Teoria da Computação

Última atualização: 2/2/2009

1 Autômatos: Introdução e Conceitos Básicos

A teoria de autômatos é o estudo de computadores abstratos, também chamados de "máquinas". Em 1930, antes de existirem computadores, A. Turing desenvolveu uma máquina abstrata que tinha todas as características dos computadores atuais, ao menos no que se refere ao quanto eles poderiam calcular. O objetivo de Turing era descrever com exatidão o que uma máquina de computação seria e o que ela não seria capaz de fazer. As conclusões de Turing se aplicam não apenas a sua máquina, mas também as máquinas reais de hoje.

Na década de 1940 e 1950, tipos de máquinas simples, que hoje são chamados de "autômatos finitos", foram estudados por diversos pesquisadores. No final dos anos 50, o lingüista N. Chomsky iniciou o estudo de "gramáticas" formais. Essas gramáticas têm um relacionamento estreito com os autômatos abstratos e hoje servem como base de importantes componentes de software, incluindo parte dos compiladores.

Em 1969, S. Cook estendeu o estudo de Turing do que podia e não podia ser calculado. Ele conseguiu separar os problemas que podem ser resolvidos de forma eficiente por computadores daqueles que poderiam em princípio ser resolvidos, mas que na prática levam tanto tempo para serem solucionados que os computadores são inúteis para solucioná-los. Tais problemas são chamados "intratáveis" ou "NP-difíceis" (NP-Hard). Mesmo com a melhoria exponencial na velocidade dos computadores (Lei de Moore) é muito pouco provável que haja impacto significativo na habilidade de resolver grandes instâncias de problemas intratáveis.

Todos esses desenvolvimentos teóricos têm ligação direta com o que os cientistas da computação fazem hoje. Por exemplo: a teoria dos problemas intratáveis nos permite deduzir se ao encontrar um problema temos chance de conseguir construir um programa para resolvê-lo (porque ele não é intratável) ou se teremos de descobrir algum modo de contornar o problema intratável.

1.1 Linguagem

Linguagem é um conceito fundamental no estudo de Teoria da Computação, pois trata da forma precisa de expressar problemas, permitindo um desenvolvimento formal adequado ao estudo da computabilidade. A seguir são mostradas algumas definições:

1.1.1 Alfabeto

Um alfabeto é um conjunto finito de *símbolos* ou *caracteres*.

- Um conjunto infinito não é um alfabeto
- O conjunto vazio é um alfabeto

Exemplos de alfabetos:

{a, b, c}

Ø (conjunto vazio)

Não são alfabetos:

Conjunto dos números naturais, inteiros, etc...

{a, b, aa, ab, ba, bb, aaa, ... }

1.1.2 Cadeia de símbolos, Palavra (String)

Uma *Cadeia de Símbolos* sobre um conjunto é uma seqüência de zero ou mais símbolos (do conjunto) justapostos.

Uma *Palavra* é uma cadeia de símbolos finita.

Uma cadeia sem símbolos é uma palavra válida e o símbolo:

ε denota uma *cadeia vazia* ou *palavra vazia*

Se Σ representa um alfabeto então

Σ^* denota o conjunto de todas as palavras possíveis sobre Σ

Σ^+ denota $\Sigma^* - \{\varepsilon\}$

Exemplo:

a) abcb é uma palavra do alfabeto {a, b, c}

b) Se Σ = {a, b} então:

Σ^+ = {a, b, aa, ab, ba, bb, aaa, ... }

Σ^* = {ε, a, b, aa, ab, ba, bb, aaa, ... }

1.1.3 Comprimento ou Tamanho de uma Palavra

O *Comprimento* ou *Tamanho* de uma palavra w, representado por |w|, é o número de símbolos que compõe a palavra.

Exemplos: |abcb| = 4 e |ε| = 0

1.1.4 Prefixo, Sufixo, Subpalavra

Um *Prefixo* de uma palavra é qualquer seqüência inicial de símbolos da palavra.

Um *Sufixo* de uma palavra é qualquer seqüência final de símbolos da palavra.

Uma *Subpalavra* de uma palavra é qualquer seqüência que compõe a palavra.

Exemplo: Seja a palavra abcb

ε, a, ab, abc, abcb são os prefixos;

ε, b, cb, bcb, abcb são os sufixos;

1.1.5 Linguagem Formal

Uma *Linguagem Formal* ou simplesmente *Linguagem* é um conjunto de palavras sobre um alfabeto.

Exemplo: Suponha o alfabeto Σ = {a, b} então:

a) O conjunto vazio e o conjunto formado pela palavra vazia são linguagens sobre Σ. Lembrando que $\emptyset \neq \{ \varepsilon \}$

b) O conjunto de palíndromos (palavras que tem a mesma leitura da esquerda para a direita e vice-versa) sobre Σ é um conjunto de linguagem infinita. Assim: ε, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, ... são palavras desta linguagem.

É comum definir linguagens usando um "formador de conjuntos":

{ *w* | algo sobre *w* }

Essa expressão é lida como "o conjunto de palavras w tais que (seja o que for dito sobre w à direita da barra vertical)". Alguns exemplos:

- {*w* | *w* consiste em um número igual de 0's e 1's}
- {*w* | *w* é um número inteiro binário primo}
- {*w* | *w* é um programa em C sintaticamente correto}

Também é comum substituir w por alguma expressão com parâmetros e descrever os strings na linguagem declarando condições sobre os parâmetros. Exemplos:

- $\{0^n1^n \mid n \geq 1\}$. Lê-se "o conjunto de 0 elevado a *n* 1 elevado a *n* tal que *n* maior ou igual a 1"; essa linguagem consiste nos strings {01, 0011, 000111, ...}. Note que, como ocorre com os alfabetos, podemos elevar

um único símbolo a uma potência *n* para representar *n* cópias desse símbolo.

- $\{0^i1^j \mid 0 \leq i \leq j\}$. Essa linguagem consiste em strings com alguns 0's (possivelmente nenhum) seguidos por um número igual ou superior de 1's.

1.1.6 Concatenação de Palavras

A concatenação de palavras é uma operação binária, definida sobre uma linguagem, a qual associa a cada par de palavras uma palavra formada pela justaposição da primeira com a segunda.

Suponha v,w,t como palavras:

v(wt) = (vw)t - associatividade

εw = w = wε - elemento neutro a esquerda ou direita

Exemplos: Suponha o alfabeto Σ = {a, b} e as palavras v = baa e w = bb, tem-se que:

vw = baabb

vε = v = baa

Uma concatenação definida sobre uma linguagem L não é necessariamente fechada sobre L, ou seja, a concatenação de duas palavras de L não é necessariamente uma palavra de L.

Exemplo: Suponha a linguagem L de palíndromos sobre Σ = {a, b}. A concatenação das palavras aba e bbb resultam na palavra ababbb a qual não é palíndromo. Portanto a operação de concatenação não é fechada sobre L.

1.1.7 Concatenação Sucessiva de Palavras

A concatenação sucessiva de uma Palavra (com ela mesma) é representada na forma de um expoente:

w^n onde n é o número de concatenações sucessivas

Exemplos:

$w^n = \varepsilon$

$w^1 = w$

$w^3 = www$

$w^5 = wwwww$

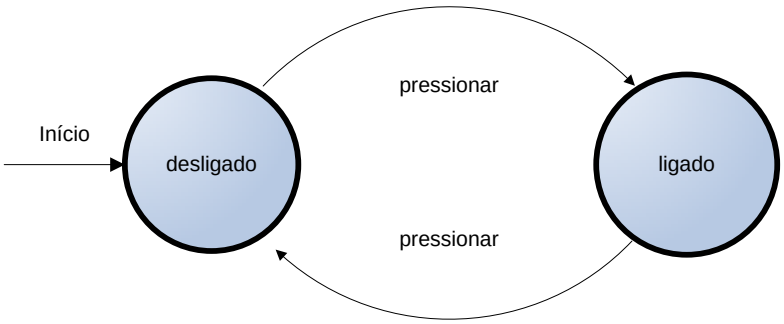
2 Autômatos Finitos

Os autômatos finitos constituem um modelo útil para muitos elementos importantes de hardware e software. Alguns exemplos:

- Software para projetar e verificar comportamento de circuitos digitais
- Analisador Léxico de um compilador típico (isto é, componente que divide o texto de entrada em unidades lógicas, como identificadores, palavras-chave, etc.)
- Software para examinar grandes corpos de texto, como páginas Web
- Software para verificar sistemas de todos os tipos que tem um número finito de estados distintos, como protocolos de comunicação ou segurança

Estes elementos têm como características estarem a todo o momento em um determinado "estado" de um conjunto finito deles. Como o conjunto é finito a história toda de execução não pode ser memorizada, assim o sistema deve ser projetado a fim de memorizar apenas o que é relevante. A vantagem de usar um número finito de estados é que é possível implementá-lo de uma forma simples em hardware como um circuito, ou em um software que possa tomar decisões examinando apenas um número limitado de dados ou a própria posição no código.

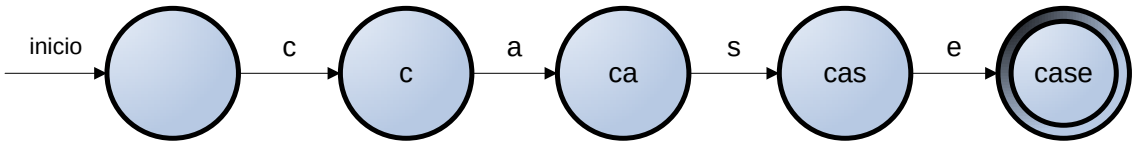
Exemplo de autômato finito: um interruptor que memoriza se está no estado "ligado" ou "desligado", e permite que o usuário pressione um único botão cujo efeito será diferente de acordo com o estado em que o interruptor se encontra, ou seja, se ele estiver desligado e for pressionado ele irá ligar, e vice-versa.



Os estados estão representados por círculos, e a ação (ou "entrada") está representada pelos arcos. Neste caso temos os estados ligado e desligado, e ambos os arcos representam a ação de pressionar o botão. O estado inicial é indicado pela palavra "início" e por uma seta que leva a este estado. Frequentemente também precisamos

indicar um ou mais estados “finais” ou de “aceitação”, que representam que a entrada é “válida”, neste caso utilizamos círculos duplos para representar tais estados.

Outro exemplo: um autômato que reconhece a palavra chave *case*, ele tem cinco estados que representam desde a string vazia até a palavra completa, passando por todos os seus prefixos.



O único estado de aceitação é aquele em que a palavra aparece completa. Podemos imaginar este autômato como parte de um analisador léxico, que estará analisando um a um os caracteres do programa.

Na teoria dos autômatos, um problema é a questão de verificar se uma dada string (palavra) é elemento de uma linguagem específica. Assim se Σ é um alfabeto e L é uma linguagem sobre Σ, então dado um string w em Σ^*, definir se w está ou não em L.

2.1 Autômatos Finitos Determinísticos

Um *autômato finito determinístico* (AFD, ou DFA em inglês) é um autômato que se encontra sempre em um único estado após ler uma seqüência qualquer de entrada. O termo “determinístico” implica que existe um e somente um estado ao qual o autômato pode transitar a partir de seu estado atual. Em contraste, autômatos finitos não-determinísticos podem estar em vários estados ao mesmo tempo.

2.1.1 Definição Formal

Um autômato finito determinístico consiste em:

a) Um conjunto finito de *estados*, frequentemente denotado por *Q*.

b) Um conjunto finito de *símbolos de entrada*, frequentemente denotado por Σ

c) Uma *função de transição* que toma como argumentos um estado e um símbolo de entrada e retorna um estado. A função de transição será comumente denotada por δ. No grafo é representada pelos arcos e rótulos entre os estados. Se *q* é um estado, e *a* é um símbolo de entrada, então $\delta(q,a)$ é o estado *p* tal que existe um arco identificado por *a* de *q* até *p*.

d) Um *estado inicial*, que é um dos estados em *Q*

e) Um conjunto de estados *finais* ou de *aceitação F*. O conjunto *F* é um subconjunto de *Q*.

Frequentemente um AFD é denotado como uma "tupla de cinco elementos", como se segue:

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

Onde *A* é o nome do AFD, *Q* é seu conjunto de estados, Σ é seu conjunto de símbolos de entrada, δ sua função de transição, q_0 é seu estado inicial e *F* é seu conjunto de estados de aceitação.

2.1.2 AFD processando strings

A "linguagem" de um AFD é o conjunto de todos os strings que ele aceita. Suponha que a_1, a_2, a_3, ... a_n seja uma seqüência de símbolos de entrada. Começamos com o AFD em seu estado inicial q_0 e procuramos por uma função de transição, algo como $\delta(q_0, a_1) = q_1$, para saber em qual estado o AFD se encontrará após processar a_1. Em seguida processaremos a_2, avaliando $\delta(q_1, a_2)$ e assim sucessivamente. Se a função procurada não for encontrada é um sinal de que a string de entrada não faz parte da linguagem definida pelo autômato e deve ser rejeitada.

Exemplo: Especificar formalmente um DFA que aceita todos e somente os strings de 0's e 1's que têm a seqüência 01 em algum lugar na string. Podemos escrever essa linguagem como:

{*x*01*y* | *x* e *y* são quaisquer strings de 0's e 1's }

Alguns exemplos de strings presentes na linguagem são 01, 1010, 01010 e 100010. Exemplos de strings que não estão na linguagem são: 0, 111000 e ε.

Sabemos então que o alfabeto de entrada é $\Sigma = \{0,1\}$, que existe um conjunto de estados *Q*, e que há um estado inicial que chamaremos de q_0. Para decidir se 01 é uma substring de entrada, o autômato precisa se lembrar:

1. Ele já viu 01? Nesse caso ele aceita toda seqüência de entrada adicional; ou seja, ele estará sempre em estado de aceitação de agora em diante
2. Ele nunca viu 01, mas sua entrada mais recente foi 0; assim se agora ele vir o 1, terá visto 01 e poderá aceitar tudo que vir por diante
3. Ele nunca viu 01, mas sua entrada mais recente foi 1 ou não existente (ele apenas iniciou). Nesse caso, ele não pode aceitar até ver um 0 seguido de 1.

Cada uma dessas condições pode ser representada por um estado. A condição 3 é representada pelo estado inicial q_0, pois quando iniciamos ainda esperamos por 01. Se estivermos em q_0 e recebermos um 1, então devemos permanecer no estado q_0. Isto é $\delta(q_0, 1) = q_0$.

Entretanto, se estamos em q_0 e vemos um 0, estaremos na condição 2, isto é, não vimos 01, mas temos um 0. Assim vamos usar q_2 para representar esta condição. A transição é, portanto $\delta(q_0, 0) = q_2$.

Agora, estando em q_2, podemos ver um 0, neste caso estaremos na mesma situação que antes, ou seja, continuamos com um 0 e esperando um 1. Portanto devemos ficar no mesmo estado $\delta(q_2, 0) = q_2$. Porém se estamos em q_2 e vemos uma entrada 1, sabemos agora que existe um 0 seguido de 1. Podemos então passar para um estado de aceitação que chamaremos de q_1 e que corresponde a condição 1. Assim: $\delta(q_2, 1) = q_1$.

Finalmente, estando no estado q_1 qualquer que seja a entrada, ainda estaremos na condição 1, em que já vimos um 01. Assim permaneceremos neste estado, $\delta(q_1, 1) = \delta(q_1, 0) = q_1$.

Portanto, $Q = \{q_0, q_1, q_2\}$, q_0 é o estado inicial como dito anteriormente e q_1 é o único estado de aceitação portanto $F = \{q_1\}$. Assim a especificação completa do autômato pode ser dada por:

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

onde δ é a função de transição descrita anteriormente.

2.1.3 Diagrama de Transições

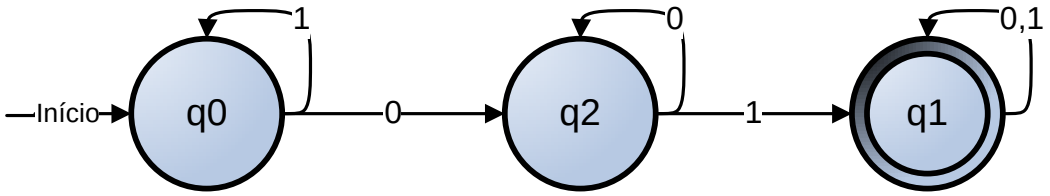
Especificar um AFD através da tupla de cinco e das funções de transição é algo tedioso e que fica difícil de ler, existem outros tipos de notações que são preferenciais, dentre elas podemos destacar o *diagrama de transições*, que é um grafo como os vistos anteriormente.

Um diagrama de transições para um AFD $A = (Q,\Sigma,\delta,q_0,F)$ é um grafo definido da seguinte forma:

1. Para cada estado em Q existe um nó correspondente.
2. Para cada estado q em Q e para cada símbolo de entrada a em Σ, seja $\delta(q,a) = p$. Então, o diagrama de transições tem um arco do nó q para o nó p, rotulado por a. Se existem vários símbolos de entrada que causam transições de q para p, então o diagrama de transições pode ter um arco rotulado pela lista desses símbolos.

3. Existe uma seta no estado inicial q_0, identificada como *Ínicio*. Essa seta não se origina em nenhum nó.
4. Os nós correspondentes aos estados de aceitação (aqueles em *F*) são marcados por um círculo duplo. Todos os outros estados tem um único círculo.

Exemplo: abaixo vemos o diagrama de transições para o AFD que projetamos na seção anterior, e que aceita todos os strings que contem o substring 01:



2.1.4 Tabela de Transições

Outra maneira mais simples de especificar um AFD é a *tabela de transições* que é uma representação convencional e tabular de uma função como δ que recebe dois argumentos e retorna um valor. As linhas da tabela correspondem aos estados, e as colunas correspondem as entradas. O conjunto de estados e o alfabeto de entrada são especificados implicitamente. O estado de entrada é indicado com uma seta, e os estados de aceitação são marcados com asterisco.

Exemplo: abaixo vemos a tabela de transição para o nosso mesmo exemplo anterior que aceita todos os strings que contem o substring 01:

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

2.1.5 Função de Transição Estendida

Função de Transição Estendida é uma função que toma um estado *q* e um string *w* e retorna um estado *p* – o estado que o autômato alcança quando começa no estado *q* e processa a seqüência de entradas *w*. A função de transição estendida é normalmente denotada por $\underline{\delta}$, δ* ou ainda δ com acento circunflexo.

Exemplos: funções de transição estendida para cada prefixo da string 11010 com o autômato do nosso exemplo que aceita strings que contem 01 como substring:

$\underline{\delta}(q_0,11010) = \underline{\delta}(\delta(q_0,1),1010) =$

$\underline{\delta}(q_0,1010) = \underline{\delta}(\delta(q_0,1),010) =$

$\underline{\delta}(q_0,010) = \underline{\delta}(\delta(q_0,0),10) =$

$\underline{\delta}(q_2,10) = \underline{\delta}(\delta(q_2,1),0) =$

$\delta(q_1,0) = q_1$

2.1.6 Linguagem de um AFD

A linguagem de um AFD $A = (Q,\Sigma,\delta,q_0,F)$, denotada por $L(A)$ é definida por:

$$L(A) = \{w \mid \underline{\delta}(q_0,w) \text{ está em } F\}$$

ou seja, dado um string *w*, se construirmos sua função de transição estendida $\underline{\delta}$ e chegarmos a um estado que está em *F* (o conjunto de estados finais), então *w* está em A (é aceito pelo autômato A). Se *L* é *L*(*A*) para algum AFD *A*, dizemos que *L* é uma *linguagem regular*.

2.2 Autômatos Finitos Não-Determinísticos

Um autômato finito "não-determinístico" (AFND, ou NFA do inglês) tem o poder de estar em vários estados ao mesmo tempo. Essa habilidade é expressa com freqüência como a capacidade de "adivinhar" algo sobre sua entrada. Por exemplo, quando o autômato é usado para procurar certas seqüências de caracteres (como por exemplo, palavras-chave) em um longo string de texto, é útil "adivinhar" que estamos no início e um desses strings e usar uma seqüência de estados apenas para verificar se o string aparece, caractere por caractere.

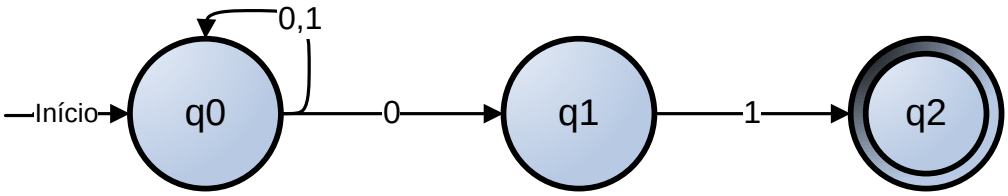
Os AFND aceitam exatamente as linguagens regulares, da mesma maneira que fazem os AFDs. Desta forma sempre é possível converter um AFND em um AFD, porém o AFD gerado pode ter exponencialmente mais estados que o AFND. Muitas vezes os AFNDs são mais sucintos e mais fáceis de projetar que um AFD.

A diferença entre um AFD e um AFND está na função de transição δ. Para um AFND esta função recebe um estado e um símbolo de entrada como argumentos (da mesma forma que um AFD), porém retorna um conjunto de zero, um ou mais estados (em vez de retornar exatamente um estado como um AFD deve fazer).

Exemplo: abaixo temos um autômato que aceita todos os strings de 0's e 1's que terminam em 01 e somente eles. O estado q_0 é o estado inicial, e podemos pensar que o autômato está nele até que "adivinhe" que o 01 final começou, É sempre possível que o

próximo símbolo não inicie o 01 final, mesmo que esse símbolo seja 0. Desse modo, o estado q_0 pode fazer uma transição para ele mesmo em 0 e em 1.

Porém, se o próximo símbolo é 0, esse AFND também adivinha que o 01 final começou. Um arco identificado por 0 leva portanto de q_0 a q_1. Note que existem dois arcos rotulados como 0 saindo de q_0. O AFND tem a opção de ir para q_0 ou q_1 e, de fato, ele segue os dois caminhos. No estado q_1 o AFND verifica se o próximo símbolo é 1 e, nesse caso, vai para o estado q_2 e aceita a entrada.



Observe que não existe nenhum arco saindo de q_1 rotulado com 0, e não existe nenhum arco saindo de q_2. Nessas situações, o encadeamento no AFND correspondente a esses a estes estados simplesmente "morre", embora outros encadeamentos possam continuar a existir.

2.2.1 Definição Formal

Um AFND consiste em:

a) Um conjunto finito de *estados*, frequentemente denotado por *Q*.

b) Um conjunto finito de *símbolos de entrada*, frequentemente denotado por Σ

c) Uma *função de transição* δ que toma como argumentos um estado e um símbolo de entrada e retorna um subconjunto de *Q*. A diferença para os AFD está no tipo de valor que δ retorna: um conjunto de estados para os AFND e um único estado para os AFD.

d) Um *estado inicial*, q_0 que é um dos estados em *Q*

e) Um conjunto de estados *finais* ou de *aceitação F*, sendo que o conjunto *F* é um subconjunto de *Q*.

A tupla de 5 elementos $A = (Q,\Sigma,\delta,q_0,F)$ utilizada nos AFD também é utilizada para definir os AFNDs, assim o nosso exemplo pode ser definido por:

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

onde a função de transição é dada pela tabela de transições abaixo:

	0	1
$\rightarrow q_0$	$\{q_0,q_1\}$	$\{q_0\}$

q_1	Ø	$\{q_2\}$
$*q_2$	Ø	Ø

Note que as tabelas de transições utilizadas nos AFD também servem para os AFND, sendo que a única diferença é que as respostas são dadas por conjunto de estados, em vez de um estado unitário, assim é necessário o uso de chaves mesmo que haja um único elemento. Quando não existe nenhuma transição de um estado dada uma entrada a resposta é conjunto Ø (vazio).

2.2.2 Função de Transição Estendida

Suponha a string de entrada 00101 para o autômato do nosso exemplo, teremos a seguinte função de transição estendida:

$\underline{\delta}(q_0,00101) = \underline{\delta}(\delta(q_0,0),0101) =$

$\underline{\delta}(\{q_0,q_1\},0101) = \underline{\delta}(\delta(q_0,0) \cup \delta(q_1,0),101) = \underline{\delta}(\{q_0,q_1\} \cup Ø,101)$

$\underline{\delta}(\{q_0,q_1\},101) = \underline{\delta}(\delta(q_0,1) \cup \delta(q_1,1),01) = \underline{\delta}(\{q_0\} \cup \{q_2\},101)$

$\underline{\delta}(\{q_0,q2\},01) = \underline{\delta}(\delta(q_0,0) \cup \delta(q_2,0),1) = \underline{\delta}(\{q_0,q_1\} \cup Ø,01)$

$\underline{\delta}(\{q_0,q_1\},1) = \delta(q_0,1) \cup \delta(q_1,1) = \{q_0\} \cup \{q_2\} = \{q_0,q_2\}$

2.2.3 Linguagem de um AFND

Um AFND aceita um determinado string *w* se é possível chegar a um estado de aceitação por pelo menos um dos caminhos possíveis. O fato de um dado caminho não chegar a um estado de aceitação ou "morrer" não impede que um AFND seja aceito como um todo. Formalmente, se $A = (Q,\Sigma,\delta,q_0,F)$ é um AFND então

$$L(A) = \{w \mid \underline{\delta}(q_0,w) \cap F \neq Ø\}$$

ou seja, $L(A)$ é o conjunto de strings *w* que submetidas a função de transição estendida $\underline{\delta}$ retornarão um conjunto que contenha pelo menos um estado de aceitação.

2.2.4 Equivalência entre autômatos finitos determinísticos e não determinísticos

Para muitas linguagens é mais fácil construir um AFND do que um AFD, entretanto sempre é possível construir um AFD que aceite a mesma linguagem de um AFND. Na prática um AFD tem quase tantos estados quantos tem o AFND correspondente, embora com freqüência tenha mais transições. Porém, no pior caso, o menor AFD pode ter 2^n estados para um AFND para a mesma linguagem que tenha *n* estados.

Podemos construir um AFD a partir de um AFND apenas construindo todos os subconjuntos de estados possíveis do AFND, sem conhecer os detalhes específicos deste último.

A construção de subconjuntos começa a partir de um AFND $N = (Q_N,\Sigma,\delta_N,q_0,F_N)$. O objetivo é descrever um AFD $D = (Q_D,\Sigma,\delta_D,q_0,F_D)$, tal que $L(D) = L(N)$. Note que os alfabetos de entrada dos dois autômatos são os mesmos, e o estado inicial de D é o mesmo de N. Os outros componentes são construídos da seguinte forma:

- Q_D é o conjunto de subconjuntos de Q_N; isto é, Q_D é o conjunto potência de Q_N. Note que, se Q_N tem n estados, então Q_D terá 2^n estados. Com freqüência, nem todos esses estados estão acessíveis a partir do estado inicial de Q_D. Os estados inacessíveis podem ser "descartados"; assim, o número de estados de D pode ser efetivamente muito menor que 2^n.
- F_D é o conjunto de subconjuntos S de Q_N tais que $S \cap F_N \neq \emptyset$. Isto é, F_D representa todos os conjuntos de estados de N que incluem pelo menos um estado de aceitação N.
- Para cada conjunto $S \subseteq Q_N$ e para cada símbolo de entrada a em Σ,

$$\delta_D(S,a) = \bigcup_{p \text{ em } s} \delta_N(p,a)$$

Isto é, para calcular $\delta_D(S,a)$, examinamos todos os estados p em S, vemos para quais estados N vai a partir de p sobre a entrada a e fazemos a união de todos esses estados.

Exemplo: seja N o autômato do nosso exemplo de autômato que aceita strings terminadas em 01. O conjunto de estados de N é $\{q_0, q_1, q_2\}$, a construção de subconjuntos produz um AFD com $2^3 = 8$ estados, correspondendo a todos os subconjuntos deste três estados. A seguir vemos a tabela de transição para estes 8 estados:

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

*{q_1, q_2}	Ø	{q_2}
*{ q_0, q_1, q_2}	{q_0, q_1}	{q_0, q_2}

Esta tabela de transições equivale a um autômato finito determinístico, pois embora as entradas na tabela sejam conjuntos, os estados do AFD gerado também são conjuntos. Para tornar este ponto mais claro podemos criar novos nomes para esses estados; por exemplo, *A* para Ø, B para {q_0} e assim por diante.

	0	1
A	*A*	*A*
→*B*	*E*	*B*
C	*A*	*D*
**D*	*A*	*A*
E	*E*	*F*
**F*	*E*	*B*
**G*	*A*	*D*
**H*	*E*	*F*

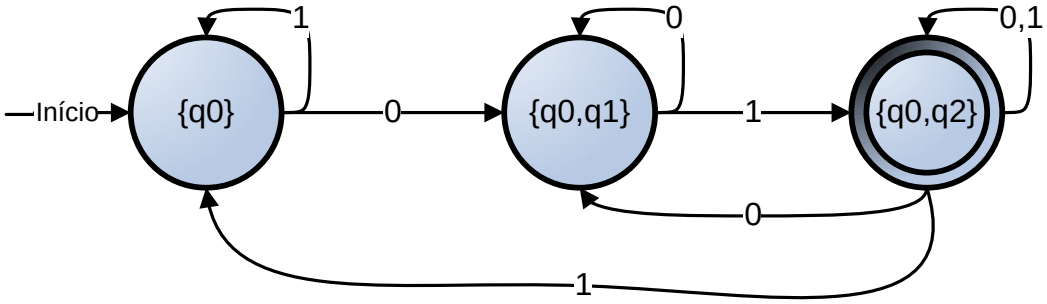
Dos oito estados, começando no estado inicial B, só podemos acessar os estados *B, E* e *F*. Os outros cinco estados são inacessíveis a partir do estado inicial e não precisam constar da tabela. É possível evitar esta etapa de tempo exponencial na construção de entradas da tabela através de uma avaliação de subconjuntos "ociosos".

Uma maneira de fazer isso é partindo do subconjunto do estado inicial {q_0} que sabemos que faz parte do AFD, descobrimos que $\delta_D(\{q_0\},0) = \{q_0,q_1\}$ e $\delta_D(\{q_0\},1) = \{q_0\}$, estes dados são obtidos do diagrama de transições do AFND. O conjunto {q_0} é antigo e já foi avaliado, porém o outro {q_0,q_1} é novo e suas transições devem ser calculadas. Encontramos $\delta_D(\{q_0,q_1\},0) = \{q_0,q_1\}$ e $\delta_D(\{q_0,q_1\},1) = \{q_0,q_2\}$. E iremos considerar todos os novos conjuntos que forem aparecendo até que nenhum novo conjunto apareça, e aí a construção terá convergido. Assim teremos a seguinte tabela de transição:

	0	1
→{q_0}	{q_0, q_1}	{q_0}
{q_0, q_1}	{q_0, q_1}	{q_0, q_2}

*$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

e o seguinte diagrama de transições correspondente:



Observe que ele só tem três estados, que por coincidência é o mesmo número de estados que AFND a partir do qual ele foi construído. Porém ele tem seis transições comparado com as quatro transições do AFND.

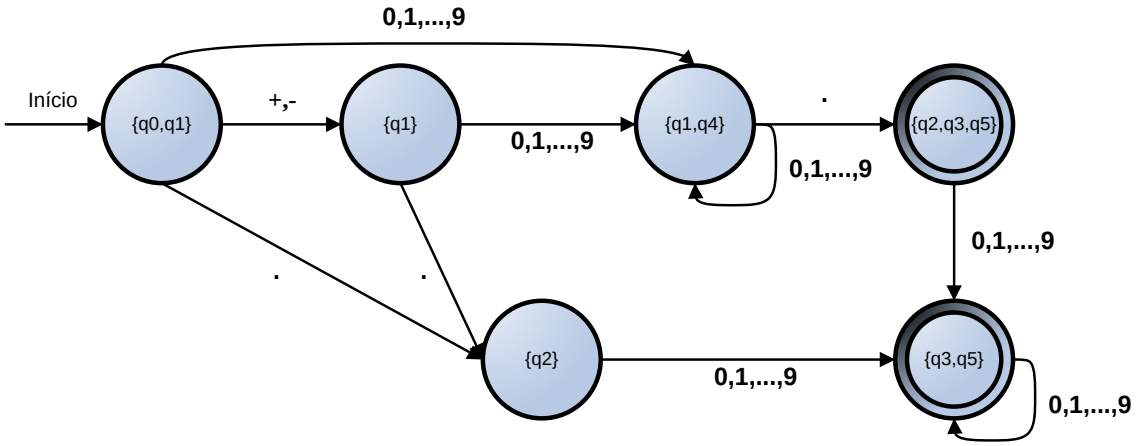
2.3 Autômatos Finitos com Transições Vazias

Os Autômatos Finitos com Transições Vazias (AFNDε, AFNε ou ainda ε-NFA) são AFNDs que aceitam transições vazias ou seja, transições sem leitura de símbolo algum da fita, o que também pode ser interpretado como a transição sobre ε, o string vazio. Esse recurso não expande a classe de linguagens que pode ser aceita pelo autômato finito, mas nos dá uma certa facilidade adicional para projetar. Assim para qualquer autômato finito não determinístico com transições vazias pode ser construído um autômato finito determinístico (AFD) equivalente.

Ao montar o diagrama de transição utilizamos o símbolo ε para representar as transições vazias. Abaixo mostramos um exemplo de AFNDε que aceita números decimais que consistem em:

1. Um sinal + ou – opcional
2. Uma string de dígitos
3. Um ponto decimal
4. Outra string de dígitos.

Tanto o string (2) quanto o string (4) podem ser vazios, mas pelo menos um dos dois deve ser não-vazio.



O estado q_1 representa a situação em que já vimos o sinal, se ele existir, e talvez alguns dígitos, mas não o ponto decimal. O estado q_2 representa a situação em que acabamos de ver o ponto decimal, tendo visto ou não dígitos anteriores. Em q_4 vimos pelo menos um dígito, mas não o ponto decimal. Assim em q_3 vimos um ponto decimal e pelo menos um dígito, esteja ele antes ou depois do ponto decimal. Podemos permanecer em q_3 lendo outros dígitos que existirem, e também podemos a qualquer momento "adivinhar" que a string de dígitos está completa e ir espontaneamente para q_5, o estado de aceitação.

2.3.1 Notação Formal

O AFNDε pode ser representado da mesma forma que um AFND, porém acrescentando informações a respeito das transições sobre ε. Um AFNDε A é representado formalmente por $A = (Q,\Sigma,\delta,q_0,F)$, onde cada componente tem a mesma interpretação que no caso de um AFND, exceto pela função de transição δ ser agora uma função que recebe como argumentos:

1. Um estado em Q
2. Um elemento de Σ U {ε}, isto é, um símbolo de entrada do alfabeto ou o símbolo ε (que representa a string vazia e não pode fazer parte do alfabeto)

O AFND do nosso exemplo é definido formalmente como

$$E = (\{q_0,q_1,q_2,q_3,q_4,q_5\},\{.,+,-,0,1,2,3,4,5,6,7,8,9\},\delta,q_0,\{q_5\})$$

onde δ é definido pela tabela de transições abaixo:

	ε	+,-	.	0,1,...,9
→q_0	$\{q_1\}$	$\{q_1\}$	Ø	Ø
q_1	Ø	Ø	$\{q_2\}$	$\{q_1, q_4\}$
q_2	Ø	Ø	Ø	$\{q_3\}$

q_3	$\{q_5\}$	Ø	Ø	$\{q_3\}$
q_4	Ø	Ø	$\{q_3\}$	Ø
q_5	Ø	Ø	Ø	Ø

2.3.2 Fecho-ε

A função Fecho-ε (também conhecida como Fecho-Vazio e ECLOSE) de um estado pode ser definida informalmente como a função que recebe um estado q e retorna o conjunto de estados composto por q e todos os estados em que é possível chegar a partir de q seguindo transições rotuladas por ε. Formalmente podemos definir Fecho-ε(q) recursivamente:

- O estado q está em Fecho-ε(q)
- Se o estado p está em Fecho-ε(q), e existe uma transição do estado p para o estado r rotulada por ε, então r está em Fecho-ε(q)

Para o nosso exemplo temos as seguintes funções Fecho-ε:

Fecho-ε(q_0) = $\{q_0, q_1\}$ Fecho-ε(q_1) = $\{q_1\}$

Fecho-ε(q_2) = $\{q_2\}$ Fecho-ε(q_3) = $\{q_3, q_5\}$

Fecho-ε(q_4) = $\{q_4\}$ Fecho-ε(q_5) = $\{q_5\}$

2.3.3 Função de Transição Estendida

Suponha a entrada **8.5** para o AFNDε do nosso exemplo, faremos a função de transição estendida da seguinte forma:

1. Primeiro calculamos o Fecho-ε do nosso estado inicial:
    - Fecho-ε(q_0) = $\{q_0,q_1\}$
2. Dado o símbolo **8** teremos que calcular sua transição a partir dos estados do Fecho-ε do nosso estado inicial, ou seja, q_0 e q_1:
    - δ(q_0,8) U δ(q_1,8) = Ø U $\{q_1, q_4\}$
3. Faremos agora o Fecho-ε de cada elemento do conjunto calculado no passo anterior:
    - Fecho-ε(q_1) U Fecho-ε(q_4) = $\{q_1\}$ U $\{q_4\}$ = $\{q_1,q_4\}$
4. Agora iremos processar o símbolo **.** a partir dos estados obtidos na etapa anterior:
    - δ(q_1,.) U δ(q_4,.) = $\{q_2\}$ U $\{q_3\}$ = $\{q_2,q_3\}$

5. Mais uma vez iremos fazer o Fecho-ε de cada elemento do conjunto calculado na etapa anterior:
    - Fecho-ε(q_2) U Fecho-ε(q_3) = {q_2} U {q_3,q_5} = {q_2,q_3,q_5}
6. Agora iremos processar o símbolo **5** a partir dos estados obtidos na etapa anterior:
    - δ(q_2,5) U δ(q_3,5) U δ(q_5,5) = {q_3} U {q_3}U Ø = {q_3}
7. E novamente iremos calcular o Fecho-ε dos elementos do conjunto que obtivemos na última etapa:
    - Fecho-ε(q_3) = {q_3,q_5}
8. Não temos mais nenhum símbolo para processar, portanto basta verificar se pelo menos um dos estados obtidos na última etapa é estado final. Neste caso, q_5 é um estado final, portanto a string **8.5** é aceita pelo nosso AFNDε.

2.3.4 Eliminação de transições vazias

Dado um AFNDε *E* qualquer, nós podemos encontrar um AFD *A* que aceita a mesma linguagem que *E*. Assim como na conversão de AFND para AFD, os estados de *A* serão subconjuntos dos estados de *E*. A única diferença é que deveremos incorporar as transições vazias de *E*, através da função Fecho-ε.

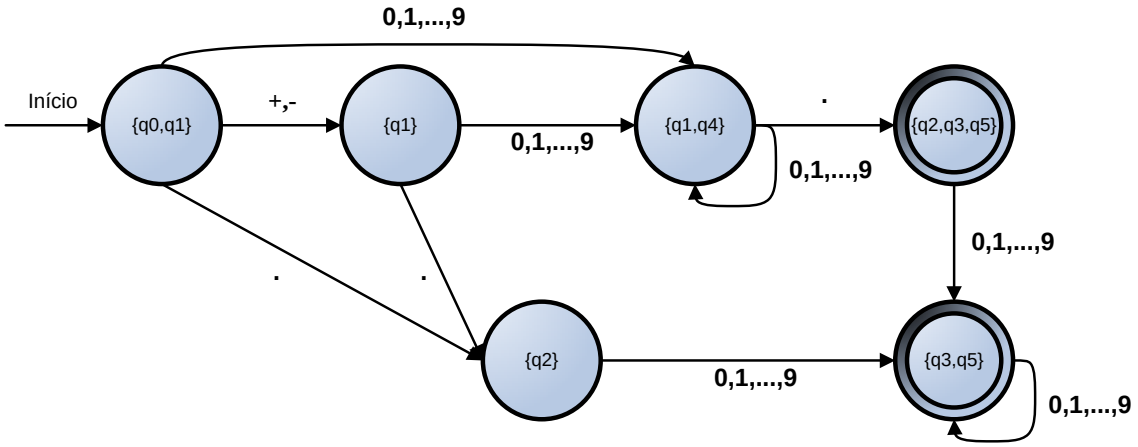
Sendo $E = (Q_E,\Sigma,\delta_E,q_0,F_E)$. Então $A = (Q_A,\Sigma,\delta_A,q_A,F_A)$ é definido assim:

1. Q_D é o conjunto dos subconjuntos de Q_E, todos os estados acessíveis de *A* são subconjuntos com Fecho-ε de Q_E, ou seja, são subconjuntos de elementos cujas transições vazias de cada elemento levam apenas a elementos que estão neste subconjunto.
2. q_A = Fecho-ε(q_0); isto é; o estado inicial de *A* é o fechamento do estado inicial de *E*.
3. F_A representa os conjuntos de estados que contêm pelo menos um estado de aceitação em *E*. Ou seja, $F_A = \{S \mid S \text{ está em } Q_D \text{ e } S \cap F_E \neq \emptyset\}$
4. $\delta_A(S,a)$ é calculado, para todo *a* em Σ e todos os conjuntos *S* em Q_A por:
    a. Seja $S = \{p_1,p_2,...,p_k\}$.
    b. Calcule $\bigcup_{i=1}^{k} \delta_E(p_i,a)$; seja esse conjunto $\{r_1,r_2,...,r_m\}$.
    c. Então $\delta_A(S,a) = \bigcup_{j=1}^{m} Fecho-\varepsilon(r_j)$

Exemplo: Vamos eliminar as transições vazias do nosso exemplo de AFNDε, que chamaremos de *E*. Construiremos um AFD *A* equivalente da seguinte forma:

1. O estado inicial de *E* é q_0, portanto o estado inicial de *A* será Fecho-ε(q_0), que é $\{q_0,q_1\}$.
2. Devemos agora obter os sucessores de q_0 e q_1 para os vários símbolos em Σ, este símbolos são os sinais de + ou -, o ponto e os dígitos de 0 a 9. Sobre + e -, q_1 não vai para lugar algum, enquanto q_0 vai para q_1. Assim δ($\{q_0,q_1\}$,+} será o Fecho-ε(q_1). Como não existe nenhuma transição vazia saindo de q_1, temos que δ($\{q_0,q_1\}$,+} = $\{q_1\}$, e da mesma forma δ($\{q_0,q_1\}$,-} = $\{q_1\}$
3. Em seguida calcularemos δ($\{q_0,q_1\}$,.}, como q_0 não vai para lugar algum lendo o ponto, q_1 e vai para q_2, devemos calcular o Fecho-ε de q_2. Como não há transições vazias saindo de q_2, δ($\{q_0,q_1\}$,.} = $\{q_2\}$
4. Finalmente, devemos calcular δ($\{q_0,q_1\}$,0}. Vemos que q_0 não vai para lugar algum lendo 0, mas q_1 vai para q_1 e q_4. Como nenhum desses dois estados tem transições vazias, concluímos que δ($\{q_0,q_1\}$,0} = $\{q_1,q_4\}$. Note que esta mesma transição é válida para todos os dígitos de 0 a 9.

Assim calculamos todos os arcos saindo de $\{q_0,q_1\}$. Todas as outras transições são calculadas de modo semelhante, conforme será demonstrado na aula, e finalmente chegaremos ao seguinte AFD *A*:



Note que, para evitar confusão, foram omitidas todas as transições para o estado Ø, assim você deve imaginar que para todo estado do diagrama em que não aparecem transições para um dado símbolo, essa transição tem como destino o estado Ø. Além disso, o estado Ø tem transições para ele mesmo para todos os símbolos de Σ.