondência com um estado q_r de M. r_0 corresponde ao estado inicial q_0 e cada rótulo final r_f corresponde a um estado final $q_f \in F$.

Teorema 2

Componentes da Máquina de Turing

- $\Sigma = \{1\}$
- $Q = \{q_i \,|\, r_i$ é rótulo de $P\}$
- $\Pi = \{\}$
- Estado inicial $= q_0$ (supondo que r_0 é o rótulo inicial de P)
- $F = \{q_i \,|\, r_i$ é rótulo final de $P\}$
- $V = \{\}$

Teorema 2

Algoritmo

Para cada instrução $i \in I$:

- r: faça add_k vá_para r'
  1. A partir do estado q_r, deslocar a cabeça de leitura/escrita até encontrar a primeira célula par (se $K = X$) ou ímpar (se $K = Y$) que contenha um símbolo β;
  2. Substituir esse β por 1;
  3. Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ($\circ$);
  4. Ir para o estado $q_{r'}$.

Teorema 2

Algoritmo

Para cada instrução $i \in I$:

- r: faça sub_k vá_para r'
  1. A partir do estado q_r, deslocar a cabeça de leitura/escrita até encontrar a última célula par (se $K = X$) ou ímpar (se $K = Y$) que contenha um símbolo 1 (se a primeira célula pesquisada for β, ir para 3);
  2. Substituir esse 1 por β;
  3. Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ($\circ$);
  4. Ir para o estado $q_{r'}$.

Teorema 2

Algoritmo

Para cada instrução $i \in I$:

- r: se $zero_K$ então vá_para r' senão vá_para r''
  - A partir do estado q_r, deslocar a cabeça de leitura/escrita até encontrar a primeira célula par (se $K = X$) ou a segunda célula ímpar (se $K = Y$);
  - Se a célula encontrada contiver o símbolo β:
    1. Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ($\circ$);
    2. Ir para o estado $q_{r'}$
  - Se a célula encontrada não contiver o símbolo β:
    1. Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ($\circ$);
    2. Ir para o estado $q_{r''}$

Teorema 2

Exemplo

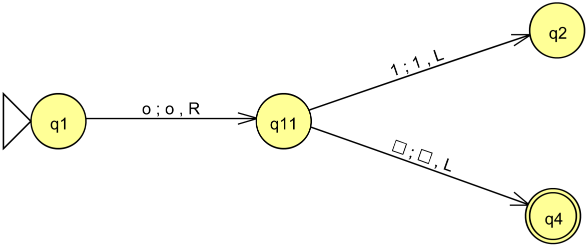
r_1: se $zero_X$ vá_para r_4 senão vá_para r_2
r_2: faça sub_X vá_para r_3
r_3: faça add_Y vá_para r_1

- $\Sigma = \{1\}$
- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Pi = \{\}$
- Estado inicial q_1 (pois r_1 é o rótulo inicial de P)
- $F = \{q_4\}$ (pois r_4 é rótulo final de P)
- $V = \{\}$

Teorema 2

Exemplo

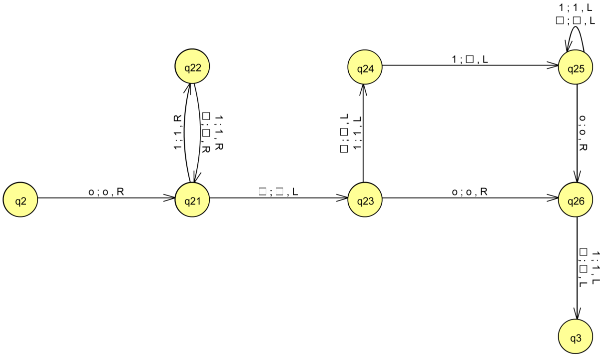
r_1 : se $zero_X$ vá_para r_4 senão vá_para r_2



Teorema 2

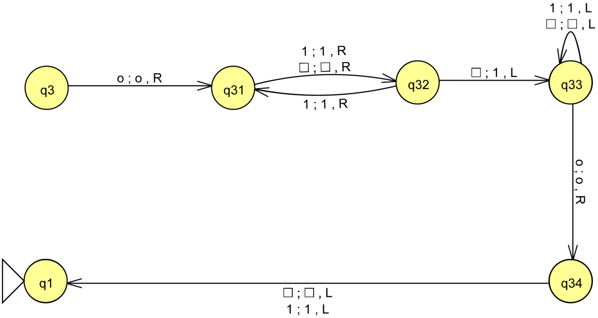
Exemplo

r_2 : faça sub_X vá_para r_3



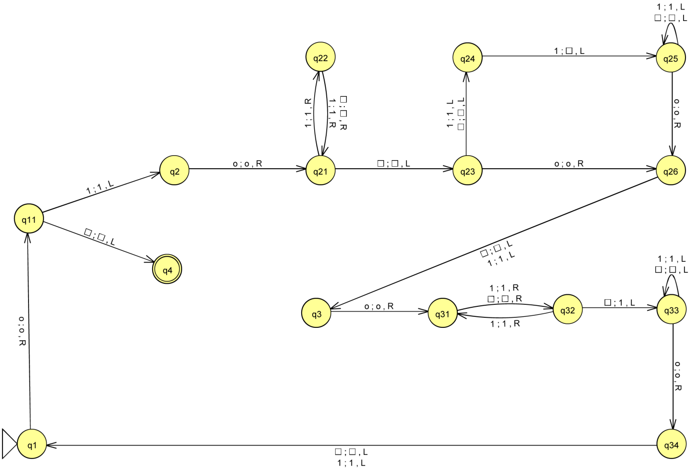
Teorema 2
Exemplo

r_3 : faça add_Y vá_para r_1



Teorema 2

Exemplo



Generalidades

- Definida por Emil Leon Post em 1936;
- Possui uma única variável, denominada X:
  - Fila (*first-in-first-out*);
  - Entrada, saída e trabalho;
  - Tamanho inicial igual ao comprimento da cadeia de entrada;
  - Tamanho pode variar, sem restrições.
- Possui um programa associado (fluxograma):
  - Partida;
  - Parada;
  - Desvio condicional;
  - Atribuição.
- Máquina Universal.

Definição

Uma Máquina de Post é uma tripla:

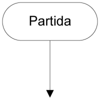
$$M = (\Sigma, D, \#)$$

onde:

- $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ é o alfabeto de entrada;
- D é o programa (ou fluxograma), constituído pelos componentes partida, parada, desvio condicional e atribuição;
- $\#$ é o símbolo auxiliar, $\# \notin \Sigma$.

Definição

Componente "Partida":



- Único em cada programa P;
- Indica o início da execução de P.

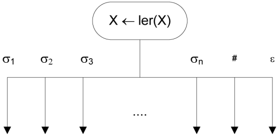
Definição

Componente “Parada”:

- Pode ser de dois tipos: parada com aceitação ou parada com rejeição;
- Múltiplas ocorrências são permitidas, sem restrições (inclusive zero ocorrências de qualquer ou de ambos os componentes).

Definição

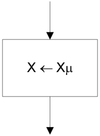
Componente "Desvio condicional":



- Analisa o primeiro símbolo da fila (variável X);
- Conforme o símbolo encontrado, desvia de acordo;
- O símbolo encontrado é removido do início da fila;
- Se $|\Sigma| = n$, devem ser previstos $n + 2$ desvios;
- Desvio para o símbolo auxiliar ($\#$) e também para o caso de X conter a cadeia vazia (ϵ).

Definição

Componente "Atribuição":



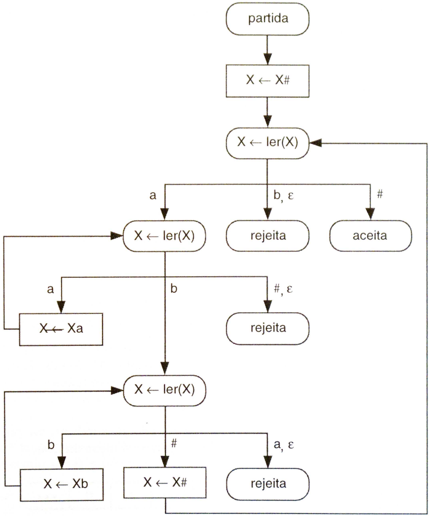
- Concatena o símbolo μ ao final da cadeia contida em X (vai para o final da fila);
- $\mu \in (\Sigma \cup \{\#\})$

Exemplo — programa P que reconhece $a^n b^n$

Estratégia:

- X contém a cadeia a ser analisada;
- Acrescentar o símbolo $\#$ ao final da mesma, para indicar final de cadeia;
- Se o primeiro símbolo for a, remover;
- Transportar todos os demais a para o final da cadeia;
- Se chegar num b, remover;
- Transportar todos os demais b para o final da cadeia;
- Repetir;
- Se a cadeia contiver apenas o símbolo de final de cadeia, ACEITA; senão REJEITA.

Exemplo — programa P que reconhece $a^n b^n$



Exemplo — programa P que reconhece $a^n b^n$

- Valores da variável X para a entrada $aaabbb$;
- Parada com a condição ACEITA:

$aaabbb$	$aaabbb\#$	$aabbb\#$
$abbb\#$	$abbb\#a$	$bbb\#a$
$bbb\#aa$	$bb\#aa$	$b\#aa$
$b\#aab$	$\#aab$	$\#aabb$
$aabb$	$aabb\#$	$abb\#$
$bb\#$	$bb\#a$	$b\#a$
$\#a$	$\#ab$	ab
$ab\#$	$b\#$	$\#$
ϵ	$\#$	ϵ

Teorema 3

Máquina de Turing $\leq$ Máquina de Post

Seja:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

uma Máquina de Turing. Então, existe uma Máquina de Post:

$$M' = (\Sigma \cup V, D, \#)$$

que simula M.

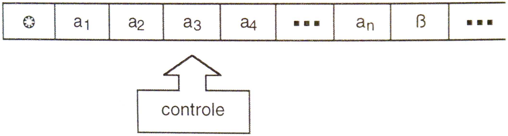
Teorema 3

Fita

- Representada pela variável X;
- A posição correntemente referenciada pelo cursor indica a primeira posição da fila contida na variável X;
- O símbolo # é usado para sinalizar o final da cadeia de entrada;
- A parte situada à esquerda da fita de entrada corresponde à parte final da fila, situada após o símbolo #.

Teorema 3

Fita



A situação da fita acima é representada na Máquina de Post por:

$$X = a_3 a_4 ... a_n \# \circ a_1 a_2$$

Teorema 3

Movimento para a DIREITA

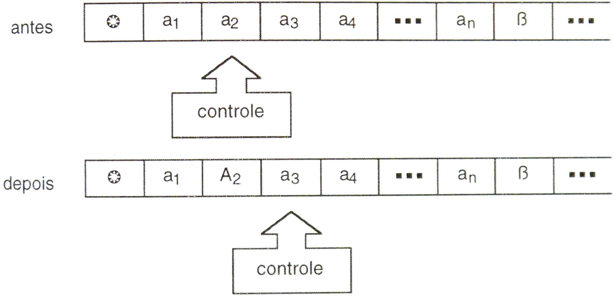
É necessário representar, na Máquina de Post:

- A substituição de um símbolo por outro, conforme a função de transição;
- O deslocamento do cursor uma posição para a direita;
- Seqüência de testes e atribuições que resultem na modificação pretendida (único teste e única atribuição).

Teorema 3

Movimento para a DIREITA

Supondo $\Pi(q_i, a_2) = (q_j, A_2, D)$:



Deseja-se mapear $a_2a_3a_4...a_n\# \circ a_1$ em $a_3a_4...a_n\# \circ a_1A_2$.

Teorema 3

Movimento para a DIREITA

Direto (quando o símbolo corrente é diferente de $\#$):

- Ler e remover o símbolo corrente a_2 do início da fila;
- Inserir o símbolo A_2 no final da fila;

Necessário considerar quando o símbolo corrente é $\#$:

- Remover $\#$ do início da fila;
- Adicionar o novo símbolo no final da fila;
- Inserir o símbolo especial $\$$ no final da fila;
- Inserir o símbolo $\#$ no final da fila.
- Transportar símbolos do início para o final, até ler (e remover) o símbolo $\$$;
- $\#abc$ - que representa (abc, q, ϵ) - é mapeado em $\#abcd$ (supondo que β é substituído por d), que representa $(abcd, q, \epsilon)$.

Teorema 3

Movimento para a DIREITA

Situação	X
INÍCIO	$\#abc$
Remove $\#$	abc
Insere d	$abcd$
Insere $\$$	$abcd\$$
Insere $\#$	$abcd\$\#$
Remove a	$bcd\$\#$
Insere a	$bcd\$\#a$

Teorema 3

Movimento para a DIREITA

Situação	X
Remove b	$cd\$\#a$
Insere b	$cd\$\#ab$
Remove c	$d\$\#ab$
Insere c	$d\$\#abc$
Remove d	$\$\#abc$
Insere d	$\$\#abcd$
Remove \$	$\#abcd$
TÉRMINO	$\#abcd$

Teorema 3

Movimento para a ESQUERDA

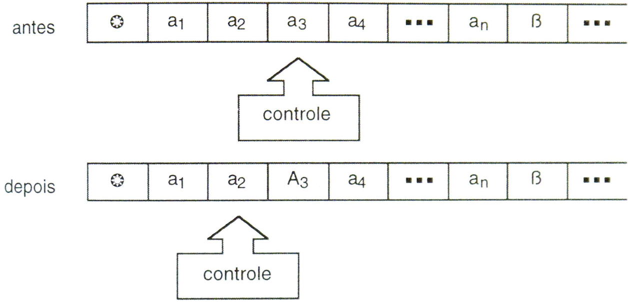
É necessário representar, na Máquina de Post:

- A substituição de um símbolo por outro, conforme a função de transição;
- O deslocamento do cursor uma posição para a esquerda;
- Seqüência de testes e atribuições que resultem na modificação pretendida.

Teorema 3

Movimento para a ESQUERDA

Supondo $\Pi(q_i, a_3) = (q_j, A_3, E)$:



Deseja-se mapear $a_3a_4...a_n\# \circ a_1a_2$ em $a_2A_3a_4...a_n\# \circ a_1$.

Teorema 3

Movimento para a ESQUERDA

Será feito em duas etapas:

- Parte 1:
  - Entrada: $a_3 a_4 ... a_n \# \circ a_1 a_2$
  - Saída: $\$ A_3 a_4 ... a_n \# \circ a_1$
- Parte 2:
  - Entrada: $\$ A_3 a_4 ... a_n \# \circ a_1$
  - Saída: $a_2 A_3 a_4 ... a_n \# \circ a_1$

Necessário, também, considerar quando o símbolo corrente é $\#$.

Teorema 3

Movimento para a ESQUERDA

Parte 1:

- Inserir $\$\$$ no final da fila;
- Inserção do novo símbolo no final da fila;
- Remover o símbolo do início da fila;
- $S_1 \leftarrow$[2] símbolo do início da fila (leitura e remoção);
- $S_2 \leftarrow$ símbolo do início da fila (leitura e remoção);
- Enquanto S_2 for diferente de $\$$ faça:
  - Inserir S_1 no final da fila;
  - $S_1 \leftarrow S_2$;
  - $S_2 \leftarrow$ símbolo do início da fila (leitura e remoção).
- Escrever S_1 na posição mais à esquerda da fila (no lugar do segundo $\$$, usando a Parte 2).

[2] pode ser simulado através do uso de estados no programa que implementa o algoritmo na Máquina de Post

Teorema 3

Movimento para a ESQUERDA

Exemplo:

- Suponha a substituição de f por X;
- Entrada: $fghij\#abcde$;
- Saída: $\$Xghij\#abcd$.

Teorema 3

Movimento para a ESQUERDA

Situação	S_1	S_2	X
INÍCIO	-	-	$fghij\#abcde$
Insere \$	-	-	$fghij\#abcde\$$
Insere \$	-	-	$fghij\#abcde\$\$$
Insere X	-	-	$fghij\#abcde\$\$X$
Remove f	-	-	$ghij\#abcde\$\$X$
Remove g	g	-	$hij\#abcde\$\$X$
Remove h	g	h	$ij\#abcde\$\$X$
Insere g	g	h	$ij\#abcde\$\$Xg$
Copia S_2 para S_1	h	h	$ij\#abcde\$\$Xg$
Remove i	h	i	$j\#abcde\$\$Xg$

Teorema 3

Movimento para a ESQUERDA

Situação	S_1	S_2	X
Insere h	h	i	$j\#abcde\$\$Xgh$
Copia S_2 para S_1	i	i	$j\#abcde\$\$Xgh$
Remove j	i	j	$\#abcde\$\$Xgh$
Insere i	i	j	$\#abcde\$\$Xghi$
Copia S_2 para S_1	j	j	$\#abcde\$\$Xghi$
Remove $\#$	j	$\#$	$abcde\$\$Xghi$
Insere j	j	$\#$	$abcde\$\$Xghij$
Copia S_2 para S_1	$\#$	$\#$	$abcde\$\$Xghij$
Remove a	$\#$	a	$bcde\$\$Xghij$

Teorema 3

Movimento para a ESQUERDA

Situação	S_1	S_2	X
Insere $\#$	$\#$	a	$bcde\$\$Xghij\#$
Copia S_2 para S_1	a	a	$bcde\$\$Xghij\#$
Remove b	a	b	$cde\$\$Xghij\#$
Insere a	a	b	$cde\$\$Xghij\#a$
Copia S_2 para S_1	b	b	$cde\$\$Xghij\#a$
Remove c	b	c	$de\$\$Xghij\#a$
Insere b	b	c	$de\$\$Xghij\#ab$
Copia S_2 para S_1	c	c	$de\$\$Xghij\#ab$
Remove d	c	d	$e\$\$Xghij\#ab$

Teorema 3

Movimento para a ESQUERDA

Situação	S_1	S_2	X
Insere c	c	d	$e\$\$Xghij\#abc$
Copia S_2 para S_1	d	d	$e\$\$Xghij\#abc$
Remove e	d	e	$\$\$Xghij\#abc$
Insere d	d	e	$\$\$Xghij\#abcd$
Copia S_2 para S_1	e	e	$\$\$Xghij\#abcd$
Remove \$	e	$\$$	$\$Xghij\#abcd$
TÉRMINO	e	$\$$	$\$Xghij\#abcd$

Teorema 3

Movimento para a ESQUERDA

Parte 2:

- Inserir o símbolo especial \$ no final da fila;
- Ler (e remover) o símbolo especial \$ do início da fila;
- Inserir o símbolo faltante no final da fila;
- Transferir os símbolos do início da fila para o final, até atingir o símbolo \$;
- Remover o símbolo \$ do início da fila.

Teorema 3

Movimento para a ESQUERDA

Exemplo (continuação do anterior):

- Entrada: $\$Xghij\#abcd$;
- Saída: $eXghij\#abcd$.

Teorema 3

Movimento para a ESQUERDA

Situação	X
INÍCIO	$\$Xghij\#abcd$
Insere \$	$\$Xghij\#abcd\$$
Remove \$	$Xghij\#abcd\$$
Insere e	$Xghij\#abcd\$e$
Remove X	$ghij\#abcd\$e$
Insere X	$ghij\#abcd\$eX$
...	...
Remove d	$\$eXghij\#abc$
Insere d	$\$eXghij\#abcd$
Remove \$	$eXghij\#abcd$
TÉRMINO	$eXghij\#abcd$

Teorema 3

Estados, aceitação e rejeição

- Instrução "Partida" simula o estado inicial q_0;
- Instrução "Aceita" simula os estados finais $q_i \in F$;
- Cada um dos demais estados corresponde a uma instrução "Desvio condicional";
- A rejeição por função de transição indefinida ou movimento inválido são simuladas pela instrução "Rejeita".

Teorema 4

Máquina de Post $\leq$ Máquina de Turing

Seja:

$$M = (\Sigma, D, \#)$$

uma Máquina de Post. Então, existe uma Máquina de Turing:

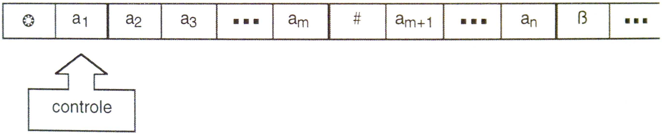
$$M' = (\Sigma, Q, \Pi, q_0, F, \{\#\}, \beta, \circ)$$

que simula M.

Teorema 4

Variável X

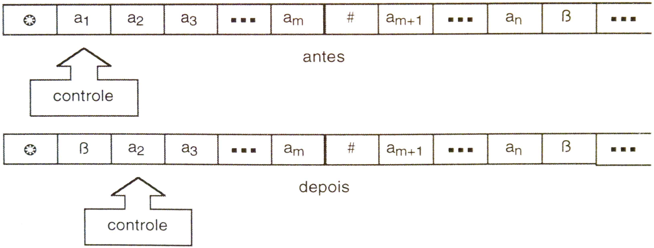
- X é simulada pela fita;
- O cursor aponta para o primeiro símbolo de X;
- Se $X = a_1 a_2 a_3 ... a_m \# a_{m+1} ... a_n$, então a representação de X na fita é:



Teorema 4

Desvio condicional

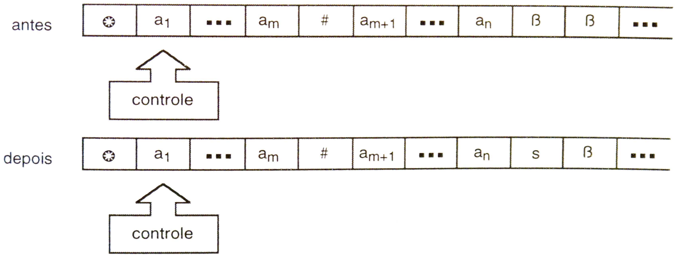
- O consumo do símbolo mais à esquerda é representado pela substituição do símbolo lido por β seguido do deslocamento do cursor para à direita;
- Se $X = a_1a_2a_3...a_m\#a_{m+1}...a_n$, um teste com o símbolo a_1 resulta em $X = a_2a_3...a_m\#a_{m+1}...a_n$



Teorema 4

Atribuição

- A atribuição de um símbolo à variável X é representado pelo acréscimo de um símbolo no final da fita seguido do retorno do cursor para a posição mais à esquerda da fita que não seja β ou $\circ$;
- Se $X = a_1a_2a_3...a_m\#a_{m+1}...a_n$, uma atribuição com o símbolo s resulta em $X = a_1a_2a_3...a_m\#a_{m+1}...a_ns$



Teorema 4

Partida, aceita e rejeita

- A instrução “Partida” é simulada pelo estado inicial q_0;
- A instrução “Aceita” é simulada por $q_F \in F$;
- A instrução “Rejeita” é simulada por movimento inválido.

Generalidades

- Formalizada por vários autores na década de 1960;
- A memória de entrada é separada das memórias auxiliar e de saída (diferente das Máquinas de Turing e de Post);
- Fita de entrada contém a cadeia a ser analisada;
- Memória auxiliar:
  - Pilha (*first-in-last-out*);
  - Uma ou mais pilhas;
  - As pilhas não tem limitação de tamanho.
- Possui um programa associado (fluxograma):
  - Partida;
  - Parada;
  - Desvio condicional (desempilha);
  - Empilha.
- Máquina Universal.

Definição

Uma Máquina com Pilhas é uma dupla:

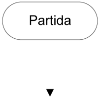
$$M = (\Sigma, D)$$

onde:

- $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ é o alfabeto de entrada;
- D é o programa (ou fluxograma), constituído pelos componentes partida, parada, desvio condicional (desempilha) e empilha;
- X representa a fita de entrada;
- $Y_i, i \geq 0$, representa as pilhas.

Definição

Componente "Partida":



- Único em cada programa P;
- Indica o início da execução de P.

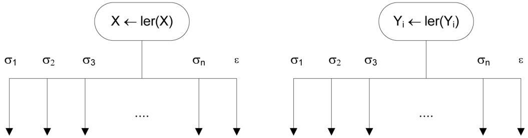
Definição

Componente “Parada”:

- Pode ser de dois tipos: parada com aceitação ou parada com rejeição;
- Múltiplas ocorrências são permitidas, sem restrições (inclusive zero ocorrências de qualquer ou de ambos os componentes).

Definição

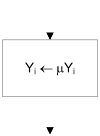
Componente “Desvio condicional (desempilha)”:



- Analisa o primeiro símbolo da entrada ou da pilha i (variáveis X e Y_i, respectivamente);
- Conforme o símbolo encontrado, desvia de acordo;
- O símbolo encontrado é removido do início da entrada ou da pilha;
- Se $|\Sigma| = n$, devem ser previstos $n+1$ desvios;
- Desvio também para o caso de X ou Y_i conter a cadeia vazia (ϵ).

Definição

Componente “Empilha”:



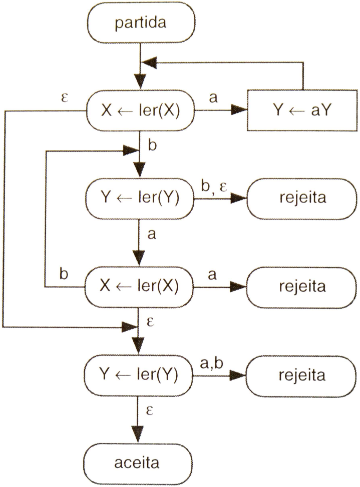
- Insere o símbolo μ no topo da pilha Y_i;
- $\mu \in \Sigma$.

Exemplo — $a^n b^n$

Estratégia:

- Usa uma única pilha;
- Ler os símbolos a da entrada (X), empilhando os mesmos em seguida (Y);
- Quando encontrar o primeiro b na entrada, começar a desempilhar os símbolos a, garantindo que para cada b em X existe um a em Y;
- Se a seqüência de símbolos b da entrada (X) acabar juntamente com a seqüência de símbolos a da pilha (Y), então ACEITA; senão, REJEITA.

Exemplo — $a^n b^n$



Exemplo — $a^n b^n$

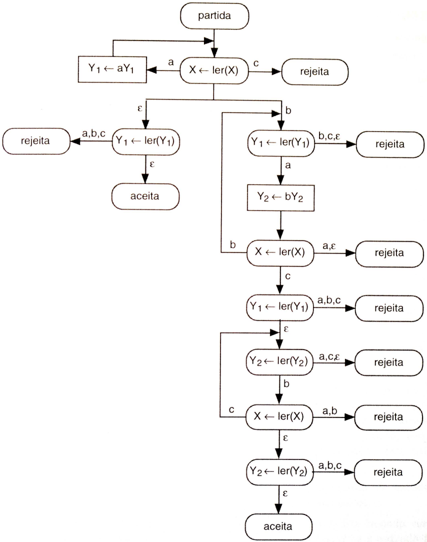
X	Y
$aaabbb$	ϵ
$aabbb$	ϵ
$aabbb$	a
$abbb$	a
$abbb$	aa
bbb	aa
bbb	aaa
bb	aaa
bb	aa
b	aa
b	a
ϵ	a
ϵ	ϵ

Exemplo — $a^n b^n c^n$

Estratégia:

- Usa duas pilhas, Y_1 e Y_2;
- Remover simbolos a da entrada, inserindo-os na piha Y_1;
- Para cada símbolo b da entrada, remover o mesmo de X, remover um símbolo a de Y_1 e inserir um símbolo b em Y_2;
- Deve-se garantir que as quantidades de a em Y_1 e b na entrada sejam idênticas;
- Para cada símbolo c da entrada, remover o mesmo de X e remover um símbolo b da Y_2;
- Se as quantidade de b em Y_2 e c em X forem iguais, ACEITA; senão REJEITA.

Exemplo — $a^n b^n c^n$



Exemplo — $a^n b^n c^n$

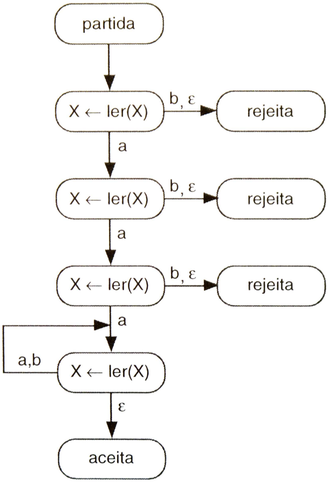
X	Y_1	Y_2
$aabbcc$	ϵ	ϵ
$abbcc$	ϵ	ϵ
$abbcc$	a	ϵ
$bbcc$	a	ϵ
$bbcc$	aa	ϵ
bcc	aa	ϵ
bcc	a	ϵ
bcc	a	b
cc	a	b
cc	ϵ	b
cc	ϵ	bb
c	ϵ	bb
c	ϵ	b
ϵ	ϵ	b
ϵ	ϵ	ϵ

Exemplo — $aaa(a|b)^*$

Estratégia:

- Não usa pilha;
- Verifica se os três primeiros símbolos da entrada X são a;
- Consome os demais símbolos da entrada;
- Quando esgotar a cadeia de entrada, ACEITA. Se as condições anteriores não forem verificadas, REJEITA.

Exemplo — $aaa(a|b)^*$



Exemplo — $aaa(a|b)^*$

X
$aaabc$
$aabc$
abc
bc
c
ϵ

Quantidade de pilhas

A classe de linguagens representadas por Máquinas com Pilhas depende da quantidade de pilhas que ela possui:

- Nenhuma pilha: corresponde ao autômato finito, capaz de reconhecer a classe das linguagens regulares (prova trivial);
- Uma pilha: corresponde ao autômato de pilha, capaz de reconhecer a classe das linguagens livres de contexto (prova trivial);
- Duas pilhas: corresponde à Máquina de Turing, capaz de aceitar a classe das linguagens recursivamente enumeráveis (prova a seguir);
- Três ou mais pilhas: podem sempre ser simuladas por uma máquina com apenas duas pilhas (não será provado).

Generalidades

- Similar à Máquina com Duas Pilhas;
- No lugar de um diagrama de fluxos usa-se um diagrama de estados;
- Componentes:
  - Fita de entrada;
  - Duas pilhas;
  - Máquina de estados.
- Máquina Universal.

Fita de entrada

- Finita;
- Contém a cadeia a ser analisada;
- Leitura apenas;
- Deslocamento do cursor para a direita apenas;
- A leitura provoca o deslocamento do cursor;
- Teste se a entrada foi esgotada;
- Leitura opcional.

Pilhas

- Tamanho ilimitado;
- Usadas como memória auxiliar;
- Leitura/escrita;
- A leitura remove o símbolo consultado (topo da pilha);
- Cada pilha é acessada por uma cabeça de leitura/escrita independente;
- Teste se a pilha está vazia;
- Leitura opcional.

Definição

Um Autômato com Duas Pilhas é uma sextupla:

$$M = (\Sigma, Q, \Pi, q_0, F, V)$$

onde:

- Σ é o alfabeto de entrada;
- Q é o conjunto de estados;
- Π é a função de transição:

$$\Pi : Q \times (\Sigma \cup \{\epsilon, ?\}) \times (V \cup \{\epsilon, ?\}) \times (V \cup \{\epsilon, ?\}) \rightarrow Q \times (V \cup \{\epsilon\}) \times (V \cup \{\epsilon\})$$

- $q_0 \in Q$ é o estado inicial;
- $F \subseteq Q$ é o conjunto de estados finais;
- V é o alfabeto auxiliar.

Parada

Para uma dada cadeia de entrada, um Autômato com Duas Pilhas pode parar e aceitar, parar e rejeitar ou ainda entrar em loop com a mesma:

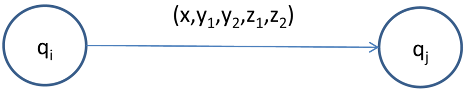
- Ele aceita se a seqüência de movimentações faz com que ele assuma um estado final (independentemente do esgotamento da cadeia de entrada ou do esvaziamento das pilhas);
- Ele rejeita se não houver possibilidade de movimentação e o estado corrente não for final.

Diagrama de estados

Se:

$$\Pi(q_i, x, y_1, z_1) = (q_j, y_2, z_2)$$

então:

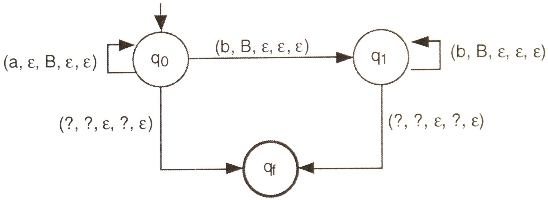


? e ϵ em Π

Seja $\Pi(q_i, x, y_1, z_1) = (q_j, y_2, z_2)$. Então:

- Se $x = \epsilon$, não lê símbolo da fita de entrada e não desloca o cursor;
- Se $x =?$, testa se a cadeia de entrada está esgotada;
- Se $y_1 = \epsilon$, não lê símbolo da primeira pilha e não desempilha símbolo;
- Se $y_1 =?$, testa se a primeira pilha está vazia;
- Se $z_1 = \epsilon$, não lê símbolo da segunda pilha e não desempilha símbolo;
- Se $z_1 =?$, testa se a segunda pilha está vazia;
- Se $y_2 = \epsilon$, mantém a primeira pilha inalterada;
- Se $z_2 = \epsilon$, mantém a segunda pilha inalterada.

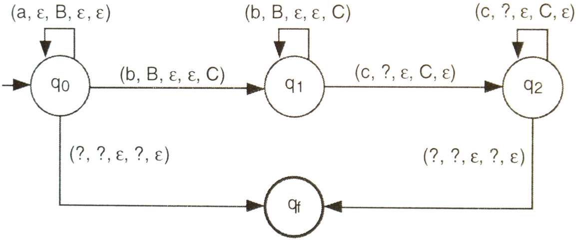
Exemplo — $a^n b^n$



Exemplo — $a^n b^n$

Estado	Entrada	Primeira pilha
q_0	$aaabbb$	ϵ
q_0	$aabbb$	B
q_0	$abbb$	BB
q_0	bbb	BBB
q_1	bb	BB
q_1	b	B
q_1	ϵ	ϵ
q_f		

Exemplo — $a^n b^n c^n$



Exemplo — $a^n b^n c^n$

Estado	Entrada	Primeira pilha	Segunda pilha
q_0	$aabbcc$	ϵ	ϵ
q_0	$abbcc$	B	ϵ
q_0	$bbcc$	BB	ϵ
q_1	bcc	B	C
q_1	cc	ϵ	CC
q_2	c	ϵ	C
q_2	ϵ	ϵ	ϵ
q_f			

Teorema 5

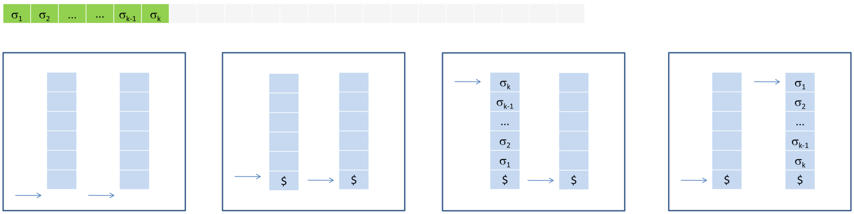
Máquina de Turing $\leq$ Autômato com Duas Pilhas

Toda Máquina de Turing pode ser simulada por algum Autômato com Duas Pilhas.

- A primeira pilha (P_1) simula o conteúdo da fita de entrada situado à esquerda da cabeça de leitura/escrita;
- A segunda pilha (P_2) simula o conteúdo da fita de entrada situado à direita da cabeça de leitura/escrita, incluindo o símbolo corrente;
- A cadeia é copiada da fita de entrada inicialmente para P_1, e depois desta para P_2;
- Marcadores de fundo de pilha \$ em P_2 (P_1) são usados para simular células em branco à direita (esquerda) do último (primeiro) símbolo diferente de branco presente na fita.

Teorema 5

Máquina de Turing $\leq$ Autômato com Duas Pilhas



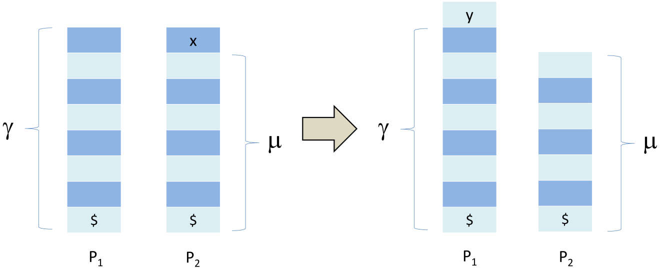
Teorema 5

Deslocamento à direita

- Considere $\Pi(q_i, x) = (q_j, y, D)$;
- Considere $P_1 = \gamma$ e $P_2 = \mu x$ (topo à direita);
- O autômato deve executar os movimentos necessários para que $P_1 = \gamma y$ e $P_2 = \mu$;
- Portanto, $(\gamma, q_i, x\mu^R) \vdash (\gamma y, q_j, \mu^R)$

Teorema 5

Deslocamento à direita



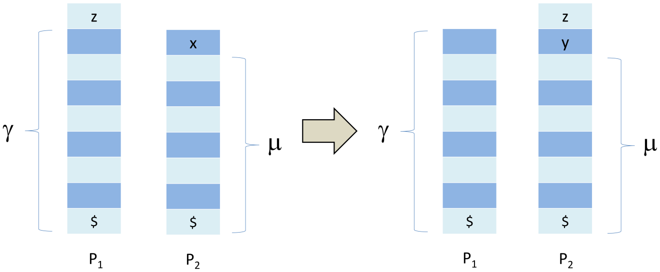
Teorema 5

Deslocamento à esquerda

- Considere $\Pi(q_i, x) = (q_j, y, E)$;
- Considere $P_1 = \gamma z$ e $P_2 = \mu x$ (topo à direita);
- O autômato deve executar os movimentos necessários para que $P_1 = \gamma$ e $P_2 = \mu y z$;
- Portanto, $(\gamma z, q_i, x\mu^R) \vdash (\gamma, q_j, zy\mu^R)$

Teorema 5

Deslocamento à esquerda



Teorema 5

Máquina de Turing $\leq$ Autômato com Duas Pilhas

- Cada estado q_i da Máquina de Turing corresponde à um estado q_i do Autômato com Duas Pilhas;
- Cada transição da Máquina de Turing entre estados q_i e q_j é mapeada numa transição do Autômato com Duas Pilhas entre os mesmos estados;
- O símbolo é lido e substituído na entrada, e o cursor é movimentado para a esquerda ou para a direita;
- Um novo estado q_j é assumido.

Máquina de Turing $\leq$ Autômato com Duas Pilhas

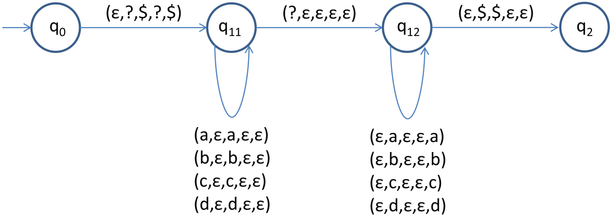
Exemplo

Suponha uma Máquina de Turing M com $\Sigma = \{a, b, c, d\}$, entrada $cddddd$ e estado inicial q_2.
O Autômato com Duas Pilhas que simula M é apresentado parcialmente a seguir (com uma transição com deslocamento para a direita e outra com deslocamento para a esquerda, as demais são semelhantes).

Máquina de Turing $\leq$ Autômato com Duas Pilhas

Exemplo

Inicialização com a cadeia $cddddd$:

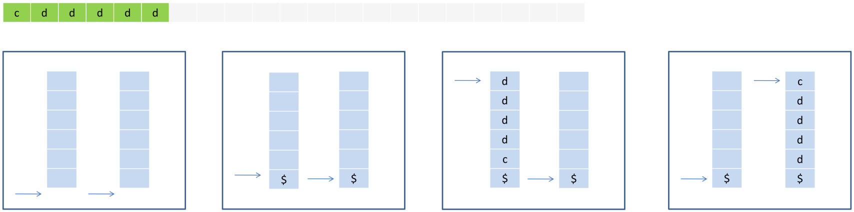


- Os marcadores de fundo de pilha ($) são inseridos ($q_0$);
- A cadeia de entrada é copiada para a pilha 1 (q_{11});
- A pilha 1 é esvaziada e copiada para a pilha 2 (q_{12}).
- A simulação pode iniciar (q_2).

Máquina de Turing $\leq$ Autômato com Duas Pilhas

Exemplo

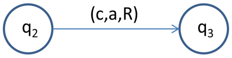
Inicialização com a cadeia $cddddd$:

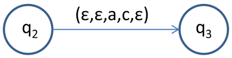


Máquina de Turing $\leq$ Autômato com Duas Pilhas

Exemplo

Suponha que M possui uma transição $\delta(q_2, c) = (q_3, a, R)$:

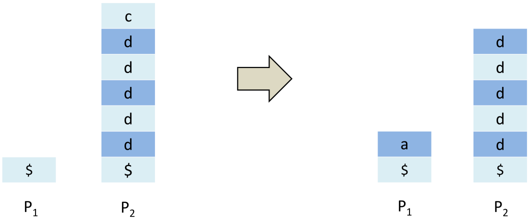




- O símbolo c é retirado da pilha 2;
- O símbolo a é inserido na pilha 1;
- A simulação continua (q_3).

Máquina de Turing $\leq$ Autômato com Duas Pilhas

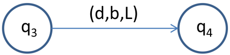
Exemplo

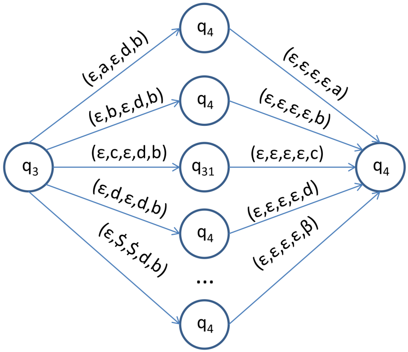


Máquina de Turing $\leq$ Autômato com Duas Pilhas

Exemplo

Suponha que M possui uma transição $\delta(q_3, d) = (q_4, b, L)$:

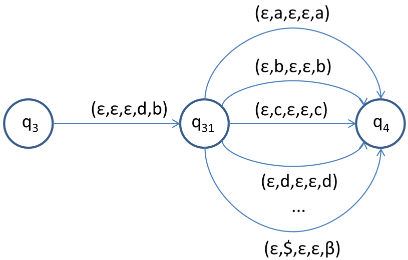




Máquina de Turing $\leq$ Autômato com Duas Pilhas

Exemplo

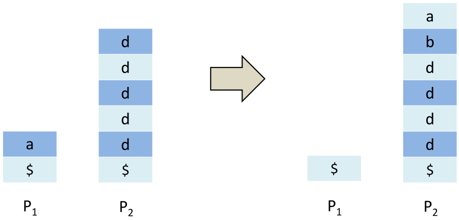
Solução alternativa:



- O símbolo d é retirado da pilha 2;
- O símbolo b é inserido na pilha 2;
- Para cada símbolo possível no topo da pilha 1, retirar e inserir na pilha 2;
- A simulação continua (q_4).

Máquina de Turing $\leq$ Autômato com Duas Pilhas

Exemplo



Teorema 6

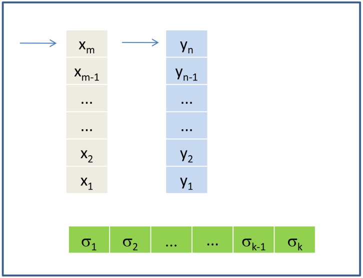
Autômato com Duas Pilhas $\leq$ Máquina de Turing

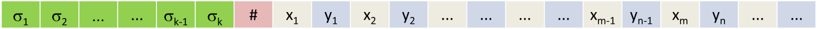
Todo Autômato com Duas Pilhas pode ser simulado por alguma Máquina de Turing.

- A cadeia de entrada ocupa as primeiras posições da fita da Máquina de Turing
- A primeira pilha (P_1) é simulada nas posições ímpares da fita da Máquina de Turing, após a cadeia de entrada;
- A segunda pilha (P_2) é simulada nas posições pares da fita da Máquina de Turing, após a cadeia de entrada;
- A cadeia $\#$ separa a cadeia de entrada de P_1 e P_2.

Teorema 6

Autômato com Duas Pilhas $\leq$ Máquina de Turing





Teorema 6

Autômato com Duas Pilhas $\leq$ Máquina de Turing

- Cada estado q_i do Autômato com Duas Pilhas corresponde à um estado q_i da Máquina de Turing;
- Cada transição do Autômato com Duas Pilhas entre estados q_i e q_j é mapeada numa transição entre os mesmos estados da Máquina de Turing;
- Para inserir ou remover um símbolo do topo da primeira (segunda) pilha, basta localizar a primeira posição ímpar (par) da fita de entrada que contém um branco;
- Para remover, basta escrever branco duas posições para a esquerda;
- Para inserir, basta substituir o branco encontrado pelo novo símbolo.

Autômato com Duas Pilhas $\leq$ Máquina de Turing

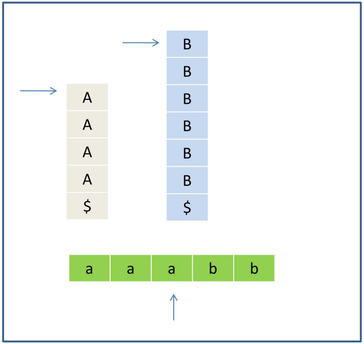
Exemplo

Suponha que A é um Autômato com Duas Pilhas e:

- $\Sigma = \{a, b\}$;
- $V = \{A, B, X, Y\}$;
- Entrada $aaabb$;
- $P_1 = AAAA$;
- $P_2 = BBBBBB$;
- $\delta(q_i, a, A, B) = (q_j, X, Y)$.

Autômato com Duas Pilhas $\leq$ Máquina de Turing

Exemplo

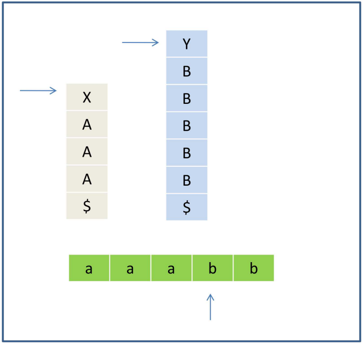


a	a	a	b	b	#	$	$	A	B	A	B	A	B	A	B		B		B

Autômato com Duas Pilhas $\leq$ Máquina de Turing

Exemplo

A transição $\delta(q_i, a, A, B) = (q_j, X, Y)$ de A pode ser simulada em uma Máquina de Turing M com o seguinte resultado:



a	a	a	b	b	#	$	$	A	B	A	B	A	B	X	B		B		Y

Autômato com Duas Pilhas $\leq$ Máquina de Turing

Exemplo

- O trecho de M que simula parcialmente a transição $\delta(q_i, a, A, B) = (q_j, X, Y)$ é apresentado a seguir;
- As seguintes etapas são executadas:
  - M identifica o símbolo corrente na fita de entrada;
  - M identifica o símbolo no topo da pilha P_1;
  - M identifica o símbolo no topo da pilha P_2;
  - Todas as informações são armazenadas no estado corrente de M.
- O símbolo M memoriza a posição que foi lida por último na fita de entrada (indicando que o símbolo lido foi a); N faz o mesmo quando o símbolo corrente é b;
- O último estado $(q_i, a, A, B, 1)$ corresponde à identificação da transição que deve ser aplicada;

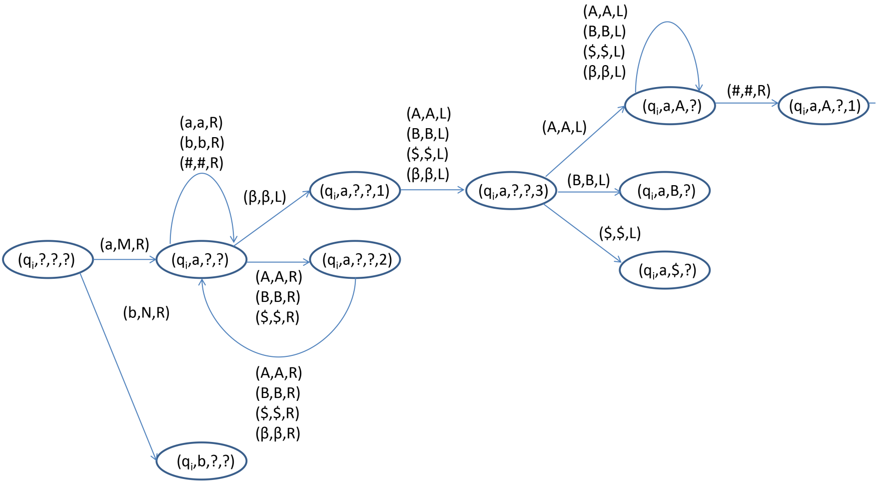
Autômato com Duas Pilhas $\leq$ Máquina de Turing

Exemplo

- A aplicação da transição deve, em seguida, modificar o topo das duas pilhas de acordo e também avançar a cabeça de leitura/escrita;
- A aplicação da transição não está representada nos slides e envolve novos estados e transições;
- A ideia, portanto (neste exemplo), é partir do estado $(q_i, ?, ?, ?)$, alcançar o estado (q_i, a, A, B) (ou seja, identificar a transição a ser aplicada), aplicar a transição e depois assumir o estado $(q_j, ?, ?, ?)$ para reiniciar o ciclo até que algum estado final seja assumido;
- Outras transições de A podem ser simuladas de forma similar em M;
- A composição de todos os trechos assim obtidos corresponde à Máquina de Turing M que simula o Autômato com Duas Pilhas A.

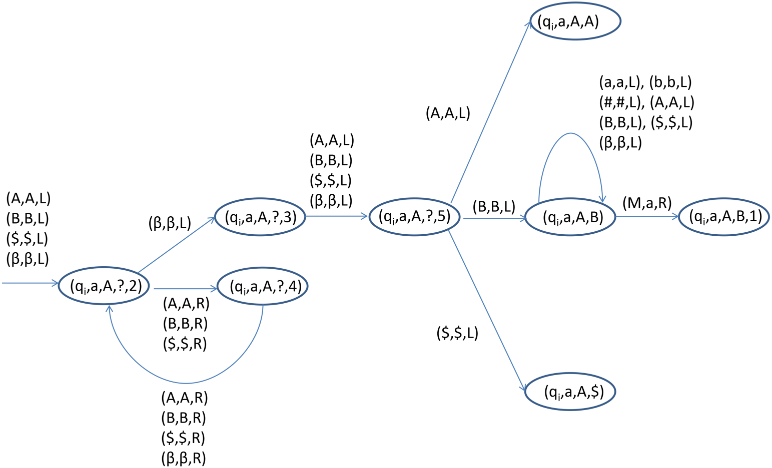
Autômato com Duas Pilhas ≤ Máquina de Turing

Exemplo



Autômato com Duas Pilhas $\leq$ Máquina de Turing

Exemplo



Conceito

Iremos explorar variações (extensões e restrições) sobre as Máquinas de Turing, mostrando que elas não alteram o poder computacional da Máquina de Turing básica.

- Extensões:
  - Fita de entrada com múltiplas trilhas;
  - Não-determinismo;
  - Múltiplas fitas de entrada.
- Restrições:
  - Nunca escrever brancos na fita de entrada;
  - Fita limitada à esquerda.

Além de reforçar a noção da Máquina de Turing como uma Máquina Universal, tais variações serão úteis na demonstração de alguns teoremas que serão vistos mais adiante.

Definição

Adotaremos uma definição um pouco diferente para Máquina de Turing, porém equivalente à anteriormente vista. Ela corresponde à definição utilizada em Hopcroft07:

- A fita é infinita em ambos os sentidos:
  Esse caso será analisado mais adiante.
- Não há marcador de início de fita:
  Pode ser simulado deslocando a cadeia de entrada uma posição para a direita e depois inserindo o marcador de início de fita na primeira posição;
- O conjunto de símbolos da fita (ou "alfabeto auxiliar") engloba o "alfabeto de entrada" ($\Sigma \subseteq \Gamma$).

Definição


Brancos
Cadeia de entrada
Brancos
.... B B a b b a c a B B ....
Fita
Cabeça de leitura e escrita
Controle finito

Formalização

Uma Máquina de Turing (determinística) é uma 7-upla:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

onde:

- Q é o conjunto (finito) de estados;
- Σ é o alfabeto de entrada;
- Γ é o conjunto de símbolos da fita, $\Sigma \subseteq \Gamma$;
- δ é a função de transição:

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

Formalização

Uma Máquina de Turing (determinística) é uma 7-upla:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

onde:

- $q_0 \in Q$ é o estado inicial;
- B representa o símbolo branco, usado para preencher todas as posições da fita não inicializadas com símbolos da cadeia de entrada; $B \in (\Gamma - \Sigma)$;
- F é o conjunto de estados finais, $F \subseteq Q$.

Complexidade no tempo

Máquina de Turing determinística

A "complexidade no tempo" ou "tempo de execução" de uma Máquina de Turing M com uma entrada w é definida como:

- A quantidade de movimentos que M executa com a entrada w até parar (aceitando ou rejeitando);
- Se M não pára com a entrada w, o tempo é infinito.

Complexidade no tempo

Máquina de Turing determinística

A “complexidade no tempo” ou “tempo de execução” de uma Máquina de Turing M é definida como:

- A função $T(n)$;
- n representa um certo comprimento da cadeia de entrada;
- $T(n)$ é a quantidade máxima de movimentos que são executados quando são consideradas todas as possíveis cadeias w de comprimento n.

Complexidade no tempo

Independentemente de fatores ou coeficientes, considera-se:

- Problemas "tratáveis" são aqueles que possuem tempo de execução polinomial, ou seja, $T(n) = O(n^k)$, para algum k;
- Problemas "intratáveis" são aqueles que possuem tempo de execução exponencial, ou seja, $T(n) = O(k^n)$, para algum k;
- Exceções à parte, funções exponenciais crescem muito mais rapidamente do que funções polinomiais;
- Problemas "tratáveis" geralmente possuem soluções viáveis em computadores; problemas "intratáveis" geralmente não possuem.

Extensões

Fita de entrada com múltiplas trilhas

Conceito

...		A							...
...		B							...
...		...							...
...		Z							...

Fita de entrada com múltiplas trilhas

Formalização

- Para uma fita de entrada com n trilhas:

$$\delta : Q \times \underbrace{\Gamma \times \Gamma ... \times \Gamma}_{\Gamma^n} \rightarrow Q \times \underbrace{\Gamma \times \Gamma ... \times \Gamma}_{\Gamma_n} \times \{L, R\}$$

  Em cada estado, o controle finito consulta o símbolo armazenado em cada uma das trilhas individualmente, providencia uma substituição para cada um deles, e desloca a cabeça de leitura/escrita para a direita ou para a esquerda;
- Se $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ é uma Máquina de Turing com n trilhas, essa máquina pode ser simulada por M' cujo conjunto de símblos da fita é $\Gamma' = \Gamma^n$; os demais elementos de M permanecem inalterados em M';
- Cada elemento de Γ^n é considerado um novo símbolo, e dessa forma um único símbolo é consultado/gravado de cada vez, como numa máquina com apenas uma trilha.

Fita de entrada com múltiplas trilhas

Exemplo

Suponha uma Máquina de Turing com 2 trilhas e:

$$\Gamma = \{a, X, B\}$$

Então:

$$\Gamma' = \{(a,a),(a,X),(a,B),(X,a),(X,X),(X,B),(B,a),(B,X),(B,B)\}$$

ou ainda $\Gamma' = \{Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9\}$ com $Y_1 = (a,a)$, $Y_2 = (a,X)$, $Y_3 = (a,B)$ etc.

- Note que o tempo de execução de M' não sofre qualquer alteração independente do número de trilhas de M.

Não-determinismo

Definição

Uma Máquina de Turing M é dita "não-determinística" se existir mais de uma possibilidade de movimentação a partir de uma mesma configuração. Formalmente:

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}$$

Não-determinismo

Linguagem definida

Seja M uma Máquina de Turing M não-determinística e $w \in \Sigma^*$. São considerados três casos, que cobrem todas as situações possíveis:

- $w \in ACEITA(M)$ se e somente se existe pelo menos uma seqüência de movimentos que conduz M a uma configuração final com a cadeia w;
- $w \in REJEITA(M)$ se e somente se todas as seqüências de movimentos de M com a cadeia w conduzem à configurações não-finais;
- $w \in LOOP(M)$ se e somente se:
  - Não existe nenhuma seqüência de movimentos que conduza M a uma configuração final com a cadeia w;
  - Existe pelo menos uma seqüência de movimentos que fazem com que M entre em loop com a cadeia w.

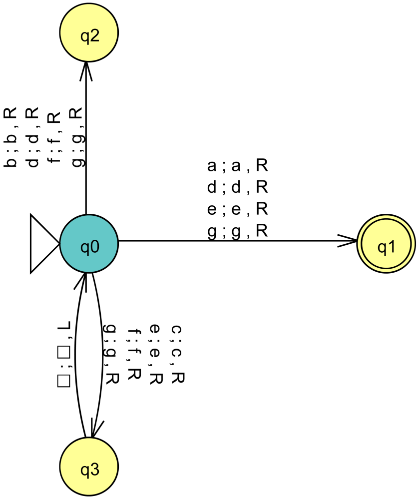
Não-determinismo

Exemplo

- A Máquina de Turing da figura seguinte é não-determinística e possui $\Sigma = \{a, b, c, d, e, f, g\}$;
- São consideradas cadeias de entrada que provocam todas as combinações possíveis entre as situações de aceitação, rejeição e loop, inclusive combinações duas a duas e as três simultaneamente;
- O resultado serve para ilustrar a determinação de $ACEITA(M), REJEITA(M)$ e $LOOP(M)$ em Máquinas de Turing não-determinísticas.

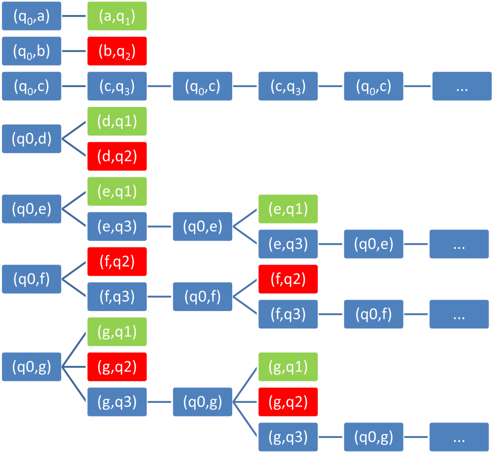
Não-determinismo

Exemplo



Não-determinismo

Exemplo



Não-determinismo

Exemplo

	CONF. FINAL	CONF. NÃO-FINAL	LOOP	
a	✓			ACEITA
b		✓		REJEITA
c			✓	LOOP
d	✓	✓		ACEITA
e	✓		✓	ACEITA
f		✓	✓	LOOP
g	✓	✓	✓	ACEITA

Não-determinismo

Exemplo

Portanto, M particiona Σ^* nos seguintes conjuntos:

- $ACEITA(M) = a\Sigma^* \,|\, d\Sigma^* \,|\, e\Sigma^* \,|\, g\Sigma^*$;
- $REJEITA(M) = b\Sigma^* \,|\, c\Sigma^+ \,|\, f\Sigma^+$;
- $LOOP(M) = c \,|\, f \,|\, \epsilon$;

Não-determinismo

Combinações de casos

Sejam M e w. As diversas seqüências de movimentação de M com w podem ser classificadas em (i) *parada em configuração final*, (ii) *parada em configuração não-final* ou (iii) *loop* (supondo o critério "Estado final"). Considere a quantidade de ocorrências de cada uma delas no conjunto de todas as ocorrências como sendo:

- "pelo menos um" (≥ 1);
- "nenhuma" (0), ou
- "todas" (all).

As tabelas seguintes mostram as várias combinações possíveis para os valores dessas três variáveis, e como a cadeia w deve ser considerada do ponto de vista de aceitação, rejeição ou loop m M. Das 27 combinações possíveis (3^3), apenas 10 são válidas.

Não-determinismo

Combinações de casos

CONF. FINAL	CONF. NÃO-FINAL	LOOP	
≥ 1	≥ 1	≥ 1	ACEITA
≥ 1	≥ 1	0	ACEITA
≥ 1	≥ 1	all	-
≥ 1	0	≥ 1	ACEITA
≥ 1	0	0	ACEITA
≥ 1	0	all	-
≥ 1	all	≥ 1	-
≥ 1	all	0	-
≥ 1	all	all	-

Não-determinismo

Combinações de casos

CONF. FINAL	CONF. NÃO-FINAL	LOOP	
0	≥ 1	≥ 1	LOOP
0	≥ 1	0	REJEITA
0	≥ 1	all	-
0	0	≥ 1	LOOP
0	0	0	-
0	0	all	LOOP
0	all	≥ 1	-
0	all	0	REJEITA
0	all	all	-

Não-determinismo

Combinações de casos

CONF. FINAL	CONF. NÃO-FINAL	LOOP	
all	≥ 1	≥ 1	-
all	≥ 1	0	-
all	≥ 1	all	-
all	0	≥ 1	-
all	0	0	ACEITA
all	0	all	-
all	all	≥ 1	-
all	all	0	-
all	all	all	-

Não-determinismo

Equivalência

Teorema:
Toda Máquina de Turing não-determinística M pode ser simulada por uma Máquina de Turing determinística M' equivalente. Ou seja:

- $ACEITA(M') = ACEITA(M)$;
- $REJEITA(M') = REJEITA(M)$;
- $LOOP(M') = LOOP(M)$.

Não-determinismo

Equivalência

Método:

- Simular as configurações de M, representando-as na fita de M';
- As configurações são delimitadas pelo símbolo especial $*$;
- A configuração corrente é marcada pelo símbolo especial X, que ocupa o lugar do $*$ situado à esquerda da mesma;
- A função de transição δ de M está armazenada no controle de M';
- Considerar a árvore de todos os caminhos possíveis;
- Fazer uma busca em largura para determinar se alguma configuração é final e aceitar quando encontrar;
- Parar e rejeitar quando não houverem novas configurações a serem consideradas;
- Parar e aceitar quando o estado corrente for final.

Não-determinismo

Equivalência

Algoritmo:

1. A fita de M' contém, inicialmente, a configuração inicial de M com a cadeia w;
2. Essa configuração é marcada como sendo a configuração corrente;
3. M' analisa a configuração corrente para determinar se o estado corrente é final;
4. Em caso afirmativo, M' pára e aceita w; em caso negativo, M' analisa a configuração corrente para determinar o estado corrente q_i e o símbolo corrente x;
5. M' insere, no final da cadeia de entrada, tantas novas configurações quantos sejam os elementos de $\delta(q_i, x)$;

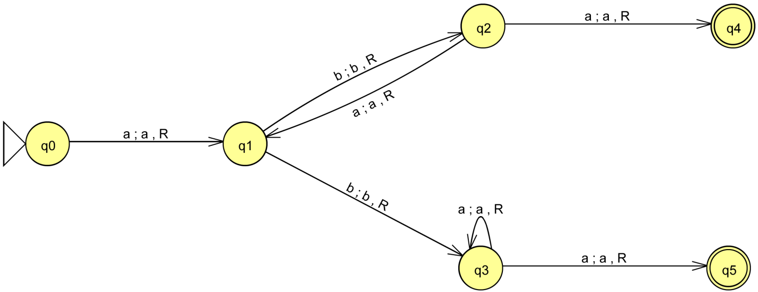
Não-determinismo

Equivalência

6. Cada uma dessas configurações é modificada para refletir a aplicação de uma particular transição:
   - Suponha que $(\alpha, q_i, x\gamma)$ seja a configuração corrente e que $\delta(q_i, x) = \{(q_j, y, R), ..., (q_m, z, R)\}$;
   - As novas configurações são $(\alpha y, q_j, \gamma), ..., (\alpha z, q_m, \gamma)$;
   - De maneira análoga se os deslocamentos forem à esquerda.
7. M' procura a próxima configuração na fita de entrada;
8. Caso não exista, M' pára e rejeita w; caso exista, vá para 2.

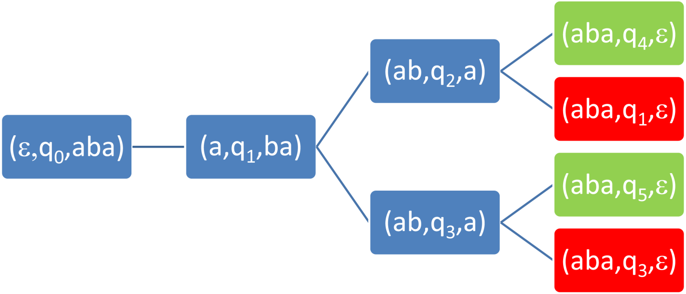
Não-determinismo

Exemplo



Não-determinismo

Exemplo



Não-determinismo

Exemplo

X	q_0	a	b	a	*														

*	q_0	a	b	a	X	a	q_1	b	a	*									

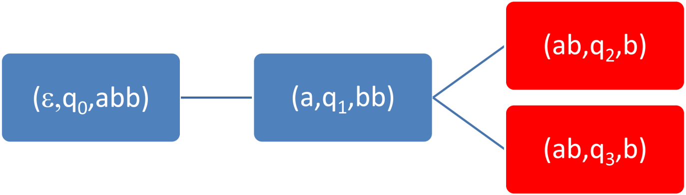
*	q_0	a	b	a	*	a	q_1	b	a	X	a	b	q_2	a	*	a	b	q_3	a
*																			

*	q_0	a	b	a	*	a	q_1	b	a	*	a	b	q_2	a	X	a	b	q_3	a
*	a	b	a	q_4	*	a	b	a	q_1	*									

*	q_0	a	b	a	*	a	q_1	b	a	*	a	b	q_2	a	*	a	b	q_3	a
X	a	b	a	q_4	*	a	b	a	q_1	*	a	b	a	q_5	*	a	b	a	q_3

Não-determinismo

Exemplo



Não-determinismo

Exemplo

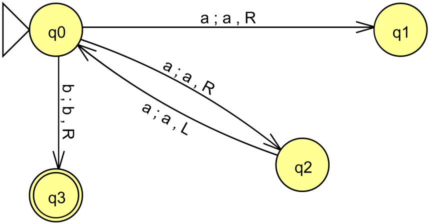
X	q_0	a	b	b	*														

*	q_0	a	b	a	X	a	q_1	b	b	*									

*	q_0	a	b	a	*	a	q_1	b	a	X	a	b	q_2	b	*	a	b	q_3	b
*																			

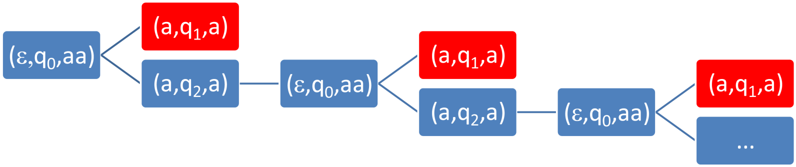
Não-determinismo

Exemplo



Não-determinismo

Exemplo



Não-determinismo

Exemplo

X	q_0	a	a	*															

*	q_0	a	a	X	a	q_1	a	*	a	q_2	a	*							

*	q_0	a	a	*	a	q_1	a	X	a	q_2	a	*	q_0	a	a	*			

*	q_0	a	a	*	a	q_1	a	*	a	q_2	a	X	q_0	a	a	*	a	q_1	a
*	a	q_2	a	*															

*	q_0	a	a	*	a	q_1	a	*	a	q_2	a	*	q_0	a	a	X	a	q_1	a
*	a	q_2	a	*	q_0	a	a	*											

*	q_0	a	a	*	a	q_1	a	*	a	q_2	a	*	q_0	a	a	*	a	q_1	a
X	a	q_2	a	*	q_0	a	a	*	a	q_1	a	*	a	q_2	a	*			

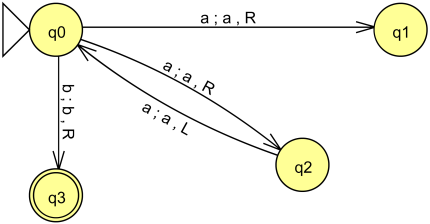
Não-determinismo

Conversão — início

- Seja M especificada no slide seguinte;
- Seja $w = aa$;
- A configuração inicial de M é (ϵ, q_0, aa);
- Os estados q_0, q_1, q_2 e q_3 de M são denotados respectivamente $0, 1, 2$ e 3 em M';
- A configuração inicial de M' é $(\epsilon, q_0, X0aa*)$;
- M' procura a configuração à direita do símbolo X;
- M' determina o estado corrente e o símbolo corrente de M.

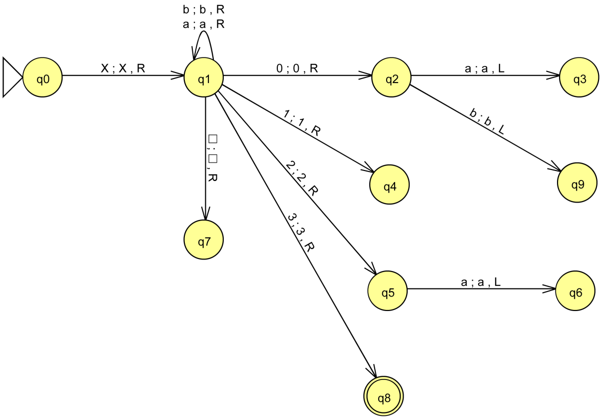
Não-determinismo

Versão não-determinística



Não-determinismo

Versão determinística equivalente (parcial)



Não-determinismo

Conversão — término

M' está especificado parcialmente:

a) A partir de q_3, gravar duas novas configurações no final da fita, uma substituindo $0a$ na configuração corrente por $a1$ e outra substituindo $0a$ por $a2$; terminar ambas com $*$; ir para (d);
b) A partir de q_9, gravar uma nova configuração no final da fita, substituindo $0b$ na configuração corrente por $b3$; terminar com $*$; ir para (d);
c) A partir de q_6, gravar uma nova configuração no final da fita, substituindo $a2$ na configuração corrente por $0a$; terminar com $*$; ir para (d);

Não-determinismo

Conversão — término

d) Procurar o X à esquerda e substituir por $*$; depois disso, procurar o primeiro $*$ à direita e substituir por X; ir para (e);

e) Deslocar a cabeça até o X e ir para o estado q_0;

f) Nos estados q_2 e q_5, se a entrada corrente for $*$ ou branco, executar os passos (d) e (e).

Não-determinismo

Conclusões

- Se M pára em uma configuração final em algum caminho, então M' pára e aceita w;
- Se M pára em configurações não-finais em todos os caminhos, então M' pára e rejeita w;
- Se M entra em loop em algum caminho, e não aceita em nenhum outro caminho, então M' entra em loop.
- Se M' pára em uma configuração final para w, então M pára e aceita w;
- Se M' pára em uma configuração não-final para w, então M pára e rejeita w;
- Se M' entra em loop, então M entra em loop em algum caminho e não existe nenhum outro caminho que seja de aceitação.

Complexidade no tempo

Máquina de Turing não-determinística

A "complexidade no tempo" ou "tempo de execução" de uma Máquina de Turing M com uma entrada w é definida como:

- A quantidade máxima de movimentos que M executa com a entrada w até parar (aceitando ou rejeitando), considerando todas as seqüências possíveis de movimentação;
- Se M não pára com a entrada w, o tempo é infinito.

Complexidade no tempo

Máquina de Turing não-determinística

A "complexidade no tempo" ou "tempo de execução" de uma Máquina de Turing M é definida como:

- A função $T(n)$;
- n representa um certo comprimento da cadeia de entrada;
- $T(n)$ é a quantidade máxima de movimentos executados quando são consideradas todas as possíveis cadeias w de comprimento n e, para cada uma delas, todas as possíveis seqüências de movimentação.

Não-determinismo

Conclusões

- Considere que M executa n movimentos na seqüência mais longa;
- Considere que o maior número de transições em qualquer configuração de M é m;
- Após a execução do primeiro movimento de M (a partir da configuração inicial) haverão, no máximo, m configurações seguintes;
- Após a execução do segundo movimento de M, haverão, no máximo, $m * m$ configurações seguintes;
- Após a execução do n-ésimo movimento de M, haverão, no máximo, m^n configurações seguintes;

Não-determinismo

Conclusões

- Portanto, o total de configurações alcançadas por M é $1 + m + m^2 + ... + m^n$;
- $1 + m + m^2 + ... + m^n \leq 1 + n * m^n, \forall m \geq 0, \forall n \geq 0$;
- Cada caminho de M analisa no máximo $1 + n$ configurações;
- M' precisa analisar, sozinha, $1 + n * m^n$ configurações;
- O tempo de execução de M' é exponencial;
- Se M é $O(n)$, então M' é $O(m^n)$;
- Se M é $O(t(n))$, então M' é $O(m^{t(n)})$.

Múltiplas fitas de entrada

Conceito

Ao invés de uma única fita, a Máquina de Turing possui uma quantidade finita de fitas:

- A cadeia de entrada é posicionada na primeira fita;
- Cada fita possui uma cabeça de leitura/escrita independente das demais;
- As transições controlam as leituras, as escritas e as movimentações de todas as cabeças;
- A cabeça pode permanecer no lugar de origem, sem se deslocar.

Múltiplas fitas de entrada

Definição

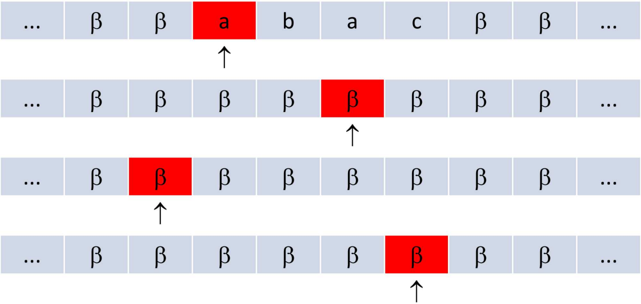
Máquina de Turing com n fitas:

$$\delta : Q \times \underbrace{\Gamma}_{\text{Fita } 1} \times \underbrace{\Gamma}_{\text{Fita } 2} \times ... \underbrace{\Gamma}_{\text{Fita } n} \rightarrow$$

$$Q \times \underbrace{(\Gamma \times \{L, R, S\})}_{\text{Fita } 1} \times \underbrace{(\Gamma \times \{L, R, S\})}_{\text{Fita } 2} ... \times \underbrace{(\Gamma \times \{L, R, S\})}_{\text{Fita } n}$$

Múltiplas fitas de entrada

Representação



Múltiplas fitas de entrada

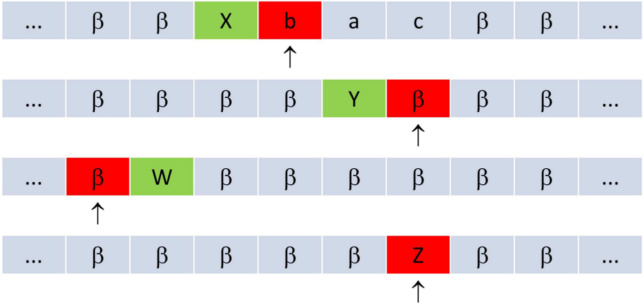
Representação

Suponha:

$$\delta(q_0, a, \beta, \beta, \beta) = (q_1, (X, R), (Y, R), (W, L), (Z, S))$$

Múltiplas fitas de entrada

Representação



Múltiplas fitas de entrada

Equivalência

Como toda Máquina de Turing é uma Máquina de Turing com múltiplas fitas, é fato que Máquinas de Turing com múltiplas fitas aceitam todas as linguagens recursivamente enumeráveis. No entanto, cabe questionar se existem linguagens que não são recursivamente enumeráveis e que são aceitas por alguma Máquina de Turing com duas ou mais fitas.

Múltiplas fitas de entrada

Equivalência

Teorema: A classe das linguagens aceitas por Máquinas de Turing com múltiplas fitas corresponde exatamente à classe das linguagens aceitas por Máquinas de Turing com uma única fita.

- MT com uma única fita simula MT com múltiplas fitas, independentemente da quantidade de fitas.

Múltiplas fitas de entrada

Convenção

- Uma MT com n fitas (M_1) será simulada por uma MT com uma única fita e $2 * n$ trilhas (M_2);
- A trilha $2 * i - 1$ representa o conteúdo da fita i, $1 \leq i \leq n$;
- A trilha $2 * i$ representa (símbolo X) a posição corrente da cabeça de leitura/escrita na fita $2 * i - 1$, $1 \leq i \leq n$;
- Exemplo para $n = 4$.

Múltiplas fitas de entrada

Exemplo

Configuração inicial de M_1:

...	β	X_1	X_2	X_3	X_4	X_5	...	β	...
		↑							
...	β	β	β	β	β	β	...	β	...
		↑							
...	β	β	β	β	β	β	...	β	...
		↑							
...	β	β	β	β	β	β	...	β	...
		↑							

Múltiplas fitas de entrada

Exemplo

Configuração inicial de M_2:

...	β	X_1	X_2	X_3	X_4	X_5	...	β	...
		X							
...	β	β	β	β	β	β	...	β	...
		X							
...	β	β	β	β	β	β	...	β	...
		X							
...	β	β	β	β	β	β	...	β	...
		X							

$\uparrow$

Múltiplas fitas de entrada

Exemplo

Configuração arbitrária de M_1:

...	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	...
	$\uparrow$								
...	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	...
			$\uparrow$						
...	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	...
					$\uparrow$				
...	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	...
							$\uparrow$		

Múltiplas fitas de entrada

Exemplo

Configuração arbitrária de M_2:

...	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	...
	X								
...	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	...
			X						
...	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	...
					X				
...	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	...
							X		

Múltiplas fitas de entrada

Equivalência

Método: M_2 simula M_1

1. M_2 precisa localizar as posições onde estão os marcadores das n cabeças de leitura/escrita de M_1 na sua fita de entrada;
2. Para não se perder, M_2 deve manter sempre, armazenado no seu conjunto de estados, a quantidade de marcadores que estão à esquerda e à direita da posição corrente de leitura/escrita;
3. Após a localização de cada marcador, M_2 deve armazenar, no seu conjunto de estados, o símbolo lido em cada uma das posições correspondentes;
4. O estado de M_1 deve estar armazenado também no conjunto de estados de M_2;

Múltiplas fitas de entrada

Equivalência

Método: M_2 simula M_1

5. M_2 determina a transição a ser aplicada;
6. M_2 revisita cada um dos marcadores, desloca os mesmos de posição (se for o caso), substitui os símbolos correspondentes e registra uma eventual mudança de estado de M_1 no seu próprio conjunto de estados;
7. Os estados de aceitação de M_2 são aqueles que representam estados de aceitação de M_1.

Múltiplas fitas de entrada

Exemplo

Considere $M_1 = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, X\}, \delta_1, q_0, F_1)$ com 3 fitas de entrada. Os estados de Q_2 são elementos de:

$$Q_1 \times \{(0,3),(1,2),(2,1),(3,0)\} \times (\Gamma \cup \{?\}) \times (\Gamma \cup \{?\}) \times (\Gamma \cup \{?\})$$

- Q_1 representa o estado corrente de M_1;
- $(0,3),(1,2),(2,1)$ e $(3,0)$ representam a quantidade de marcadores X que estão, respectivamente, à esquerda e à direita (ou sob) a cabeça de leitura/escrita em M_2;
- $\Gamma \cup \{?\}$ representa o símbolo lido da respectiva fita (trilha); $?$ indica que o símbolo ainda não foi lido.

Múltiplas fitas de entrada

Exemplo

Portanto, $M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, (q_0, (0,3), ?, ?, ?), F_2)$, M_2 possui seis trilhas e o seu estado inicial:

$$(q_0, (0,3), ?, ?, ?)$$

indica, simultaneamente:

- Estado q_0 (inicial de M_1);
- Cabeça de leitura/escrita posicionada de tal forma que os três marcadores X encontram-se à direita ou sob o mesmo;
- Nenhum símbolo de nenhuma fita (trilha) foi lido ainda.

Múltiplas fitas de entrada

Exemplo

Na medida em que os símbolos de cada uma das trilhas ímpares (correspondentes a cada uma das fitas) vão sendo lidos, o estado corrente se modifica. Suponha, por exemplo, que os símbolos destas trilhas sejam, respectivamente, a, b e c. Então, os próximos estados assumidos por M_2 seriam, na seqüência:

- $(q_0, (0, 3), ?, ?, ?)$
- $(q_0, (1, 2), a, ?, ?)$
- $(q_0, (2, 1), a, b, ?)$
- $(q_0, (3, 0), a, b, c)$

ou q_{01}, q_{02}, q_{03} etc, caso sejam necessários estados intermediários para executar as operações de M_2.

Múltiplas fitas de entrada

Exemplo

No estado $(q_0, (3,0), a, b, c)$, M_2 reúne todas as informações necessárias para escolher e, em seguida, aplicar alguma uma transição de M_1 (se ela existir), uma vez que M_2 conhece a função de transição de M_1. Em particular:

- M_2 conhece o estado corrente de M_1 (q_0);
- M_2 conhece os símbolos correntes de cada uma das fitas de M_1 (a, b e c).

Inicia-se, então, o procedimento para fazer a modificação do conteúdo das seis trilhas de M_2 para refletir a aplicação desta transição. Supondo que, depois disso, os marcadores X estejam novamente à direita (ou sob) a cabeça de leitura/escrita, e que o novo estado de M_1 seja q_1, então o novo estado de M_2 será $(q_1, (0,3), ?, ?, ?)$ e o ciclo se repete até que algum estado final seja alcançado.

Múltiplas fitas de entrada

Equivalência

Teorema: O tempo que M_2 (com uma única fita) leva para simular n movimentos de M_1 (com múltiplas fitas) é $O(n^2)$.

Múltiplas fitas de entrada

Equivalência

- Após 1 movimento de M_1, os marcadores de M_2 estarão separados por no máximo 2 posições; após 2 movimentos de M_1, os marcadores de M_2 estarão separados por no máximo 4 posições;
- Após n movimentos de M_1, os marcadores de M_2 estarão separados por no máximo $2 * n$ posições;
- O cursor de leitura de M_2 se posiciona inicialmente à esquerda do marcador mais à esquerda (ou apontando para ele);
- Para localizar todos os marcadores, M_2 deve executar, no máximo, $2 * i$ movimentos para a direita, onde i é o número de movimentos executados por M_1 até o momento;
- Uma vez determinada a transição a ser aplicada, M_2 deve substituir os símbolos nas trilhas ímpares na fita de entrada; para isso são requeridos, no máximo, $2 * i$ movimentos para a esquerda;

Múltiplas fitas de entrada

Equivalência

- Também é necessário deslocar os marcadores das trilhas pares para a esquerda ou para a direita; para isso serão necessários outros 2 movimentos por marcador (um em cada sentido), num total de $2*k$ movimentos, onde k é o número de fitas sendo simuladas. Esse cálculo independe do valor de i.

Movimentos de M_1	Distância máxima em M_2	Movimentos de M_2
1	2	$4+2*k$
2	4	$8+2*k$
...	...	...
n	$2*n$	$4*n+2*k$

Múltiplas fitas de entrada

Equivalência

- Portanto, para simular n movimento de M_1 são requeridos $\sum_{i=1}^{n}(4*i+2*k) \leq n*(4*n+2*k) = 4*n^2+2*k*n$ movimentos;
- Logo, para simular n movimentos de M_1 serão requeridos, no máximo, $O(n^2)$ movimentos.

Restrições

Nunca escrever brancos na fita de entrada

Objetivo

Máquinas de Turing que não escrevem branco na fita de entrada são úteis na demonstração de alguns teoremas.

- Toda MT que escreve brancos na fita pode ser convertida numa equivalente que não escreve brancos na fita;
- Não escrever brancos na fita de entrada garante que a mesma é composta por uma seqüência finita de símbolos não-brancos seguida de uma seqüência infinita de brancos (não há fragmentação);
- Se, adicionalmente, a fita de entrada for também limitada à esquerda, então a seqüência de símbolos não-brancos inicia na primeira posição da fita de entrada.

Nunca escrever brancos na fita de entrada

Algoritmo de conversão

Seja $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_{01}, B, F_1)$. Então:

- Fazer $\Gamma_1 \leftarrow \Gamma_1 \cup \{B'\}$;
  (criar um novo símbolo B' para representar o branco B)
- Substituir toda regra do tipo $\delta_1(q, X) = (p, B, D)$, $D \in \{L, R\}$, por $\delta_1(q, X) = (p, B', D)$;
  (escrever B' em vez de B)
- $\forall q \in Q_1$, fazer $\delta_1(q, B') = \delta_1(q, B)$.
  (ler B' em vez de B)

Fita limitada à esquerda

Conceito

Ao invés de uma fita com tamanho ilimitado em ambos os sentidos, a Máquina de Turing possui uma fita limitada à esquerda e sem limitação à direita:

- A fita possui duas trilhas;
- A cadeia de entrada é posicionada na primeira trilha no início da fita;
- A fita é preenchida com brancos à direita do último símbolo da cadeia de entrada na primeira trilha, e integralmente na segunda trilha;
- Qualquer tentativa de movimentação da cabeça de leitura/escrita para a esquerda da primeira posição da fita gera uma condição de parada com rejeição da cadeia de entrada.

Fita limitada à esquerda

Equivalência

Teorema: A classe das linguagens aceitas por Máquinas de Turing com fita ilimitada em ambos os sentidos corresponde exatamente à classe das linguagens aceitas por Máquinas de Turing com fita limitada à esquerda.

- MT M_2 com fita limitada à esquerda simula MT M_1 com fita ilimitada;
- A cabeça de leitura/escrita de M_2 nunca se desloca para a esquerda da primeira posição.

Fita limitada à esquerda

Representação

Fita ilimitada em ambos os sentidos

...	X_{-7}	X_{-6}	X_{-5}	X_{-4}	X_{-3}	X_{-2}	X_{-1}	X_0	X_1	X_2	X_3	X_4	X_5	X_6	...

Fita limitada à esquerda

X_0	X_1	X_2	X_3	X_4	X_5	X_6	...
*	X_{-1}	X_{-2}	X_{-3}	X_{-4}	X_{-5}	X_{-6}	...

Fita limitada à esquerda

Equivalência

Método:

- Usar a trilha superior para representar o lado direito da fita, e a trilha inferior para representar o lado esquerdo da fita;
- Memorizar, nos estados de M_2, se a cabeça de leitura/escrita está posicionada à esquerda ou à direita da posição inicial em M_1;
- Conforme o estado de M_2, manipular apenas a trilha superior ou inferior da fita de entrada;
- Garantir que toda movimentação para a direita da primeira posição seleciona a trilha superior, e que toda movimentação à esquerda da primeira posição seleciona a trilha inferior.

Fita limitada à esquerda

Equivalência

Considere $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_{01}, B, F_1)$ modificado para nunca escrever brancos na fita. M_1 é simulado por M_2 com fita limitada à esquerda, $M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, q_{02}, (B, B), F_2)$, onde:

- $Q_2 = \{q_{02}, q_{12}\} \cup (Q_1 \times \{U, L\})$;
  U indica a manipulação da trilha superior da fita, L indica a manipulação da trilha inferior;
- $\Sigma_2 = \Sigma \times \{B\}$;
- $\Gamma_2 = (\Gamma_1 \times \Gamma_1) \cup \{(X, *) \mid X \in \Gamma_1\}$;
  $* \notin \Gamma_1$ é usado para indicar o início da fita.
- (B, B) representa o branco de M_2;
- $F_2 = F_1 \times \{U, L\}$.

Fita limitada à esquerda

Equivalência

Obtenção de δ_2:

1. $\delta_2(q_{02}, (\sigma, B)) = (q_{12}, (\sigma, *), R)$, $\forall \sigma \in (\Sigma \cup \{B\})$;
   primeiro movimento: inserção do marcador de início de fita na segunda trilha;
2. $\delta_2(q_{12}, (X, B)) = ((q_{01}, U), (X, B), L)$, $\forall X \in \Gamma_1$;
   segundo movimento: retornar para a posição inicial da fita, selecionar trilha superior (parte direita da fita de M_2) e ir para o estado inicial q_{01};
3. Se $\delta_1(q, X) = (p, Y, D)$, então, $\forall Z \in \Gamma_1$:
   - $\delta_2((q, U), (X, Z)) = ((p, U), (Y, Z), D)$ e
   - $\delta_2((q, L), (Z, X)) = ((p, L), (Z, Y), \overline{D})$.

   simula M_1 levando em consideração a trilha corrente, exceto se estiver na primeira posição.

Fita limitada à esquerda

Equivalência

Obtenção de δ_2:

4. Se $\delta_1(q, X) = (p, Y, R)$, então:
   $\delta_2((q, L), (X, *)) = ((p, U), (Y, *), R)$
   $\delta_2((q, U), (X, *)) = ((p, U), (Y, *), R)$
   deslocamento à direita da primeira posição seleciona a trilha superior;
5. Se $\delta_1(q, X) = (p, Y, L)$, então:
   $\delta_2((q, L), (X, *)) = ((p, L), (Y, *), R)$
   $\delta_2((q, U), (X, *)) = ((p, L), (Y, *), R)$
   deslocamento à esquerda a primeira posição seleciona trilha inferior.

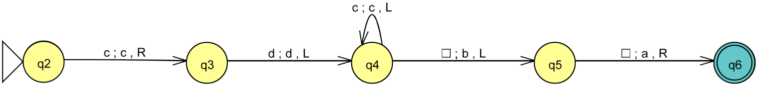
Fita limitada à esquerda

Conclusão

- M_2 reproduz as configurações de M_1;
- M_2 entra em um estado de aceitação se e somente se M_1 também entra;
- $L(M_2) = L(M_1)$.

Fita limitada à esquerda

Exemplo



Fita limitada à esquerda

Exemplo

- $Q_2 = \{q_{02}, q_{12},$
  $(q_2, U), (q_2, L), (q_3, U), (q_3, L),$
  $(q_4, U), (q_4, L), (q_5, U), (q_5, L),$
  $(q_6, U), (q_6, L)\}$
- $\Sigma_2 = \{(a, B), (b, B), (c, B), (d, B)\}$
- $\Gamma_2 = \{(a, *), (b, *), (c, *), (d, *),$
  $(a, a), (a, b), (a, c), (a, d), (a, B),$
  $(b, a), (b, b), (b, c), (b, d), (b, B),$
  $(c, a), (c, b), (c, c), (c, d), (c, B),$
  $(d, a), (d, b), (d, c), (d, d), (d, B),$
  $(B, a), (B, b), (B, c), (B, d), (B, B)\}$
- $F_2 = \{(q_6, U), (q_6, L)\}$

Fita limitada à esquerda

Exemplo

Obtenção de δ_2:

1. $\delta_2(q_{02}, (a, B)) = (q_{12}, (a, *), R)$
   $\delta_2(q_{02}, (b, B)) = (q_{12}, (b, *), R)$
   $\delta_2(q_{02}, (c, B)) = (q_{12}, (c, *), R)$
   $\delta_2(q_{02}, (d, B)) = (q_{12}, (d, *), R)$
   $\delta_2(q_{02}, (B, B)) = (q_{12}, (B, *), R)$
2. $\delta_2(q_{12}, (a, B)) = ((q_2, U), (a, B), L)$
   $\delta_2(q_{12}, (b, B)) = ((q_2, U), (b, B), L)$
   $\delta_2(q_{12}, (c, B)) = ((q_2, U), (c, B), L)$
   $\delta_2(q_{12}, (d, B)) = ((q_2, U), (d, B), L)$
   $\delta_2(q_{12}, (B, B)) = ((q_2, U), (B, B), L)$

Fita limitada à esquerda

Exemplo

A partir de $\delta_1(q_2, c) = (q_3, c, R)$:

3. $\delta_2((q_2, U), (c, a)) = ((q_3, U), (c, a), R)$
   $\delta_2((q_2, U), (c, b)) = ((q_3, U), (c, b), R)$
   $\delta_2((q_2, U), (c, c)) = ((q_3, U), (c, c), R)$
   $\delta_2((q_2, U), (c, d)) = ((q_3, U), (c, d), R)$
   $\delta_2((q_2, U), (c, B)) = ((q_3, U), (c, B), R)$
   $\delta_2((q_2, L), (a, c)) = ((q_3, L), (a, c), L)$
   $\delta_2((q_2, L), (b, c)) = ((q_3, L), (b, c), L)$
   $\delta_2((q_2, L), (c, c)) = ((q_3, L), (c, c), L)$
   $\delta_2((q_2, L), (d, c)) = ((q_3, L), (d, c), L)$
   $\delta_2((q_2, L), (B, c)) = ((q_3, L), (B, c), L)$

Fita limitada à esquerda

Exemplo

A partir de $\delta_1(q_3, d) = (q_4, d, L)$:

3. $\delta_2((q_3, U), (d, a)) = ((q_4, U), (d, a), L)$
   $\delta_2((q_3, U), (d, b)) = ((q_4, U), (d, b), L)$
   $\delta_2((q_3, U), (d, c)) = ((q_4, U), (d, c), L)$
   $\delta_2((q_3, U), (d, d)) = ((q_4, U), (d, d), L)$
   $\delta_2((q_3, U), (d, B)) = ((q_4, U), (d, B), L)$
   $\delta_2((q_3, L), (a, d)) = ((q_4, L), (a, d), R)$
   $\delta_2((q_3, L), (b, d)) = ((q_4, L), (b, d), R)$
   $\delta_2((q_3, L), (c, d)) = ((q_4, L), (c, d), R)$
   $\delta_2((q_3, L), (d, d)) = ((q_4, L), (d, d), R)$
   $\delta_2((q_3, L), (B, d)) = ((q_4, L), (B, d), R)$

Fita limitada à esquerda

Exemplo

A partir de $\delta_1(q_4, c) = (q_4, c, L)$:

3. $\delta_2((q_4, U), (c, a)) = ((q_4, U), (c, a), L)$
   $\delta_2((q_4, U), (c, b)) = ((q_4, U), (c, b), L)$
   $\delta_2((q_4, U), (c, c)) = ((q_4, U), (c, c), L)$
   $\delta_2((q_4, U), (c, d)) = ((q_4, U), (c, d), L)$
   $\delta_2((q_4, U), (c, B)) = ((q_4, U), (c, B), L)$
   $\delta_2((q_4, L), (a, c)) = ((q_4, L), (a, c), R)$
   $\delta_2((q_4, L), (b, c)) = ((q_4, L), (b, c), R)$
   $\delta_2((q_4, L), (c, c)) = ((q_4, L), (c, c), R)$
   $\delta_2((q_4, L), (d, c)) = ((q_4, L), (d, c), R)$
   $\delta_2((q_4, L), (B, c)) = ((q_4, L), (B, c), R)$

Fita limitada à esquerda

Exemplo

A partir de $\delta_1(q_4, B) = (q_5, b, L)$:

3. $\delta_2((q_4, U), (B, a)) = ((q_5, U), (b, a), L)$
   $\delta_2((q_4, U), (B, b)) = ((q_5, U), (b, b), L)$
   $\delta_2((q_4, U), (B, c)) = ((q_5, U), (b, c), L)$
   $\delta_2((q_4, U), (B, d)) = ((q_5, U), (b, d), L)$
   $\delta_2((q_4, U), (B, B)) = ((q_5, U), (b, B), L)$
   $\delta_2((q_4, L), (a, B)) = ((q_5, L), (a, b), R)$
   $\delta_2((q_4, L), (b, B)) = ((q_5, L), (b, b), R)$
   $\delta_2((q_4, L), (c, B)) = ((q_5, L), (c, b), R)$
   $\delta_2((q_4, L), (d, B)) = ((q_5, L), (d, b), R)$
   $\delta_2((q_4, L), (B, B)) = ((q_5, L), (B, b), R)$

Fita limitada à esquerda

Exemplo

A partir de $\delta_1(q_5, B) = (q_6, a, R)$:

3. $\delta_2((q_5, U), (B, a)) = ((q_6, U), (a, a), R)$
   $\delta_2((q_5, U), (B, b)) = ((q_6, U), (a, b), R)$
   $\delta_2((q_5, U), (B, c)) = ((q_6, U), (a, c), R)$
   $\delta_2((q_5, U), (B, d)) = ((q_6, U), (a, d), R)$
   $\delta_2((q_5, U), (B, B)) = ((q_6, U), (a, B), R)$
   $\delta_2((q_5, L), (a, B)) = ((q_6, L), (a, a), L)$
   $\delta_2((q_5, L), (b, B)) = ((q_6, L), (b, a), L)$
   $\delta_2((q_5, L), (c, B)) = ((q_6, L), (c, a), L)$
   $\delta_2((q_5, L), (d, B)) = ((q_6, L), (d, a), L)$
   $\delta_2((q_5, L), (B, B)) = ((q_6, L), (B, a), L)$

Fita limitada à esquerda
Exemplo

A partir de $\delta_1(q_2, c) = (q_3, c, R)$:

4. $\delta_2((q_2, L), (c, *)) = ((q_3, U), (c, *), R)$
4. $\delta_2((q_2, U), (c, *)) = ((q_3, U), (c, *), R)$

A partir de $\delta_1(q_5, B) = (q_6, a, R)$:

4. $\delta_2((q_5, L), (B, *)) = ((q_6, U), (a, *), R)$
4. $\delta_2((q_5, U), (B, *)) = ((q_6, U), (a, *), R)$

Fita limitada à esquerda

Exemplo

A partir de $\delta_1(q_3, d) = (q_4, d, L)$:

5. $\delta_2((q_3, L), (d, *)) = ((q_4, L), (d, *), R)$
5. $\delta_2((q_3, U), (d, *)) = ((q_4, L), (d, *), R)$

A partir de $\delta_1(q_4, c) = (q_4, c, L)$:

5. $\delta_2((q_4, L), (c, *)) = ((q_4, L), (c, *), R)$
5. $\delta_2((q_4, U), (c, *)) = ((q_4, L), (c, *), R)$

A partir de $\delta_1(q_4, B) = (q_5, b, L)$:

5. $\delta_2((q_4, L), (B, *)) = ((q_5, L), (b, *), R)$
5. $\delta_2((q_4, U), (B, *)) = ((q_5, L), (b, *), R)$

Fita limitada à esquerda

Transições

M_1:

- 5 transições.

M_2:

- Inicialização: 5+5 transições (casos 1 e 2);
- Posições 2 em diante: 10*5 transições (caso 3);
- Posição 1, direita: 2*2 transições (caso 4);
- Posição 1, esquerda: 3*2 transições (caso 5).

ou seja, é necessária uma MT M_2 com 70 transições para simular uma MT M_1 com apenas 5 transições.

Fita limitada à esquerda

Exemplo

O slide seguinte ilustra a operação de M_1 (com fita ilimitada em ambos os sentidos) com a cadeia de entrada cdd.

Fita limitada à esquerda

Exemplo

...	B	B	B	c	d	d	B	B	...

$\uparrow q_2$

...	B	B	B	c	d	d	B	B	...

$\uparrow q_3$

...	B	B	B	c	d	d	B	B	...

$\uparrow q_4$

...	B	B	B	c	d	d	B	B	...

$\uparrow q_4$

...	B	B	b	c	d	d	B	B	...

$\uparrow q_5$

...	B	a	b	c	d	d	B	B	...

$\uparrow q_6$

Fita limitada à esquerda

Exemplo

Os slides seguintes ilustram a operação de M_2 (com fita limitada à esquerda) com a mesma cadeia de entrada cdd.

Fita limitada à esquerda

Exemplo

c	d	d	B	B	...
B	B	B	B	B	...

↑ q_{02}

c	d	d	B	B	...
*	B	B	B	B	...

↑ q_{12}

c	d	d	B	B	...
*	B	B	B	B	...

↑ (q_2,U)

c	d	d	B	B	...
*	B	B	B	B	...

↑ (q_3,U)

Fita limitada à esquerda

Exemplo

c	d	d	B	B	...
*	B	B	B	B	...
↑	(q_4,U)				

c	d	d	B	B	...
*	B	B	B	B	...
	↑	(q_4,L)			

c	d	d	B	B	...
*	b	B	B	B	...
		↑	(q_5,L)		

c	d	d	B	B	...
*	b	a	B	B	...
	↑	(q_6,L)			