Resumo de Linguagens e Computação

Conceitos iniciais

Σ – alfabeto, conjunto finito de símbolos

Σ^n – conjunto de cadeias de comprimento n compostas por símbolos do alfabeto Σ

ε – cadeia vazia

Σ^0 – conjunto formado apenas pela cadeia vazia

Σ^* – $\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^n, n \in \mathbb{N}$

Σ^+ – $\Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^n, n \in \mathbb{N}$

Linguagem – conjunto de símbolos que obedece a uma propriedade num alfabeto. Sendo Σ um alfabeto, diz-se que a linguagem L é uma linguagem sobre Σ se $L \subseteq \Sigma^*$.

Para definir linguagens podem usar-se **construtores de conjuntos**:

$\{ w \mid$ algo sobre $w \}$, ex: $\{ w \mid w$ é um inteiro primo em notação binária $\}$

Problema: pode ser encarado como a decisão se uma cadeia pertence ou não a uma linguagem.

Autómato Finito Determinista (DFA)

Notação formal: $A = (Q, \Sigma, \delta, q_0, F)$

Q: conjunto finito de estados do autómato

Σ: alfabeto (conjunto de símbolos) de entrada

δ: função de transição; tem como parâmetros um estado e um símbolo de entrada devolvendo um estado: $\delta(q, a) = p$ em que q é o estado de entrada, a o símbolo de entrada e p o estado de saída

q_0: estado inicial

F: conjunto de estados finais; F é um subconjunto de Q

Função de transição estendida: Se δ for a função de transição de um autómato DFA, então a sua função estendida é a função $\hat{\delta}$ que aceita um estado q e uma cadeia (palavra) w e devolve o estado atingido p aplicando a δ sucessivamente aos símbolos da cadeia e estados intermédios:

$$\hat{\delta}(q, xw) = \delta(\hat{\delta}(q, w), x)$$

Linguagem de um DFA: conjunto de cadeias que aplicadas com o estado inicial pela função de transição estendida a um autómato, devolvem um dos estados finais. Ou seja:

$L(A) = \{ w \mid \hat{\delta}(q_0, w) \in F \}$ se $A = (Q, \Sigma, \delta, q_0, F)$

Diagrama de transição (representa a função de transição):

$A = \{ x01y \mid x, y \in \{0, 1\} \}$

$A = (\{ q_0, q_1, q_2 \}, \{ 0, 1 \}, \delta, q_0, \{ q_2 \})$

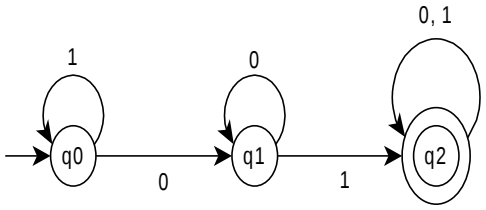


Tabela de transição (representa a função de transição):

	0	1
→ q_0	q_1	q_0
q_1	q_1	q_2
* q_2	q_2	q_2

Autómato Finito Não Determinista (NFA)

Notação formal: A = (Q, Σ, δ, q_0, F)

Q: conjunto finito de estados do autómato

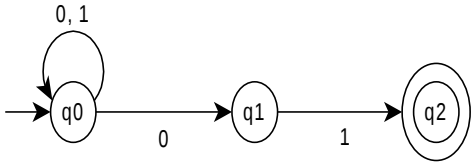
Σ: alfabeto (conjunto de símbolos) de entrada

δ: função de transição; tem como parâmetros um estado e um símbolo de entrada devolvendo um conjunto de estados de Q: $\delta(q, a) = E$ em que q é o estado de entrada, a o símbolo de entrada e E o conjunto de estados de saída, contido em Q

q_0: estado inicial

F: conjunto de estados finais; F é um subconjunto de Q

Exemplo: { w | $w \in$ { 0, 1 } e termina em 01 }



	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	∅	$\{q_2\}$
* q_2	∅	∅

Função estendida de um NFA: função que toma um estado inicial e uma cadeia e devolve o conjunto de estados atingidos depois de seguir todos os caminhos pela aplicação sucessiva da função de transição do NFA.

Linguagem de um NFA: conjunto de cadeias que aplicadas pela função estendida ao estado inicial de um NFA devolvem um conjunto de estados em que existe pelo menos um estado final:

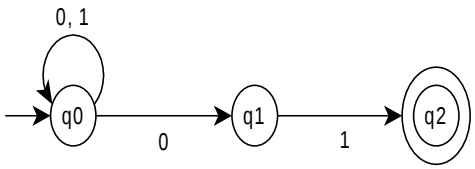
L(A) = { w | $\hat{\delta}(q_0, w) \cap F \neq \emptyset$ } se A = (Q, Σ, δ, q_0, F)

Equivalência entre um DFA e um NFA: são equivalentes e, na prática, um DFA pode ter tantos estados quantos os de um NFA, embora com mais transições. No entanto, um DFA pode ter 2^n estados para um NFA equivalente com apenas n, no caso extremo.

Conversão de um NFA em um DFA:

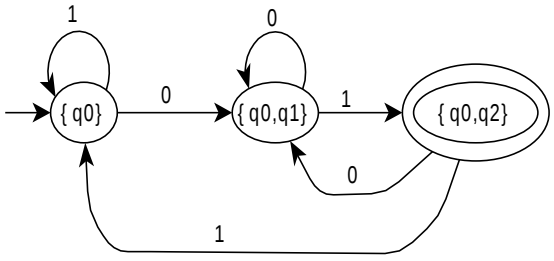
Iniciando no estado inicial, seguem-se todas as transições fazendo-se um estado do DFA com o conjunto de estados atingidos por cada par estado/símbolo. Exemplo:

NFA:



DFA:

	0	1
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
* $\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$



NFA com transições-ε

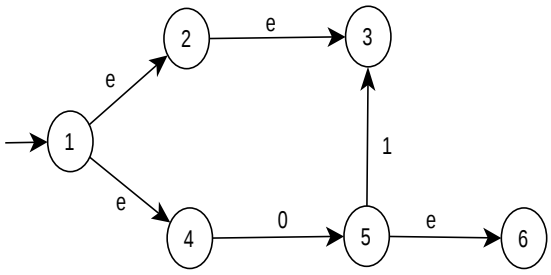
Notação formal: A = (Q, Σ, δ, q_0, F)

Como um NFA com excepção da função de transição δ que tem como parâmetros um estado e um símbolo de Σ ∪ {ε} devolvendo um conjunto de estados de Q: $\delta(q, a) = E$ em que q é o estado de

entrada, *a* o símbolo de entrada e *E* o conjunto de estados de saída, contido em Q.

Linguagem de um ε-NFA: a mesma que um NFA ou DFA (linguagens regulares).

Fechamento-ε (ε-closure): o fechamento-ε de um estado é o conjunto de estados que se atingem seguindo todas as transições-ε (espontânea) a partir dele. Notar que qualquer estado pode ser atingido por uma transição-ε a partir de si próprio. Exemplo:



ECLOSE(1) = { 1, 2, 3, 4 }; ECLOSE(2) = { 2, 3 }; ECLOSE(3) = { 3 }
ECLOSE(4) = { 4 }; ECLOSE(5) = { 5, 6 }; ECLOSE(6) = { 6 }

Converter ε-NFA em DFA: igual à conversão de NFA em DFA, excepto que em cada estado tem de se seguir as transições-ε e fechá-las (fechamento-ε).

Gramática Livre de Contexto (CFG)

G = (V, T, P, S)

V: conjunto de variáveis (normalmente letras maiúsculas)

T: conjunto de símbolos terminais

P: conjunto de produções

S: variável inicial

Exemplo (gramática de palíndromos binários):

$P \rightarrow \varepsilon$
$P \rightarrow 0$
$P \rightarrow 1$
$P \rightarrow 1P1$
$P \rightarrow 0P0$

G = ({ P }, { 0, 1 }, A, P), em que A é o conjunto das cinco produções acima.

Autómato de Pilha (PDA)

Autómato análogo aos ε-NFA, mas incluindo um pilha (stack).

Notação formal: P = (Q, Σ, Γ, δ, q_0, Z_0, F)

Q: conjunto finito de estados do autómato

Σ: alfabeto de entrada

Γ: alfabeto da pilha (conjunto finito de símbolos que podem aparecer na pilha)

δ: função de transição; aceita um estado de Q, um símbolo de $\Sigma \cup \{ \varepsilon \}$ e o símbolo de topo da pilha, devolvendo um conjunto de pares com o estado para o qual é feita a transição e a cadeia que substitui o símbolo no topo da pilha; esta cadeia pode ser ε (foi feito um POP à pilha), o mesmo símbolo que se encontrava no topo (não há alteração), ou qualquer outra cadeia de símbolos de Γ (foi feito um PUSH à pilha).

q_0: estado inicial

Z_0: símbolo inicial da pilha

F: conjunto de estados de Q que aceitam a linguagem do PDA (estados finais).

Diagrama de transição de um PDA: igual ao de um ε-NFA, com excepção que os arcos são nomeados na forma $a, x / \alpha$ em que:
a: símbolo de entrada (pode ser ε)
x: símbolo no topo da pilha
α: cadeia que substitui x no topo da pilha (pode ser ε ou x ou qualquer outra cadeia de símbolos de Γ)

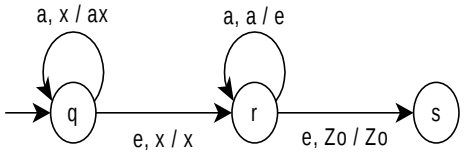
Normalmente assume-se que Z_0 é o símbolo inicial da pilha.

Exemplo: $\{ ww^R \mid w \in \{ 0, 1 \} \}$ (linguagem de palíndromos em {0,1} de tamanho par)

$P = (\{ q, r, s \}, \{ 0, 1 \}, \{ 0, 1, Z_0 \}, \delta, q, Z_0, \{ s \})$

1. $\delta (q, a, x) = \{ (q, ax) \}$ com $a \in \Sigma \wedge x \in \Gamma$
2. $\delta (q, \varepsilon, x) = \{ (r, x) \}$ com $x \in \Gamma$
3. $\delta (r, a, a) = \{ (r, \varepsilon) \}$ com $a \in \Sigma$
4. $\delta (r, \varepsilon, Z_0) = \{ (s, Z_0) \}$

A regra 1. permite ir lendo símbolos da entrada, colocando-os na pilha. A 2. permite a transição espontânea (em ε) para o estado r (adivinhando-se que se chegou ao meio da cadeia de entrada). A regra 3. permite ir lendo símbolos da entrada e consumindo símbolos iguais da pilha. Finalmente, a 4. faz a transição espontânea para o estado s quando se atinge o fim da pilha (e da cadeia de entrada)



Descrição instantânea de um PDA (ID): (q, w, γ) em que q é o estado corrente, w é a cadeia de entrada que falta ler (símbolo inicial à esquerda) e γ é o conteúdo da pilha (topo à esquerda).

Exemplo para o PDA acima com a cadeia de entrada 1111:

$(q, 1111, Z_0) \vdash (q, 111, 1Z_0) \vdash (q, 11, 11Z_0) \vdash (r, 11, 11Z_0) \vdash (r, 1, 1Z_0) \vdash (r, \varepsilon, Z_0) \vdash (s, \varepsilon, Z_0)$

Linguagem de um PDA (aceitação por estado final)

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$L_{(P)} = \{ w \mid (q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \alpha) \}$ em que $q \in F$ e α é uma qualquer cadeia da pilha.

Ou seja, começando no estado inicial e aplicando δ sucessivamente chegamos a um estado final tendo consumido toda a cadeia de entrada. Portanto, $L_{(P)}$ é o conjunto de cadeias que P consegue consumir terminando num estado final.

Linguagem de um PDA (aceitação por pilha vazia)

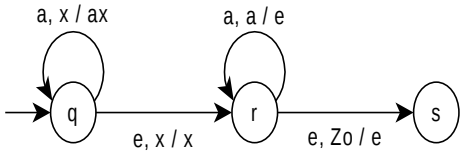
$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$N_{(P)} = \{ w \mid (q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon) \}$ em que $q \in Q$.

Ou seja, $N_{(P)}$ é o conjunto de cadeias que P consegue consumir e simultaneamente esvaziar a pilha. Uma vez que qualquer estado pode ser um estado final, pode-se omitir o seu conjunto na descrição formal do PDA:

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

Exemplo: o PDA acima poderia ser transformado num PDA por pilha vazia se consumisse o símbolo inicial da pilha na última transição:



As linguagens aceites pelos PDA, quer sejam de estado final quer de pilha vazia, são as mesmas – são as Linguagens Livres de Contexto.

Converter PDA pilha vazia para PDA estado final:

1. O alfabeto da pilha é o alfabeto da pilha do PDA de pilha vazia original com mais um símbolo, não presente no anterior, e que será o símbolo inicial da pilha do PDA de estado final.
2. Adicionar um novo estado inicial, com uma única transição-ε cuja finalidade é adicionar à pilha o símbolo inicial do PDA de pilha vazia original.
3. Adicionar transições-ε a partir de todos os estados para um novo estado final, ao encontrar no topo da pilha o símbolo inicial do PDA de pilha vazia. Esse símbolo marca efectivamente o esvaziar da pilha.

Ou seja, formalmente:

Sendo $P_N = (Q_N, \Sigma, \Gamma, \delta_N, q, Z_0)$, então $P_F = (Q_F, \Sigma, \Gamma \cup \{X_0,\}, \delta_F, p, X_0, \{r\})$

em que $Q_F = Q_N \cup \{p, r\}$ e

$\delta_F = \delta_N$; $\delta_F (p, \varepsilon, X_0) = \{ (q, Z_0 X_0) \}$; $\delta_F (q_x, \varepsilon, X_0) = \{ (r, \varepsilon) \}$ para todo o $q_x \in Q_N$

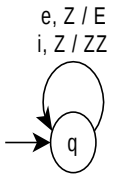
Exemplo:

PDA de pilha vazia que aceita a linguagem *if-else* quando há um *else* sem o *if* correspondente.

$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z)$

em que *i* significa que apareceu um *if* e *e* significa que apareceu um *else*.

$\delta_N(q, i, Z) = \{ (q, ZZ) \}$; $\delta_N(q, e, Z) = \{ (q, \varepsilon) \}$

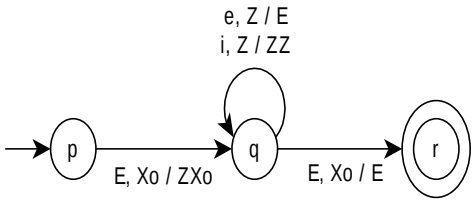


$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\})$

$\delta_F (p, \varepsilon, X_0) = \{ (q, ZX_0) \}$ → adicionar símbolo inicial de P_N à pilha

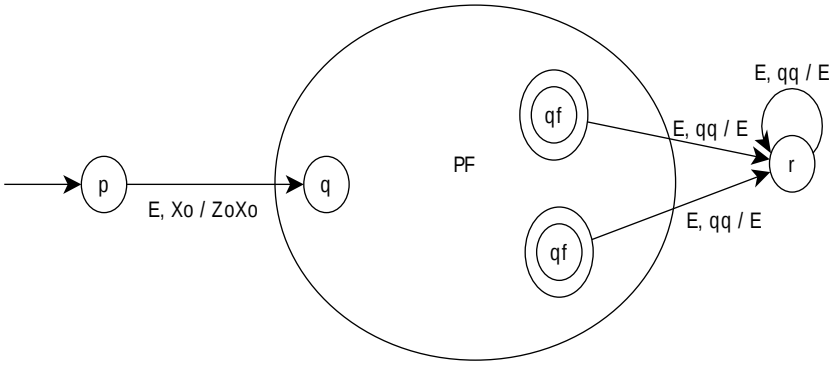
$\delta_F(q, i, Z) = \{ (q, ZZ) \}$; $\delta_F(q, e, Z) = \{ (q, \varepsilon) \}$ → simular P_N

$\delta_F (q, \varepsilon, X_0) = \{ (r, \varepsilon) \}$ → salto para o estado final ao encontrar o marcador de pilha vazia



Converter PDA estado final em PDA pilha vazia:

1. O alfabeto da pilha é o alfabeto da pilha do PDA de estado final original com mais um símbolo, não presente no anterior, e que será o símbolo inicial da pilha do PDA de pilha vazia.
2. Adicionar um novo estado inicial, com uma única transição-ε cuja finalidade é adicionar à pilha o símbolo inicial do PDA de estado final original.
3. Adicionar uma transição-ε de cada estado final do PDA original para um novo estado final: $\delta_N (qf, \varepsilon, qualquer) = \{ (r, \varepsilon) \}$
4. Adicionar uma transição-ε do novo estado final para si próprio em que consome toda a pilha: $\delta_N (r, \varepsilon, qualquer) = \{ (r, \varepsilon) \}$



Converter CFG em PDA de pilha vazia:

Tomando uma gramática livre de contexto G = (V, T, P, S) é possível convertê-la num PDA de pilha vazia desta forma:

- Construir um PDA com um único estado P = ({ q }, T, V ∪ T, δ, q, S) em que as transições são definidas por:
    1. δ (q, ε, A) = { (q, β) | A → β é uma produção de G }
    2. δ (q, a, a) = { (q, ε) } , para cada terminal (a)

Exemplo:

Sendo G = ({I, E}, { a, b, 0, 1, *, +, (,) }, A, E)

em que A:

I → a | b | Ia | Ib | I0 | I1

E → I | E*E | E+E | (E)

então P_G = ({q}, {a, b, 0, 1, *, +, (,)}, {a, b, 0, 1, *, +, (,), E, I}, δ, q, E)

em que:

δ (q, ε, I) = { (q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1) }

δ (q, ε, E) = { (q, I), (q, E*E), (q, E+E), (q, (E)) }

δ (q, a, a) = { (q, ε) }

δ (q, b, b) = { (q, ε) }

δ (q, 0, 0) = { (q, ε) }

δ (q, 1, 1) = { (q, ε) }

δ (q, *, *) = { (q, ε) }

δ (q, *, +) = { (q, ε) }

δ (q, (, () = { (q, ε) }

δ (q,),)) = { (q, ε) }

Converter PDA de pilha vazia em CFG:

AA

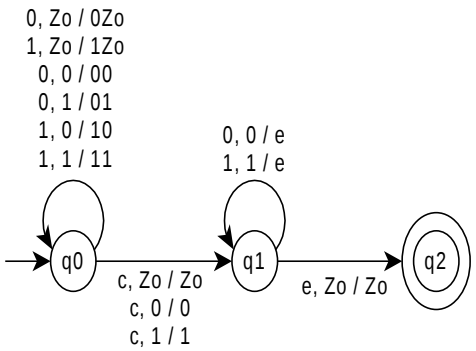
Autómatos de Pilha Deterministas (DPDA)

Um autómato de pilha P = (Q, Σ, Γ, δ, q_0, Z_0, F) é determinista se e só se:

1. δ (q, a, X) tem apenas um membro (resultado) para qualquer q em Q, a em Σ ou a = ε e X em Γ.
2. Se δ (q, a, X) não for o conjunto vazio para qualquer a em Σ, então δ (q, ε, X) não pode estar definido (tem de ser vazio).

Ou seja, um autómato de pilha determinista é um que não tem qualquer escolha de transição em qualquer triplo estado/símbolo de entrada/topo da pilha.

Exemplo: L = { wcw^R | $w \in \{0,1\}^*$ } : palíndromos binários com um marcador central



Linguagens de um DPDA: não são equivalentes às linguagens de um PDA, encontrando-se entre as linguagens regulares e as linguagens livres de contexto (ou seja, incluem as primeiras mas são apenas um subconjunto das segundas – há linguagens livres de contexto que não podem ser representadas por um DPDA). Para além disso, todas as linguagens livres de contexto que podem ser representadas por um DPDA são também representadas por uma gramática livre de contexto não ambígua (no entanto, nem todas as CFG não ambíguas podem ser representadas por um DPDA).

Máquinas de Turing

Uma máquina de Turing é composta por um controlo com um número finito de estados e uma *fita* composta por um número infinito de *células* cada qual contendo um símbolo. A máquina tem uma *cabeça* que está sempre posicionada sobre uma das células. Inicialmente, a palavra ou cadeia de entrada é colocada na fita, sendo o espaço restante à esquerda e à direita (infinito) preenchido com um símbolo especial (*blank*) significando *vazio*. Este símbolo pertence ao alfabeto dos símbolos da fita mas não ao dos símbolos da entrada – pode haver outros símbolos que pertençam à fita mas não à entrada. A posição inicial da cabeça é na célula com o símbolo de entrada mais à esquerda. Cada *movimento* é resultante do estado em que o controlo se encontra e do símbolo que a cabeça está a ler no momento e tem como efeito uma mudança de estado (que pode ser para o mesmo), a substituição do símbolo na fita (que pode ser o mesmo que lá estava) e o movimentar da cabeça para a célula à esquerda ou à direita.

Notação formal: M = (Q, Σ, Γ, δ, q_0, B, F) em que:

Q → número finito de estados da máquina

Σ → conjunto finito de símbolos de entrada

Γ → conjunto finito de símbolos da fita; contém Σ

δ → função de transição: $\delta\ (q, X) = (p, Y, D)$ em que $\{q, p\} \in$ Q, $\{X, Y\} \in \Gamma$ e D é a direcção do movimento da cabeça (direita – R – ou esquerda – L)

q_0 → estado inicial

B → símbolo que marca os espaços em branco (*blanks*)

F → conjunto finito de estados finais, subconjunto de Q

Descrição instantânea de Máquinas de Turing: wqx em que w é uma cadeia de símbolos da fita desde o primeiro símbolo não branco até ao símbolo anterior à célula em que a cabeça está

posicionada, q é o estado em que a máquina se encontra e x é uma cadeia de símbolos desde a célula em que a cabeça da máquina se encontra (inclusive) até ao símbolo mais à direita após o qual todos os restantes são brancos. Se w ou x forem de tamanho zero, o estado corrente será apresentado a uma ponta.

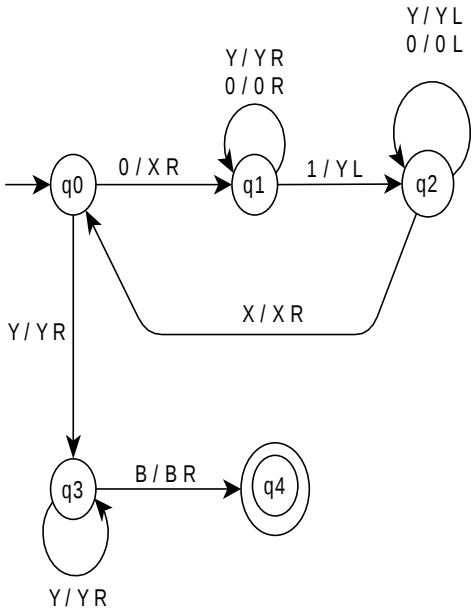
Exemplo: $\{ 0^n1^n \mid n \geq 1 \}$

M = ($\{q_0, q_1, q_2, q_3, q_4\}$, $\{0, 1\}$, $\{0, 1, X, Y, B\}$, δ, q_0, B, $\{q_4\}$)

Tabela de transição:

E \ S	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	(q_1, 0, R)	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	(q_2, 0, L)	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_2, Y, R)	(q_4, B, R)
q_4	-	-	-	-	

Diagrama de transição:



Sequência de transições usando descrições instantâneas para a entrada 0011:

$q_0 0011 \vdash X q_1 011 \vdash X0 q_1 11 \vdash X q_2 0Y1 \vdash q_2 X0Y \vdash 1\, X q_0 0Y1 \vdash XX q_1 Y1 \vdash XXY q_1 1 \vdash XX q_2 YY \vdash XX q_2 YY \vdash XX q_0 YY \vdash XXY q_3 Y \vdash XXYY q_3 B \vdash XXYYB q_4 B$

Linguagens de uma máquina de Turing: são normalmente chamadas *linguagens recursivamente enumeráveis* e podem definir-se da seguinte forma:

Se M = (Q, Σ, Γ, δ, q_0, B, F), então L(M) = $\{ w \mid w \in \Sigma^*,\ q_0 w \overset{*}{\vdash} \alpha p \beta$ para $p \in$ F e $\alpha, \beta \in \Gamma^* \}$

Formas de aceitação: Para além de aceitar entrando num estado final, também se considera por vezes que uma máquina de Turing aceita uma cadeia se entrar num estado em que *pare*, ou seja, em que não tenha movimento definido para o par estado/entrada.