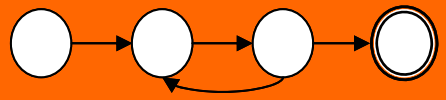
Automata e Linguagens Formais

CTC 34

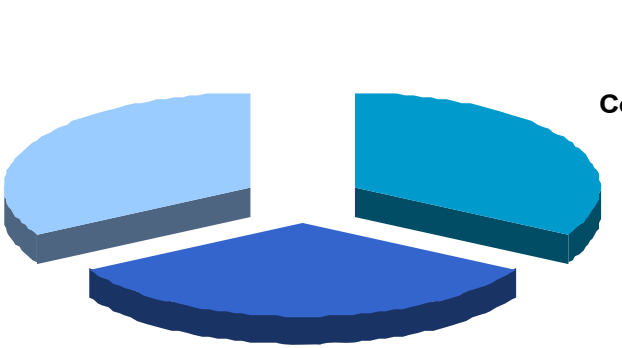


1

Prof. Carlos H. C. Ribeiro
carlos@ita.br
Ramal: 5895 Sala: 106

Motivação, bibliografia, orientações gerais
Teoria da Computação: Visão Geral
Relembrando Teoria dos Conjuntos
Grafos e árvores, provas
Cadeias, alfabetos e linguagens
Definições recursivas e com operadores
Linguagens regulares
Automata finitos: definição e exemplos

Teoria da Computação: Visão Geral



Estudo do processo computacional através de modelos abstratos de computação

Exemplos de modelos computacionais:

- Autômato finito: processamento de texto, compiladores e projeto de *hardware*.
- Gramática livre de contexto: linguagens de programação e Inteligência Artificial.

EXISTEM MODELOS GERAIS DE COMPUTADORES, INDEPENDENTES DE IMPLEMENTAÇÃO?

O QUE UM COMPUTADOR PODE FAZER? E COMO VERIFICAR ISSO?

Há problemas que podem ser resolvidos por computadores, e outros que não podem.

Teorema de Gödel

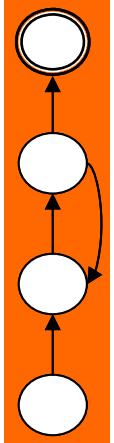
O QUE TORNA ALGUNS PROBLEMAS COMPUTÁVEIS E OUTROS NÃO COMPUTÁVEIS?

QUAIS AS CONSEQUÊNCIAS DISSO PARA O PROJETO DE COMPUTADORES?

Há problemas simples e problemas difíceis:

- Ordenamento: simples..
- Alocação de recursos: complexo.

O QUE TORNA ALGUNS PROBLEMAS FÁCEIS E OUTROS DIFÍCEIS? COMO CLASSIFICAR PROBLEMAS?



Motivação

Objetivo do curso:

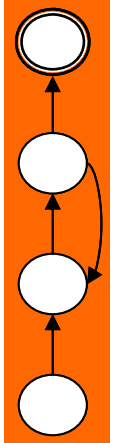
Apresentar os fundamentos da Teoria da Computação.

- Como definir o processo computacional de maneira formal?
- Quais as capacidades / limitações da computação?
- Como definir uma máquina abstrata que realiza computações?

Em suma: quais as capacidades e limitações de um computador?

Veremos que existem vários modelos, e que a capacidade computacional de uma máquina é caracterizada pela linguagem formal que ela pode reconhecer.

Quanto mais "poderosa" uma máquina, maior a complexidade da linguagem que ela pode reconhecer.



Alan Turing

- Nascido em 1912 (Londres), faleceu em 1954 (Cheshire, Inglaterra).
- Doutorado em 1929 (Univ. de Viena).
- Idéia revolucionária: propôs em 1936 a Máquina Universal capaz de realizar qualquer computação matemática, e conjecturou que problemas matemáticos que não podem ser resolvidos por esta máquina não podem ser resolvidos de nenhuma maneira por um procedimento computacional.
- Uma das 100 mais importantes personalidades do século XX (Time).

Talvez o mais importante artigo na História da Computação...

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers X, *e*, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In §8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel [1] . These results {231} have valuable applications. In particular, it is shown (§11) that the Hilbertian Entscheidungsproblem can have no solution.

In a recent paper Alonzo Church [2] has introduced an idea of "effective calculability", which is equivalent to my "computability", but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem. [3] The proof of equivalence between "computability" and "effective calculability" is outlined in an appendix to the present paper.

1. Computing machines.

We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach §9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q1, q2, ..., qR which will be called "*m*-configurations". The machine is supplied with a "tape", (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the *r*-th, bearing the symbol S(*r*) which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its *m*-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the *m*-configuration qn and the scanned symbol S(*r*). This pair qn, S(*r*) will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the *m*-configuration may be changed. Some of the symbols written down {232} will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.

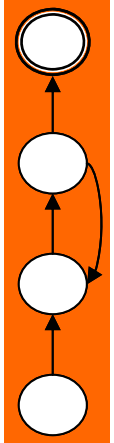
It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by "machine", "tape", "scanned", etc.

2. *Definitions.*

Automatic machines.

Kurt Gödel

- Nascido em 1906 (Brno, Rep. Tcheca), faleceu em 1978 (Princeton, EUA).
- Doutorado em 1929 (Univ. de Viena).
- Teorema provado em 1931: *qualquer sistema matemático baseado em axiomas tem proposições que não podem ser provadas*.
- Teorema de Gödel encerrou uma busca de mais de 100 anos por uma base axiomática completa para a Matemática (Hilbert, Russell e outros). É um dos dez mais importantes teoremas provados, considerando-se qualquer área da Ciência.



Bibliografia Básica

- **Introduction to Automata Theory, Languages and Computation** - J.E. Hopcroft e J.D. Ullman. Addison-Wesley. *Um clássico na área, razoavelmente completo e didático. Provas sumárias de teoremas. Apresenta soluções comentadas para alguns exercícios.*
- **Introdução à Teoria de Autômatos, Linguagens e Computação** – J.E. Hopcroft, J.D. Ullman e R. Motwami. Campus. *Tradução da 2a. edição americana de versão estendida do Hopcroft/Ullman. Vários erros de tradução. Bem completo.*
- **Elementos de Teoria da Computação** - H.R. Lewis e C.H. Papadimitriou. Bookman. 2a. edição. *Um pouco menos didático do que o anterior.Conciso e objetivo.*
- **Languages and Machines** - T. Sudkamp. Addison-Wesley. *Mais didático do que os anteriores. Ordenação não muito ortodoxa: primeiro discute linguagens e gramáticas e só então introduz a teoria de autômatos.*
- **Introduction to the Theory of Computation** – M. Sipser. PWS. *Conciso e didático.*
- **Notas de aula**

Orientações Gerais: Horários e Avaliação

Horários:

- 2 tempos semanais de teoria, 2 tempos quinzenais de prática (nanoprovas ou projetos)

•Avaliação:

- Quatro “nanoprovas” individuais com consulta (duas por bimestre), cada uma com valor 25.
- Quatro projetos em dupla (dois por bimestre), cada um com valor 25.
- Um exame final individual, sem consulta, H8.

- **Nota (1° bimestre): N1 = Sn1 + Sp1**

Sn1 = soma das notas das nanoprovas (bimestre 1)
Sp1 = soma das notas dos projetos (bimestre 1)

- **Nota (2° bimestre): N2 = Sn2 + Sp2**

Sn2 = soma das notas das nanoprovas (bimestre 2)
Sp2 = soma das notas dos projetos (bimestre 2)

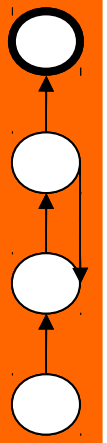
- **Nota final: (2*mN + NE)/3**

mN = média (N1,N2)
NE = nota do exame final

Projeto: Critérios

- **Máximo de 6 páginas em fonte 12.**
  - Adequação da metodologia
  - Qualidade dos resultados experimentais
  - Capacidade crítica
  - Qualidade do texto (clareza e concisão)

Cada projeto desenvolvido por grupo de <u>2 alunos</u>



Orientações Gerais: Material, Dúvidas

Página do curso – Portal TIDIA: https://tidia.ita.br/portal

Descrição e plano de aulas, notas de aula (slides pdf), calendário de atividades, notícias do curso, links úteis.

Requer cadastramento (login e senha):

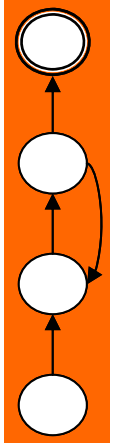
Alunos que já estão cadastrados no TIDIA devem apenas informar o userID ao professor (e-mail para carlos@ita.br)
Alunos que ainda não estão cadastrados no TIDIA devem seguir o procedimento abaixo:

1. Acessar https://tidia.ita.br/portal
2. Na aba esquerda, clicar em New Account e entrar com um userID.
3. Entrar com primeiro nome, sobrenome, email e senha nos campos apropriados.
4. Clicar em Create Account. O cadastramento no sistema TIDIA estará realizado.

Uma vez feito o cadastramento, enviar email ao professor (carlos@ita.br) avisando. Em seguida, receberá um convite para entrar no site da disciplina, que poderá passar a acessar a partir de https://tidia.ita.br/portal a partir de então.

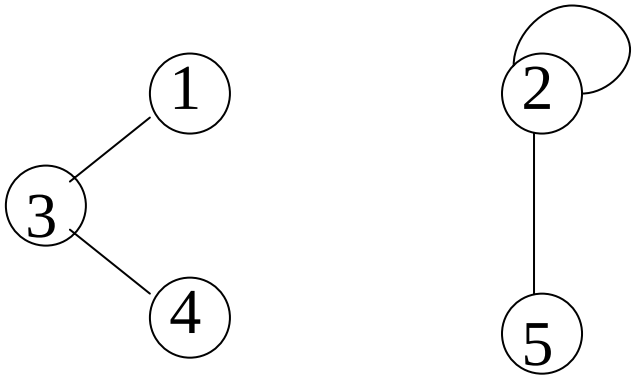
Dúvidas sobre a disciplina: agendar reunião via

Ramal 5895
E-mail carlos@ita.br



Grafos

Um grafo G=(V,E) é um conjunto finito de vértices (nós) V e um conjunto de pares (ligações) de vértices E.



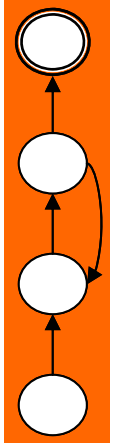
V={1,2,3,4,5}

E={(n,m)| n+m=4 ou n+m=7}

Um **caminho** em um grafo G=(V,E) é uma sequência de vértices $v_1,v_2,...,v_k$, $k \geq 1$, tais que existem ligações (v_i, v_{i+1}), $1 \leq i \leq k$.

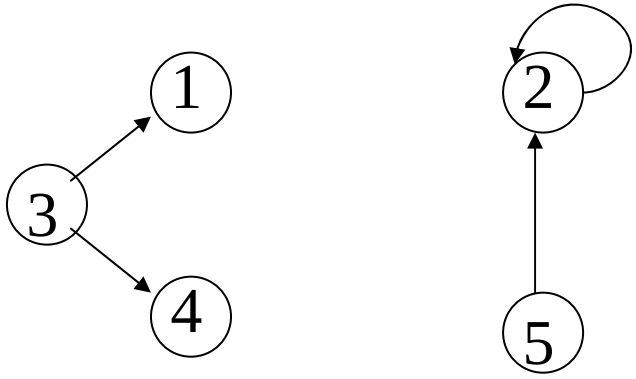
O **comprimento do caminho** é k-1. Se $v_1 = v_k$, o caminho é um **ciclo**.

O **grau** de um nó é o número de ligações àquele nó.



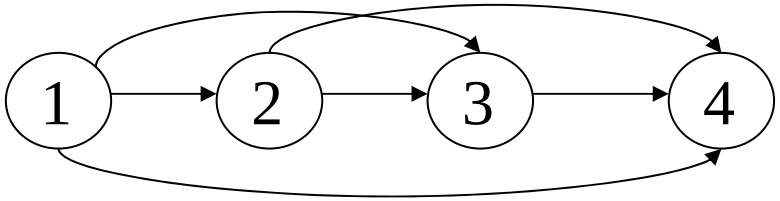
Grafos Direcionados

Um grafo direcionado G=(V,E) é um conjunto finito de vértices (nós) V e um conjunto de pares ordenados (arcos) de vértices E.

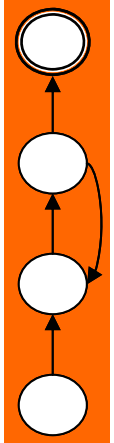


Um caminho em um grafo direcionado G=(V,E) é uma sequência de vértices $v_1,v_2,...,v_k$, $k \geq 1$, tais que existem arcos (v_i, v_{i+1}), $1 \leq i \leq k$.

Se $v \rightarrow w$ é um arco, v é dito **predecessor** de w e w é dito **sucessor** de v.



$\{\{1,2,3,4\},\{i \rightarrow j \mid i<j\}\}$



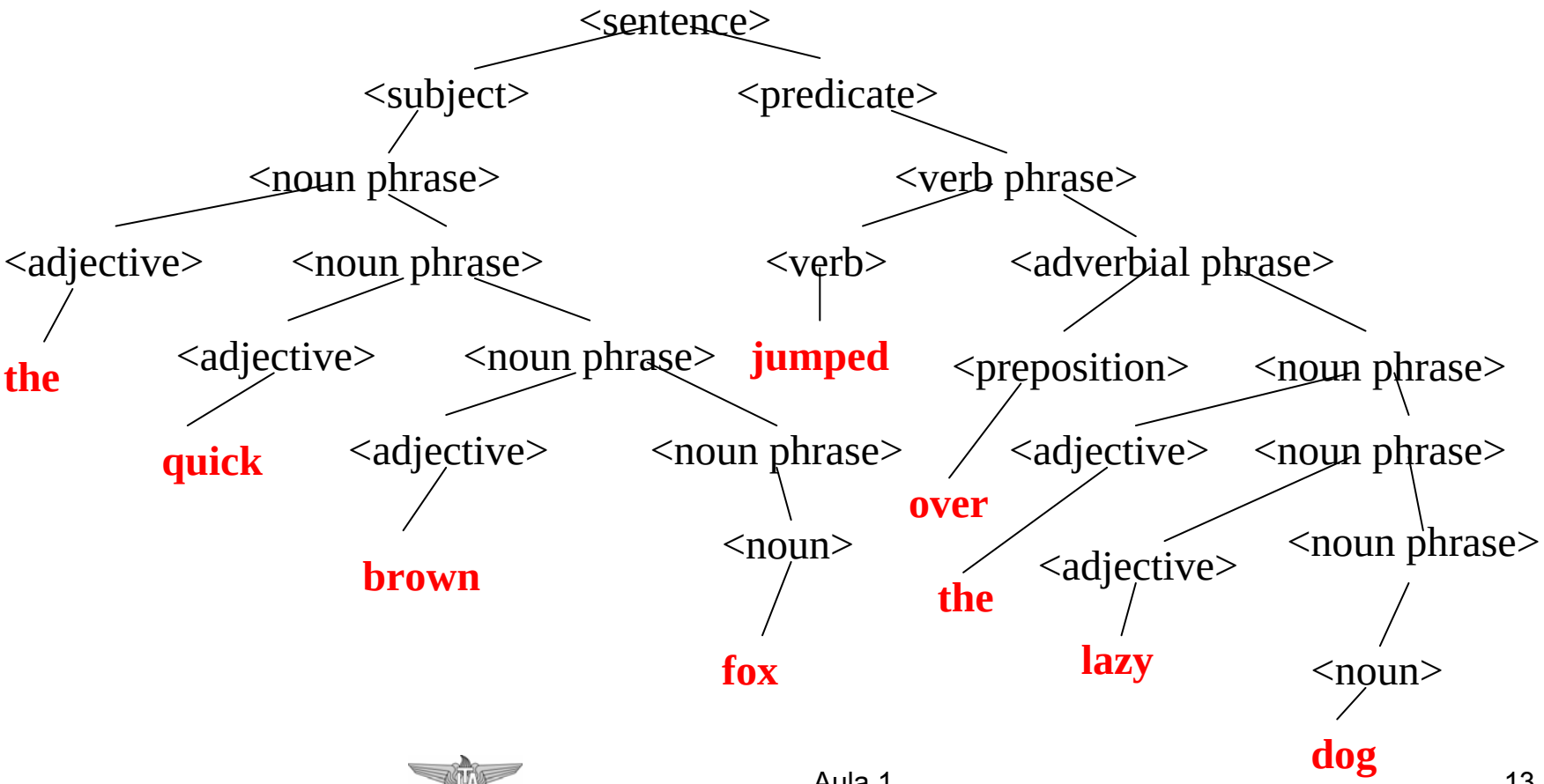
Árvores

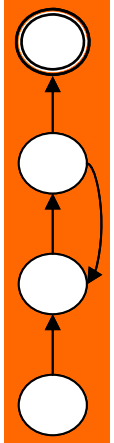
Uma árvore é um grafo direcionado com as seguintes propriedades:

• Um nó *raiz*, sem predecessores, a partir do qual há um caminho para todo nó.

• Todos os outros nós têm exatamente um predecessor.

•Os sucessores de cada nó são ordenados a partir da esquerda.

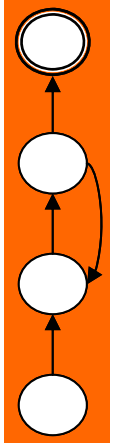




Provas

Três mecanismos básicos:

1. Prova por construção
2. Prova por contradição (ou por absurdo)
3. Prova por indução



Prova por construção: exemplo

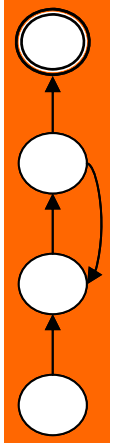
Teorema: Para qualquer número par $n > 2$ existe um grafo formado apenas por n vértices de grau 3.

Prova: *Construa um grafo G=(V,E) da seguinte maneira:*

$V = \{0,1,...,n-1\}$

$E = \{\{i, i+1\} \text{ para } 0 \leq i < n-1\} \cup \{\{n-1,0\}\} \cup \{\{i, i+n/2\} \text{ para } 0 \leq i \leq n/2-1\}$

Desenhe o grafo e verifique.



Prova por contradição: exemplo

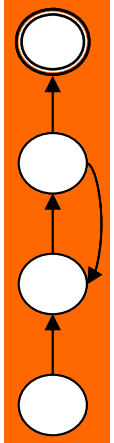
Teorema: $\sqrt{2}$ é irracional

Prova: *Suponha* $\sqrt{2}$ *racional. Então* $\sqrt{2} = \frac{m}{n}$

onde m e n são inteiros. Divida agora m e n pelo MDC(m,n), de modo que a fração não tem seu valor alterado, mas agora m e n não podem mais ser ambos pares.

Multiplicando os dois lados por n e elevando ao quadrado: $2n^2 = m^2$.
Portanto, m^2 *é par. Logo, m é par, e então m=2k. Substituindo:*

$2n^2=(2k)^2=4k^2 \Rightarrow n^2 = 2k^2 \Rightarrow$ *n é par (contradição)*



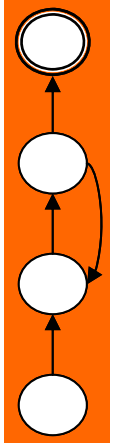
Prova por Indução

O Princípio da Indução Finita (P.I.F.):

Seja P(n) uma proposição. Então, P(n) é verdadeira se:

•P(0) é verdadeira (base)

•P(n-1) implica P(n) para $n \geq 1$ (passo indutivo)



Provas por Indução: Exemplo 1

Prove que: $$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

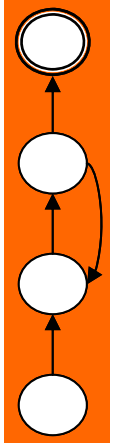
i) Base para n=0: $$\sum_{i=0}^{0} i^2 = 0, \quad \frac{0(0+1)(2.0+1)}{6} = 0$$

ii) Passo indutivo: $$\sum_{i=0}^{n-1} i^2 = \frac{(n-1)(n)(2n-1)}{6}$$

$$\sum_{i=0}^{n} i^2 = \sum_{i=0}^{n-1} i^2 + n^2 = \frac{(n-1)(n)(2n-1)}{6} + n^2$$

$$= \frac{n(n+1)(2n+1)}{6}$$

QED



Provas por Indução: Exemplo 2

Prove que 3 é um fator de $n^3 - n + 3$

i) Base para n=0: $0^3 - 0 + 3 = 3 = 1.3$

ii) Passo indutivo: $n^3 - n + 3 = k.3$

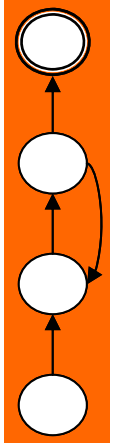
$$(n+1)^3 - (n+1) + 3 = n^3 + 3n^2 + 2n + 3$$

$$= n^3 - n + 3 + (n^2 + n)3$$

$$= k.3 + (n^2 + n)3$$

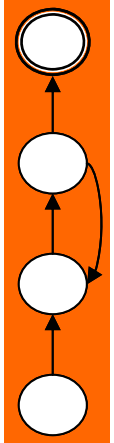
$$= k'.3$$

QED



Linguagens: Conceitos Básicos

- Um **símbolo** é um símbolo...
- Um **alfabeto** Σ é um conjunto finito de símbolos.
- Uma **cadeia *(string)*** é uma sequência finita de símbolos.
  - Sequência de **zero** símbolos: ε (*string* vazia)
  - Conjunto de **todas** as cadeias de um alfabeto Σ: Σ^*
  - Operação de **Concatenação** de duas cadeias *w* e *v*: *wv*
- Uma **linguagem** é um subconjunto de Σ^*.
- A **sintaxe** da linguagem é o conjunto das propriedades satisfeitas por suas cadeias.

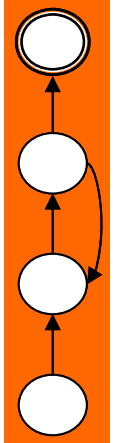


Exemplos

- O conjunto vazio $\varnothing$ é uma linguagem.
- O conjunto formado pela cadeia vazia $\{\varepsilon\}$ é uma linguagem.
- O conjunto de palíndromos sobre {0,1}.
  - Símbolos: 0 1
  - Alfabeto: {0,1}
  - Linguagem: $\{\varepsilon, 0, 1, 00, 11, 010, 1101011, ...\}$
  - $u = 00$, $v = 010 \rightarrow uv = 00010$
- Σ^* é uma linguagem sobre um alfabeto Σ.
- $\Sigma=\{a\}$. $\Sigma^*=?$
- $\Sigma=\{0,1\}$. $\Sigma^*=?$

Qual a sintaxe da linguagem Σ^* ?

- Exercício: Uma linguagem natural encaixa-se na definição formal de linguagem? Em caso positivo, que símbolos seriam mais adequados: letras ou palavras?



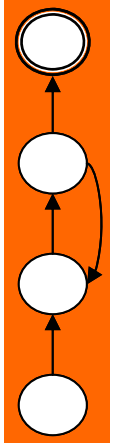
Representações Finitas de Linguagens

Vantagens:

- evitar enumeração exaustiva
- explicitar a sintaxe da linguagem de modo não-ambíguo

Técnicas:

- representação recursiva
- representação com uso de operadores sob linguagens



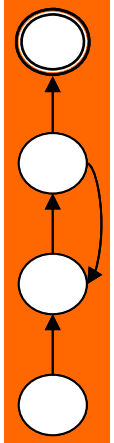
Representação Recursiva

Idéia:

1. Defino um conjunto de elementos básicos da linguagem.

2. Defino um passo recursivo que permite obter qualquer string que obedeça à sintaxe da linguagem.

3. Imponho o fechamento: passos 1 e 2 definem todos os elementos da linguagem, e somente estes.



Representação Recursiva: Exemplos

L = cadeias sobre {a,b} começando com a e com comprimento par.

• aa, ab $\in$ L (elementos básicos)

• $u \in L \Rightarrow u$aa, $u$ab, $u$ba, $u$bb $\in$ L (passo recursivo)

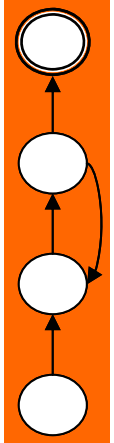
• $u \in$ L apenas se obtido dos elementos básicos por um número finito de aplicações do passo recursivo (fechamento)

L = cadeias sobre {a,b} em que cada b é imediatamente precedido por um a.

• $\varepsilon \in$ L (elemento básico)

• $u \in L \Rightarrow u$a, $u$ab $\in$ L (passo recursivo)

• $u \in$ L apenas se obtido dos elementos básicos por um número finito de aplicações do passo recursivo (fechamento)



Operadores sobre Linguagens

a) Concatenação de linguagens X e Y: XY = {*uv* | *u* ∈X **e** *v* ∈ Y}

Exemplo: X = {a,b,c}, Y = {abb,ba}

XY = {aabb, aba, babb, bba, cabb, cba}

$X^0 = \{\varepsilon\}$

$X^1 = X = \{a,b,c\}$

$X^2 = XX = \{aa, ab, ac, ba, bb, bc , ca, cb, cc\}$

b) União de linguagens X e Y: $X \cup Y$

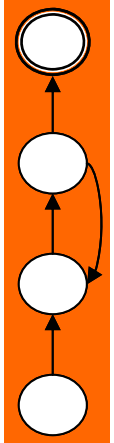
Exemplo: X = {a,b,c}, Y = {abb,ba}

$X \cup Y = \{a, b, c, abb, ba\}$

c) $X^* = \bigcup_{i=0}^{\infty} X^i$ (**Fechamento de Kleene** da linguagem X)

d) $X^+ = \bigcup_{i=1}^{\infty} X^i$

(Note que $X^+ = XX^*$)



Representação com Operadores: Exemplos

Exemplo 1: L = cadeias sobre {a,b} com comprimento par.

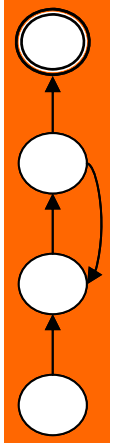
$$\{aa, bb, ab, ba\}^*$$

Exemplo 2: L = cadeias que começam e terminam com a e que contém pelo menos um b.

$$\{a\}\{a, b\}^*\{b\}\{a, b\}^*\{a\}$$

Exemplo 3: L = cadeias sobre {a,b} começando com aa ou terminando com bb.

$$\{aa\}\{a, b\}^* \cup \{a, b\}^*\{bb\}$$

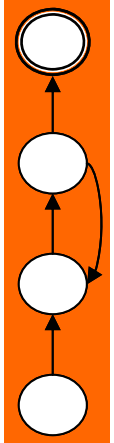


Linguagens Regulares: Definição

- $\emptyset$, $\{\varepsilon\}$ e $\{a\}$, para todo $a \in \Sigma$, são linguagens regulares.
- **Passo recursivo:** X e Y linguagens regulares $\Rightarrow$ $X \cup Y$, XY e X^* são linguagens regulares.
- **Fechamento:** X é linguagem regular apenas se obtido dos elementos básicos por um número finito de aplicações do passo recursivo.

Uma notação simplificada: **expressões regulares**

$\{a\} \rightarrow \mathbf{a}$, $\{a, b\} \rightarrow \mathbf{a} \cup \mathbf{b}$, precedência * > concatenação > $\cup$



Exemplo: O conjunto de cadeias contendo a sub-cadeia bb.

$\{a\}, \{b\} \rightarrow \{a, b\}^*$	união, fechamento de Kleene
$\{b\}\{b\} \rightarrow \{bb\}$	concatenação
$\{a, b\}^*\{bb\}\{a, b\}^*$	concatenação (duas vezes)
$\mathbf{(a \cup b)^*\ bb\ (a \cup b)^*}$	**expressão regular correspondente**

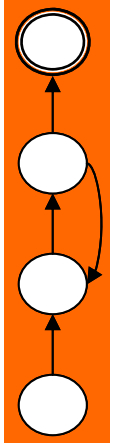
Exemplo: O conjunto de cadeias contendo dois ou mais b's.

$\mathbf{a^* ba^* b\ (a \cup b)^*}$

$\mathbf{(a \cup b)^* ba^* ba^*}$

$\mathbf{(a \cup b)^*\ b(a \cup b)^* b(a \cup b)^*}$

expressões **equivalentes**

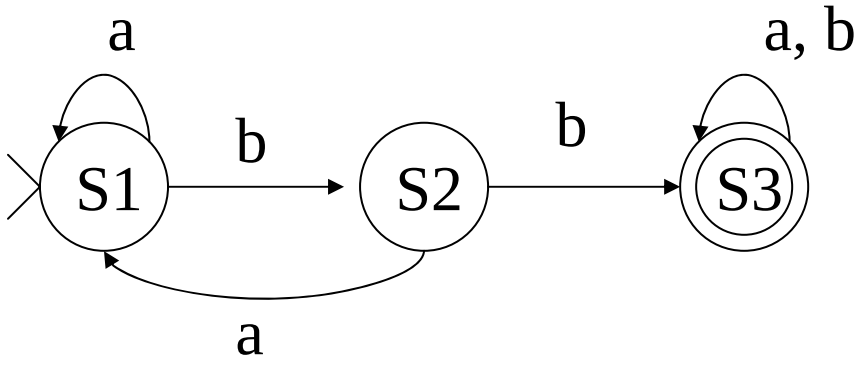


Autômato Finito

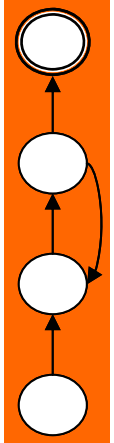
- Modelo de sistema dinâmico em que:
  - Existe um número finito de **estados** do sistema.
  - Existem transições entre estados, mediadas por entradas provenientes de um conjunto discreto finito (alfabeto Σ).
  - Para cada estado, **todas** as transições produzidas por cada um dos símbolos do alfabeto estão definidas.

Estados: S1 (inicial), S2, S3 (final)

Entradas: {a,b}



Representação por grafo direcionado (**diagrama de transição**)

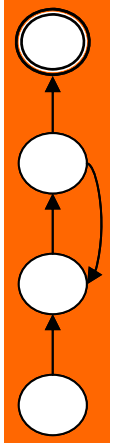


Automata Finitos: Definição Formal

Um autômato finito é uma **quíntupla**

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

- Q = conjunto finito de estados
- Σ = alfabeto
- δ = função de transição Q x $\Sigma \rightarrow$ Q
- q_0 = estado inicial
- F $\subseteq$ Q = conjunto de estados finais (aceitadores)

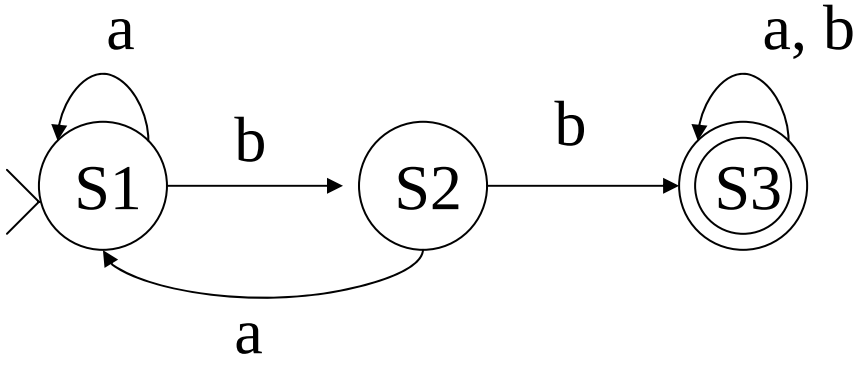


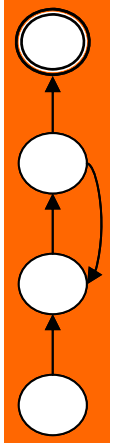
Automata Finitos: Exemplos

1. $Q = \{S1, S2, S3\}$, $\Sigma = \{a,b\}$, $q_0 = S1$, $F = \{S3\}$

δ:

	a	b
S1	S1	S2
S2	S1	S3
S3	S3	S3

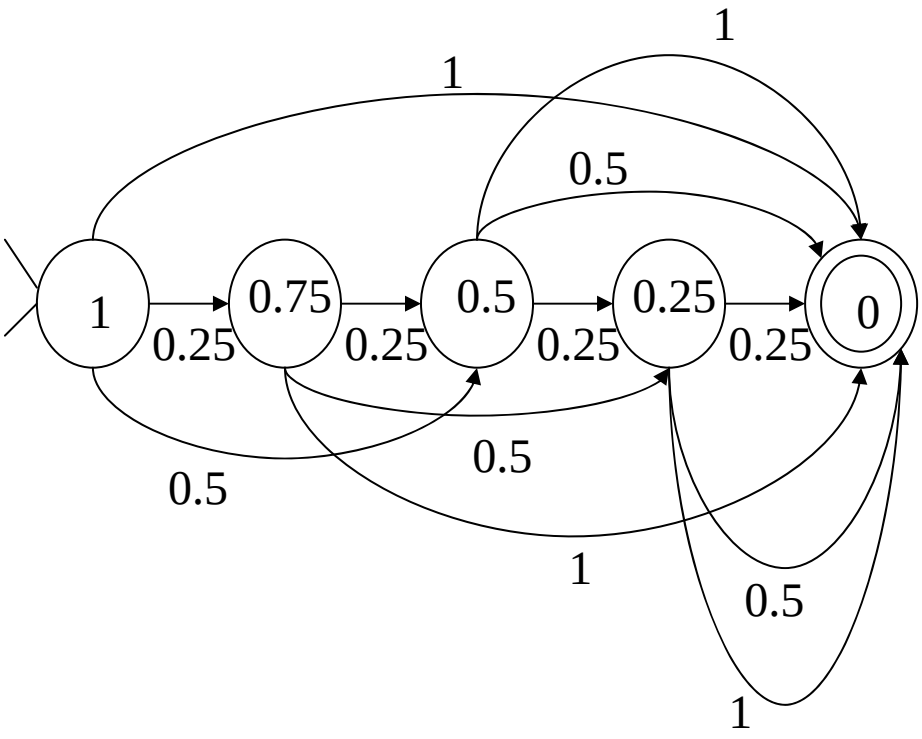




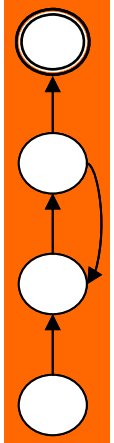
2. Máquina para vender bilhetes de metrô (R$ 1,00 cada). Não dá troco.

Q = {S1=1, S2=0.75, S3=0.5,S4=0.25,S5=0}, cada estado identificando o quanto falta para que se libere um bilhete de metrô.

$\Sigma = \{1, 0.5, 0.25, 0\}$, $\delta = \ldots$, $q_0 = S1 = 1.0$, $F = \{S5=0\}$



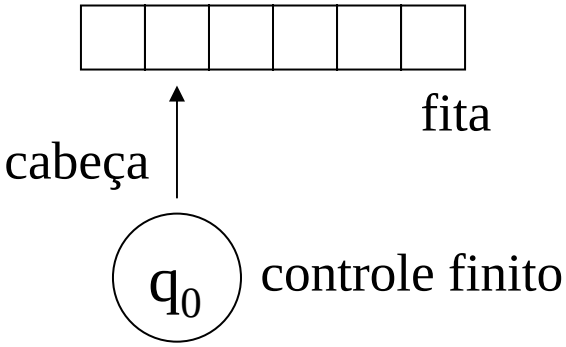
Observe que o estado codifica o que é relevante (memória implícita).



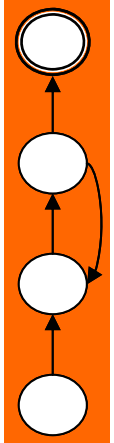
Automata Finitos e Máquinas de Estados

Um automato finito é uma **máquina de estados**

- um registrador de estado interno (controle finito) + uma fita segmentada + cabeça de leitura



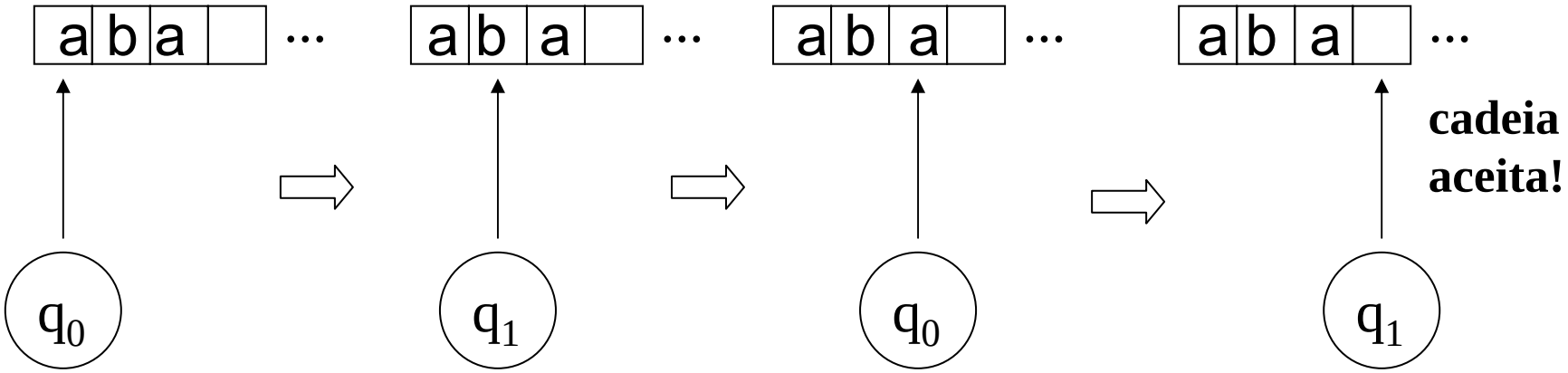
- fita: armazena uma cadeia de Σ (1 símbolo/segmento).
- cabeça: lê segmento da fita (um símbolo da cadeia).
- registrador: altera estado de acordo com δ e move a fita um segmento para a esquerda (computação).
- uma computação **termina** quando a cadeia "acaba".
- cadeia é **aceita** pelo AF se a computação termina em um estado $q \in F$.
- cadeia é **rejeitada** pelo AF se a computação termina em um estado $q \notin F$.

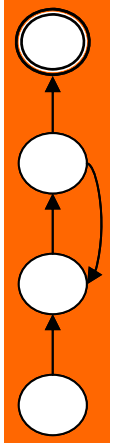


Exemplos

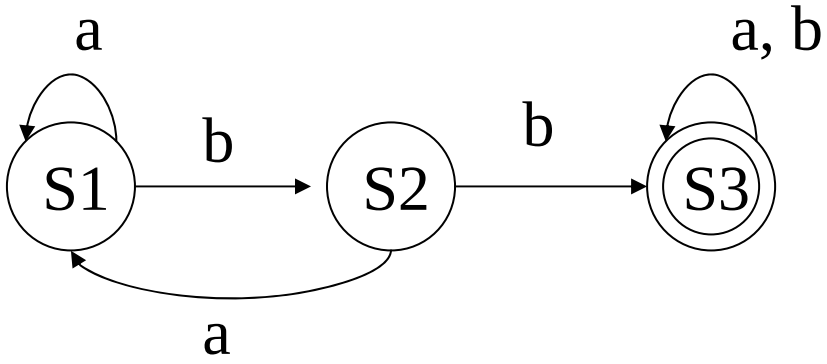
1. M: $Q = \{q_o, q_1\}$ $\Sigma = \{a, b\}$ $F = \{q_1\}$

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_0$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_0$



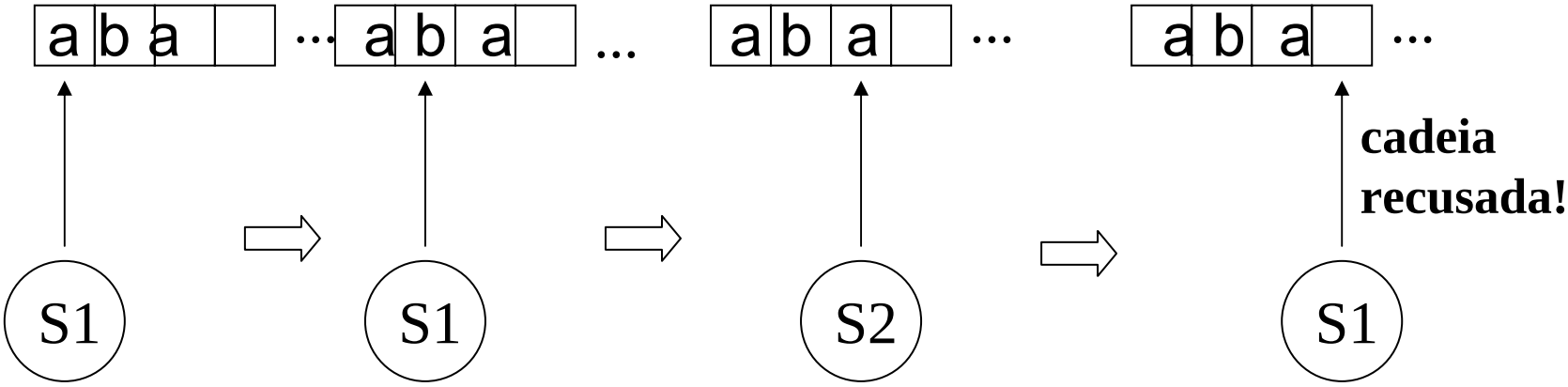


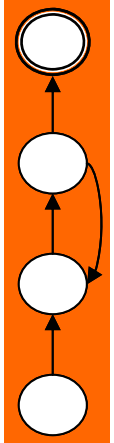
2.



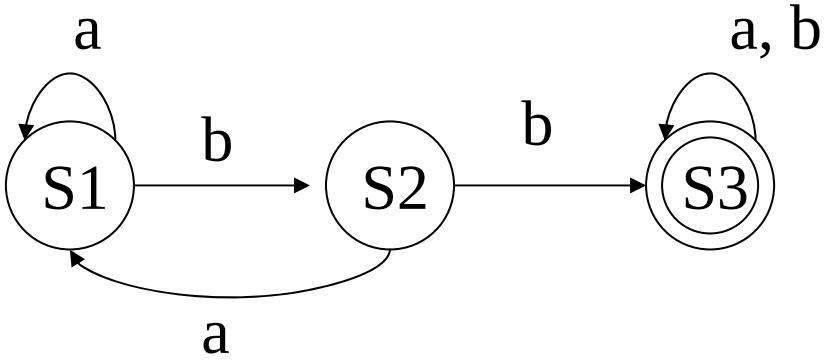
M: $Q = \{S1, S2, S3\}$ $\Sigma = \{a, b\}$ $F = \{S3\}$ $q_o = S1$

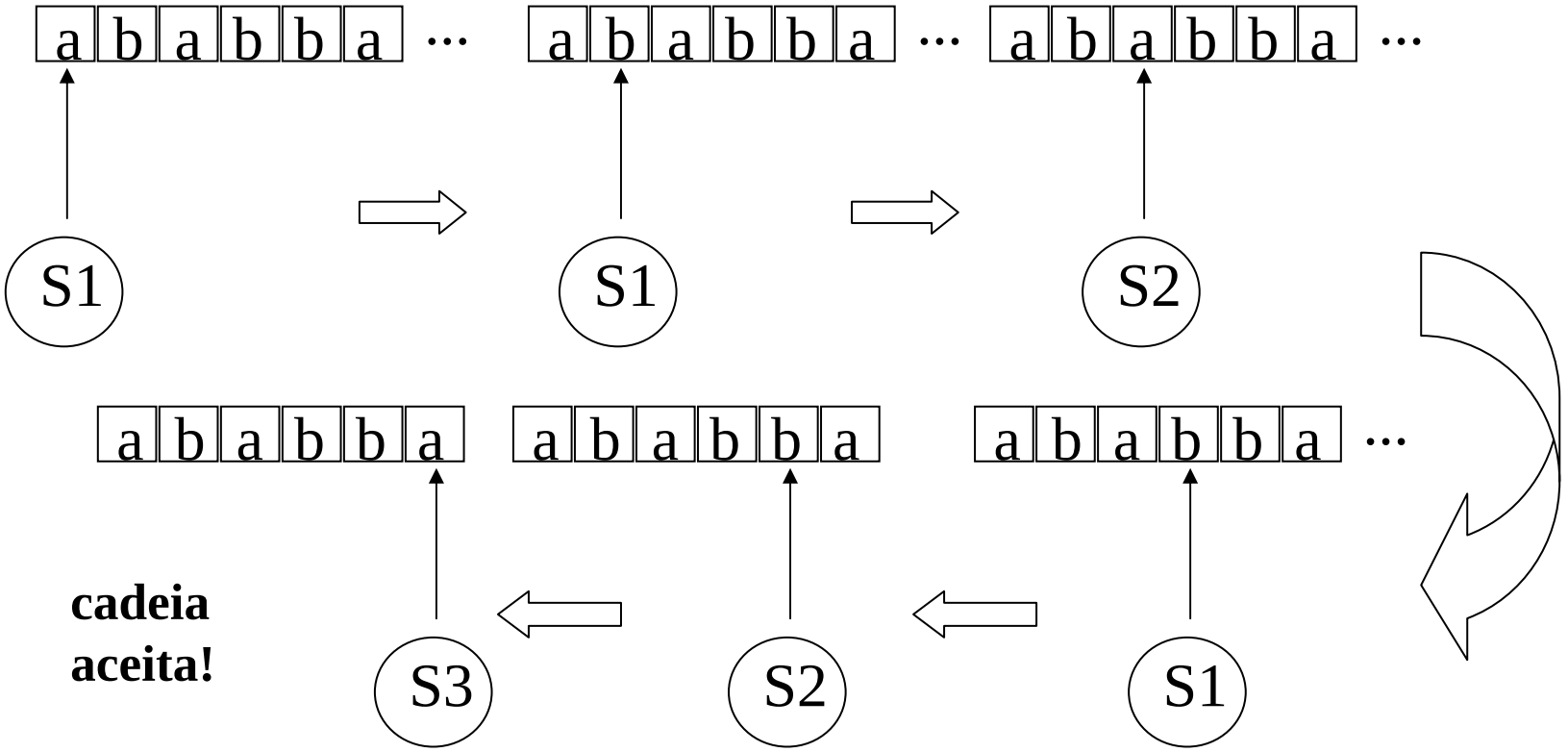
$\delta(S1, a) = S1$
$\delta(S1, b) = S2$
$\delta(S2, a) = S1$
$\delta(S2, b) = S3$

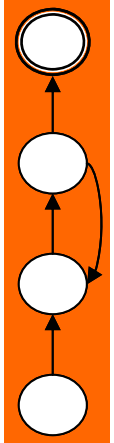




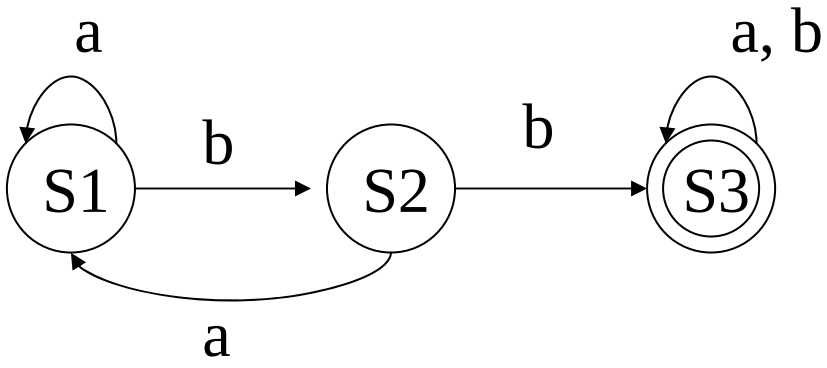
3.


a
a, b
b
b
S1
S2
S3
a


a b a b b a ...
a b a b b a ...
a b a b b a ...
S1
S1
S2
a b a b b a
a b a b b a
a b a b b a ...
cadeia
aceita!
S3
S2
S1

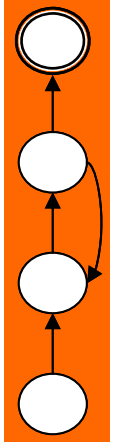


4.



Este automato aceita qualquer cadeia que contenha dois b´s seguidos...

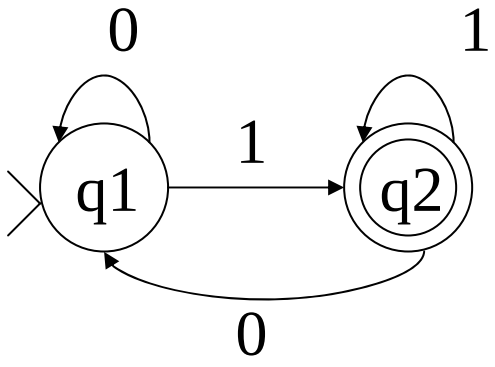
Diz-se que este automato finito **reconhece** a linguagem definida por cadeias com dois b´s seguidos.



5. M_5: Q = {q1, q2}, Σ = {0,1}, δ , q_0 = q1, F = {q2}

δ:

	0	1
q1	q1	q2
q2	q1	q2

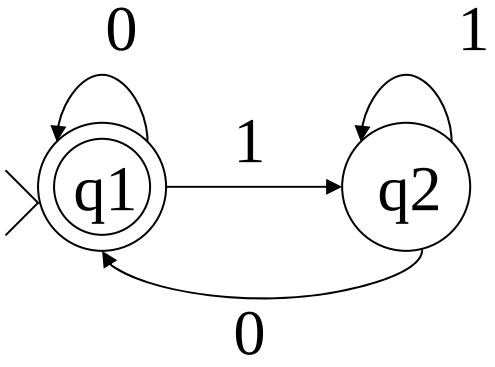


$L(M_5)$ = ?

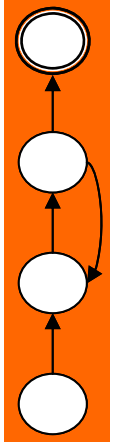
6. M_6 : Q = {q1, q2}, Σ = {0,1}, δ, q_0 = q1, F = {q1}

δ:

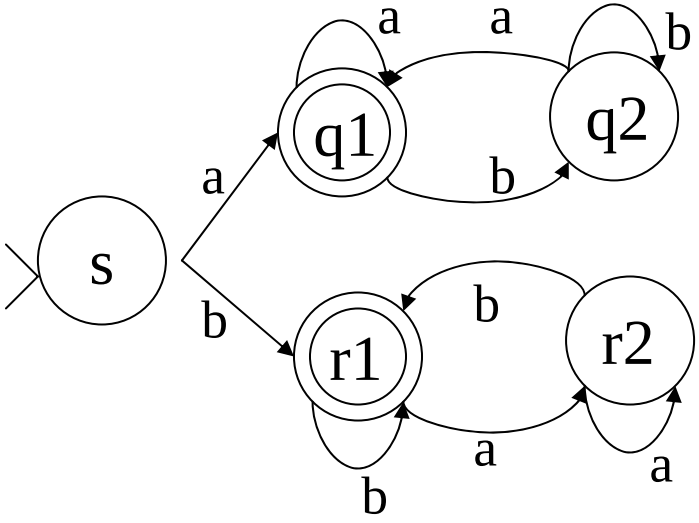
	0	1
q1	q1	q2
q2	q1	q2



$L(M_6)$ = ?

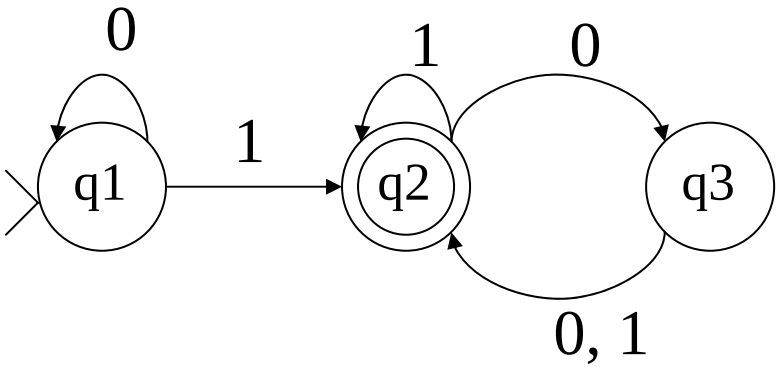


7. M_7:

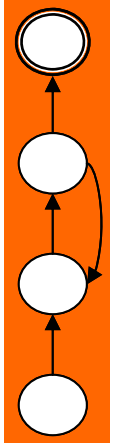


$L(M_7) = ?$

8. M_8 :



$L(M_8) = ?$



AFs: Mais Definições

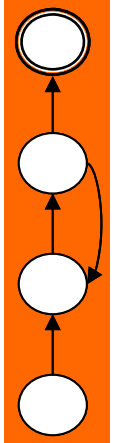
Seja $M = (Q, \Sigma, \delta, q_0, F)$ um AF

- A **configuração instantânea** [q_i, **w**] de um AF corresponde ao estado q_i e cadeia *w* ainda não processados no dado instante (um elemento de Q x Σ^*). A configuração instantânea define o comportamento futuro do AF.
- A função $\vdash$ em Q x Σ^+ é definida por [q_i, a**w**] $\vdash$ [$\delta(q_i, a)$, **w**]
- A notação [q_i, **u**] $\vdash^*$ [q_j, **v**] é usada para indicar que uma dada configuração [q_j, *v*] pode ser obtida de [q_i, *u*] por zero ou mais transições.
- A **função de transição estendida** $\hat{\delta}$ de Q x Σ^* em Q descreve formalmente a operação de um AF sobre uma cadeia, e é definida por:

$$\hat{\delta}(q,\varepsilon) = q$$

$$\hat{\delta}(q,\mathbf{w}v) = \delta(\hat{\delta}(q,\mathbf{w}), v)$$

- Uma cadeia *x* é aceita por *M* se $\hat{\delta}(q_0, \mathbf{x}) = p \in F$.
- A **linguagem L(*M*) aceita por *M*** é o conjunto de cadeias em Σ^* aceitas por *M.*
- Dois AFs que aceitam a mesma linguagem são ditos **equivalentes**.



Exemplo

M:Q = $\{q_o, q_1, q_2\}$

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

$F = \{q_2\}$

δ	**a**	**b**
$\mathbf{q_0}$	q_0	q_1
$\mathbf{q_1}$	q_0	q_2
$\mathbf{q_2}$	q_2	q_2

$[q_o, abba] \vdash [q_o, bba] \vdash [q_1, ba] \vdash [q_2, a] \vdash [q_2, \varepsilon]$

$[q_o, abab] \vdash [q_o, bab] \vdash [q_1, ab] \vdash [q_0, b] \vdash [q_1, \varepsilon]$

$\hat{\delta}(q_0, abba) = q_2$ **cadeia aceita!**

$\hat{\delta}(q_0, abab) \neq q_2$ **cadeia rejeitada!**