ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 2 – Introdução a autômatos

Profa. Ariane Machado Lima
ariane.machado@usp.br

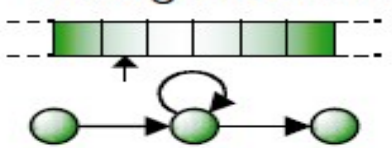
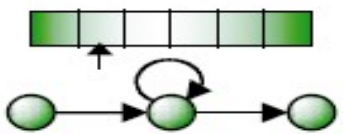
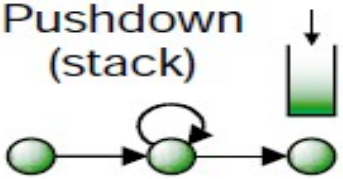
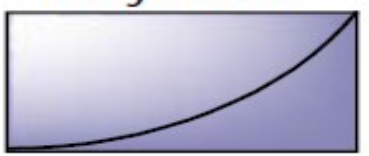
Disciplina

- Ordem dos temas:
  - Teoria dos autômatos (teoria e prática)
  - Computabilidade
  - Complexidade

Cap 1 – Linguagens regulares

- Autômatos finitos determinísticos
- Autômatos finitos não determinísticos
- Relação com modelos de Markov
- Expressões regulares
- Gramáticas regulares
- Linguagens não-regulares

Linguagens, modelos computacionais (dispositivos, gramáticas) e suas complexidades

Recursively enumerable languages	Turing machine 	Unrestricted $Baa \rightarrow A$	Undecidable 
Context-sensitive languages	Linear bounded 	Context sensitive $At \rightarrow aA$	Exponential? 
Context-free languages	Pushdown (stack) 	Context free $S \rightarrow gSc$	Polynomial 
Regular languages	Finite-state automaton 	Regular $A \rightarrow cA$	Linear 

Cap 1 – Linguagens regulares

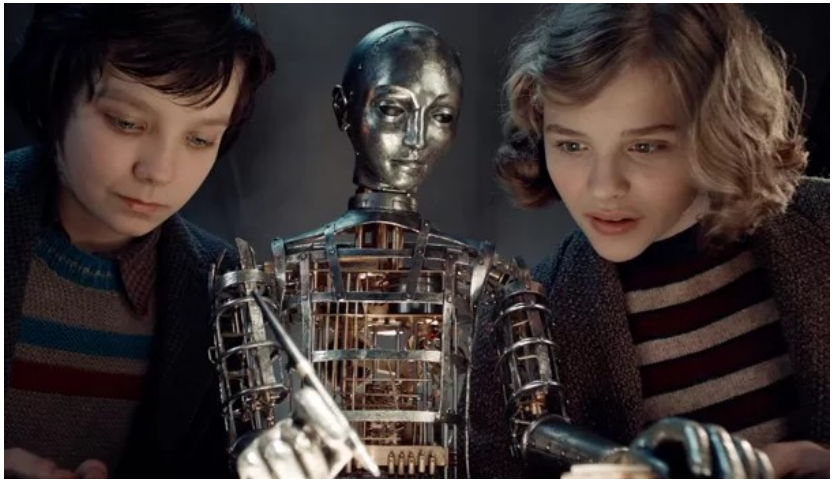
- Autômatos finitos determinísticos
- Autômatos finitos não determinísticos
- Relação com modelos de Markov
- Expressões regulares
- Gramáticas regulares
- Linguagens não-regulares

Modelos de computação

- Necessidade de um modelo para entender o que é possível computar
  - 1936 : Máquinas de Turing (modelo atual de computador) – história da computação?

ENIAC em 1946...

- Vários modelos computacionais com diferentes características (e complexidades)
- O modelo mais simples:
  - Máquina de estados finitos ou
  - Autômato de estados finitos ou
  - Autômato finito
  - *Finite State Automaton (FSA)*
  - 1943 – tentativa de modelar o raciocínio humano (base para as redes neurais artificiais)

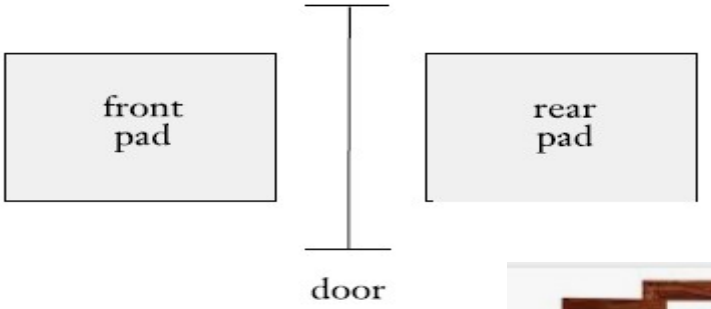


Cena do filme "*A invenção de Hugo Cabret*"

  - Séc 18: autômatos – bonecos automatizados (robôs) - https://www.youtube.com/watch?v=Jwof3KriEHw
  - Documentário: https://www.youtube.com/watch?v=gdSRAKRuZsE

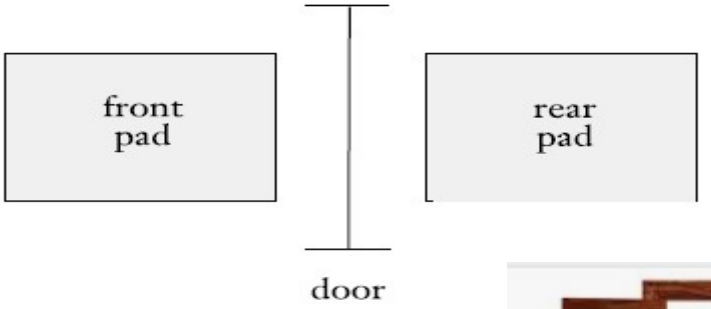
Autômatos finitos (quase)

- O exemplo de um controlador de portas que abre (para trás) só para quem está chegando
- Um sensor na frente e outro atrás
- Quais são os estados possíveis da porta?



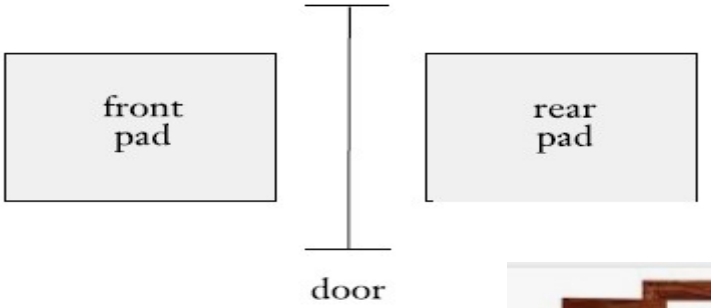
Autômatos finitos (quase)

- O exemplo de um controlador de portas que abre (para trás) só para quem está chegando
- Um sensor na frente e outro atrás
- Quais são os estados possíveis da porta?
  - Aberta / Fechada



Autômatos finitos (quase)

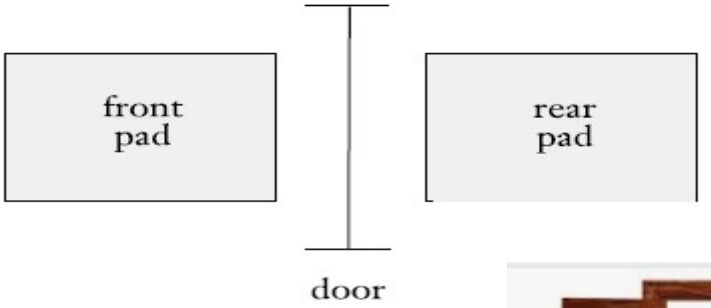
- O exemplo de um controlador de portas que abre (para trás) só para quem está chegando
- Um sensor na frente e outro atrás
- Quais são os estados possíveis da porta?
  - Aberta / Fechada



- Entradas possíveis vindas dos sensores (presença de pessoas):

Autômatos finitos (quase)

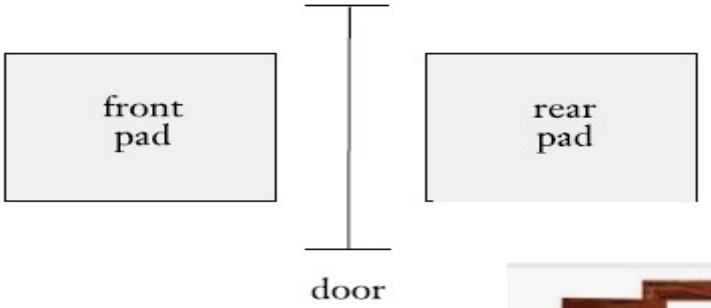
- O exemplo de um controlador de portas que abre (para trás) só para quem está chegando
- Um sensor na frente e outro atrás
- Quais são os estados possíveis da porta?
  - Aberta / Fechada



- Entradas possíveis vindas dos sensores (presença de pessoas):
  - FRONT (tem gente só na frente)
  - REAR (tem gente só atrás)
  - BOTH (tem gente na frente e atrás)
  - NEITHER (ninguém na frente nem atrás)

Autômatos finitos (quase)

- O exemplo de um controlador de portas que abre (para trás) só para quem está chegando
- Um sensor na frente e outro atrás
- Quais são os estados possíveis da porta?
  - Aberta / Fechada

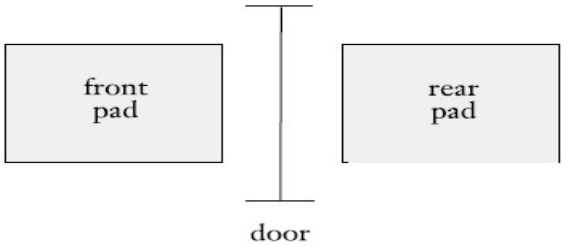


- Entradas possíveis vindas dos sensores (presença de pessoas):
  - FRONT (tem gente só na frente)
  - REAR (tem gente só atrás)
  - BOTH (tem gente na frente e atrás)
  - NEITHER (ninguém na frente nem atrás)

Como deve ser (e como descrever) o funcionamento desse controlador?

Autômatos finitos (quase)

- O exemplo de um controlador de portas que abre (para trás) só para quem está chegando
- Um sensor na frente e outro atrás
- Quais são os estados possíveis da porta?
  - Aberta / Fechada

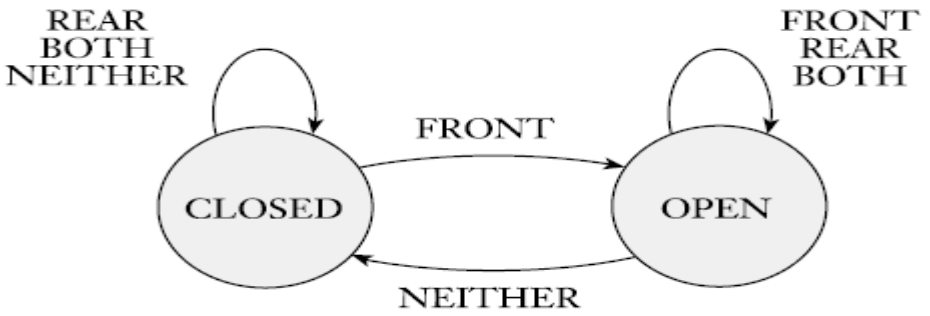


- Entradas possíveis vindas dos sensores (presença de pessoas):
  - FRONT (tem gente só na frente)
  - REAR (tem gente só atrás)
  - BOTH (tem gente na frente e atrás)
  - NEITHER (ninguém na frente nem atrás)

	FRONT	REAR	BOTH	NEITHER
OPEN	OPEN	OPEN	OPEN	CLOSED
CLOSED	OPEN	CLOSED	CLOSED	CLOSED

Autômatos finitos (quase)

- Representação gráfica do funcionamento de um controlador de portas

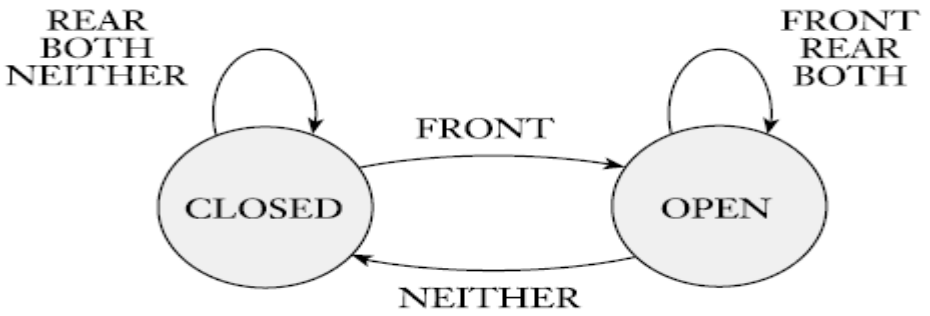


Estados: círculos
Entradas: texto nas setas indicando mudança de estado

	FRONT	REAR	BOTH	NEITHER
OPEN	OPEN	OPEN	OPEN	CLOSED
CLOSED	OPEN	CLOSED	CLOSED	CLOSED

Autômatos finitos (quase)

- O que esse controlador precisa guardar em memória?

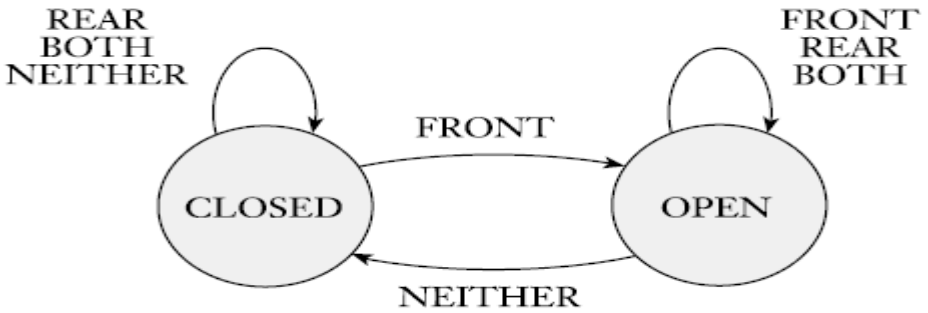


Estados: círculos
Entradas: texto nas setas indicando mudança de estado

	FRONT	REAR	BOTH	NEITHER
OPEN	OPEN	OPEN	OPEN	CLOSED
CLOSED	OPEN	CLOSED	CLOSED	CLOSED

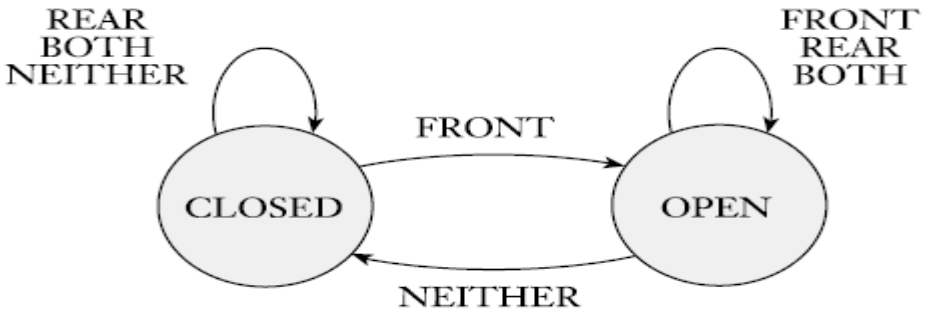
Autômatos finitos (quase)

- O que esse controlador precisa guardar em memória?
  - Estado atual (aberto/fechado – neste caso 1 bit)
- Vários outros dispositivos (ex: eletrodomésticos) podem ser implementados de forma semelhante, com uma memória limitada



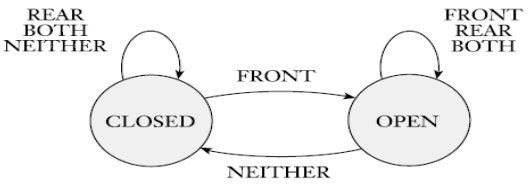
Autômatos finitos

- A diferença entre este controlador de portas em um autômato finito (AF) é que um AF precisa definir qual é o estado inicial e qual(is) é(são) o(s) estado(s) final(is)



Autômatos finitos

- Um AF pode ser definido por um diagrama de estados



Círculos: estados

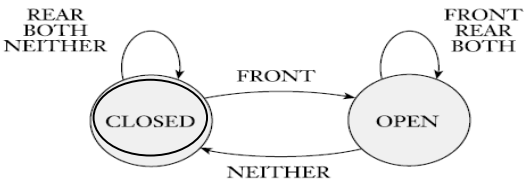
Setas entre estados: define a transição entre dois estados após a leitura do símbolo que está sobre a seta

Círculos duplos: estados finais ou de aceitação

Seta sem início (somente uma): indica quem é o estado inicial

Autômatos finitos

- Um AF pode ser definido por um diagrama de estados



Círculos: estados

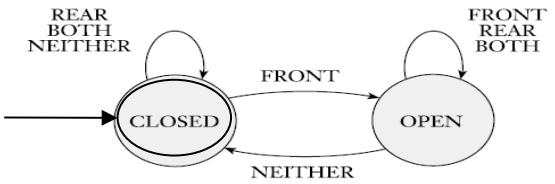
Setas entre estados: define a transição entre dois estados após a leitura do símbolo que está sobre a seta

Círculos duplos: estados finais ou de aceitação

Seta sem início (somente uma): indica quem é o estado inicial

Autômatos finitos

- Um AF pode ser definido por um diagrama de estados



Círculos: estados

Setas entre estados: define a transição entre dois estados após a leitura do símbolo que está sobre a seta

Círculos duplos: estados finais ou de aceitação

Seta sem início (somente uma): indica quem é o estado inicial

Autômatos finitos

- Autômatos finitos são MAIS que esses controladores, são mecanismos RECONHECEDORES
  - Reconhece quais entradas fazem o autômato sair do estado inicial e chegar em um estado final (ou de aceitação)
- Ex: reconhecer sequências válidas de páginas visitadas em um site de *e-commerce…*

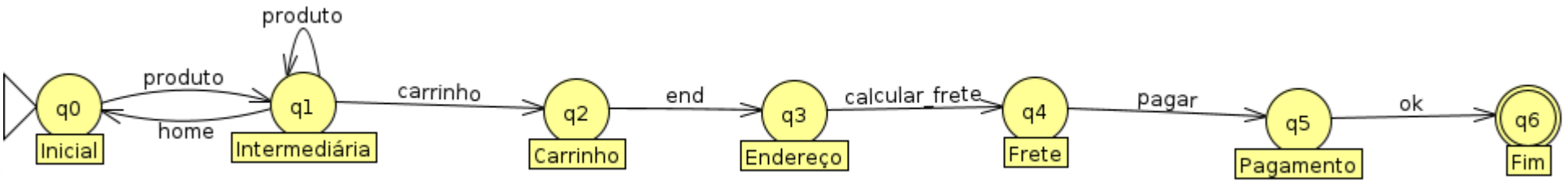
Exemplo 1

- Imagine um site de *e-commerce* em que:
  - Há uma página “**Inicial**” de entrada
  - Várias páginas “**Intermediárias**”
  - as 5 últimas páginas para finalizar a compra são:
    - **Carrinho** (visualização do conteúdo selecionado para compra)
    - **Endereço** (seleção do endereço de entrega)
    - **Frete** (seleção do tipo de envio)
    - **Pagamento** (seleção do tipo de pagamento e pagamento propriamento dito)
    - **Fim** (tela do “Parabéns por ter comprado conosco”)

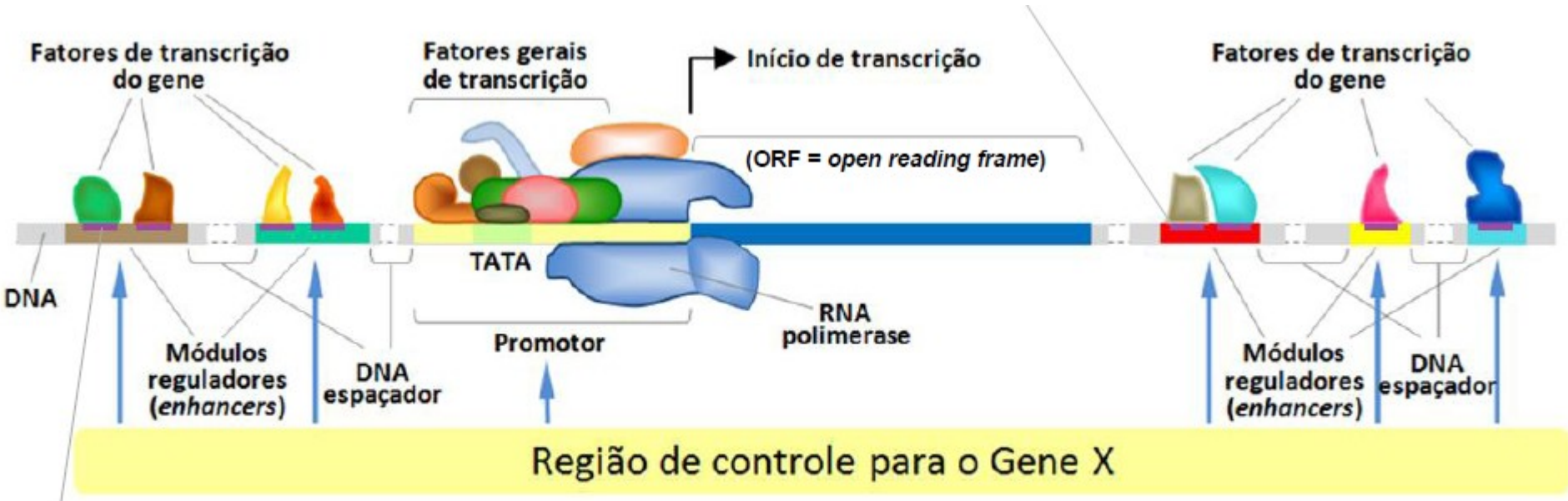
- Poderíamos modelar AF para isso? Como seria o seu “esquema gráfico”?

Exemplo 1

- Imagine um site de *e-commerce* em que:
  - Há uma página “**Inicial**” de entrada
  - Várias páginas “**Intermediárias**”
  - as 5 últimas páginas para finalizar a compra são:
    - **Carrinho** (visualização do conteúdo selecionado para compra)
    - **Endereço** (seleção do endereço de entrega)
    - **Frete** (seleção do tipo de envio)
    - **Pagamento** (seleção do tipo de pagamento e pagamento propriamento dito)
    - **Fim** (tela do “Parabéns por ter comprado conosco”)



Exemplo 2: Bioinformática!
Sítios de ligação de fatores de transcrição

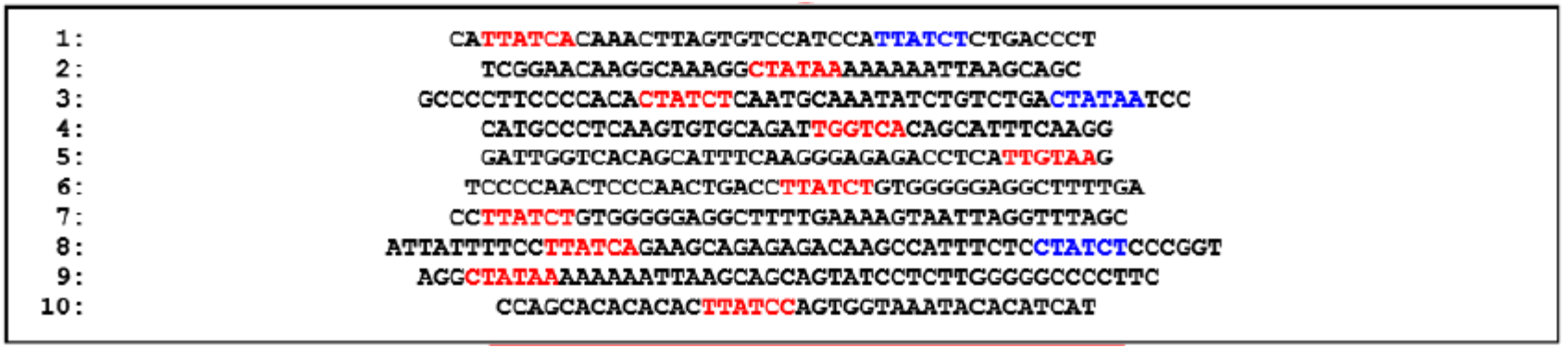


Sítios de Ligação dos Fatores de Transcrição:
Trechos de 5 a 30pbs no DNA, reconhecidos pelos Fatores de Transcrição (▬)

Adaptado de Alberts et al., 2002.

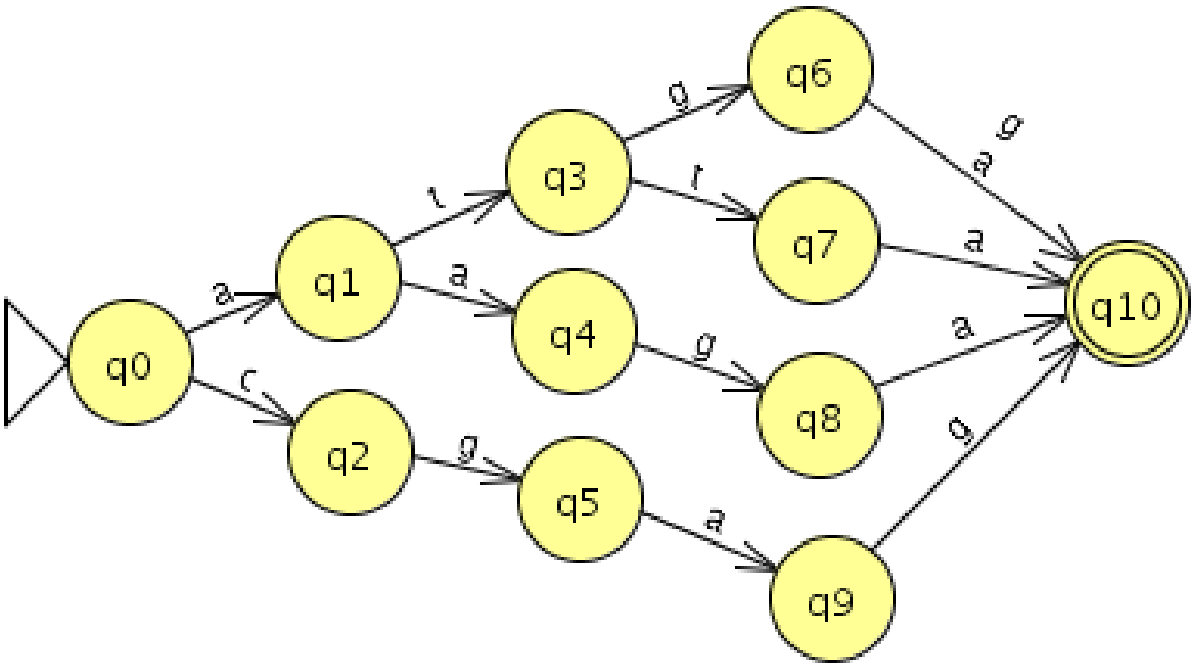
7

Sítios de ligação de UM DADO FT



Vamos imaginar que só essas sequências são possíveis

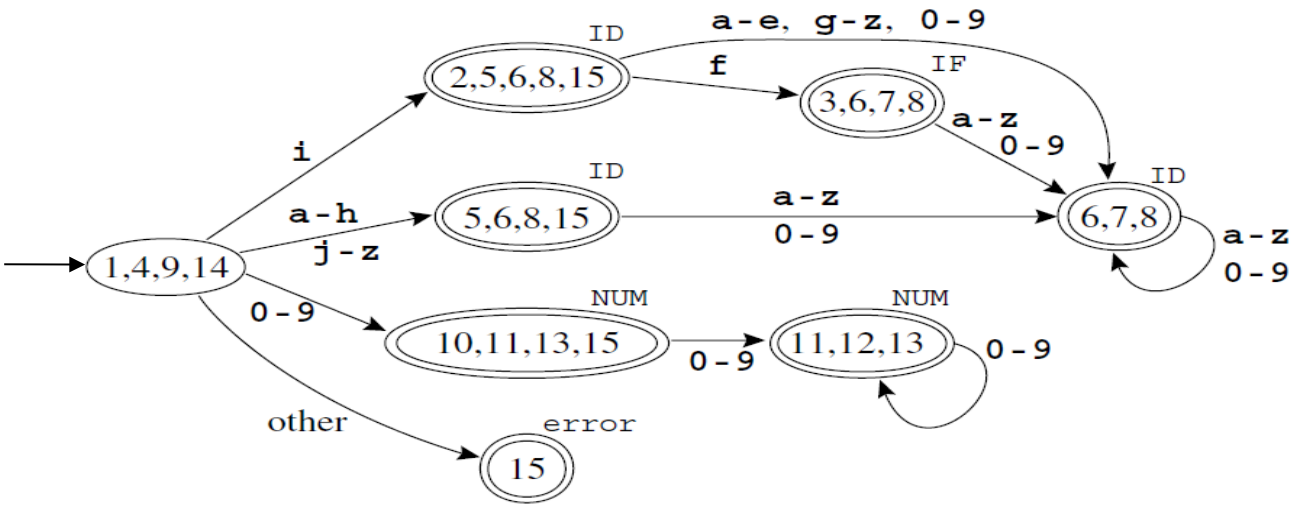
```
atga
atgg
atta
aaga
cgag
```



Fase inicial (análise léxica) de compilação de linguagens de programação

```
if(num1>0){
    if  (num2>0){
        print("Multiply");
    }
    else{
        printf("Divide");
    }
}
else{
    if(num2>0){
        printf("Divide");
    }
    else if(num1>num2){
        printf("Divide");
    }

    else if(num2 == num1){
        printf("END");
    }
```



Poderíamos ficar aqui dias falando de várias aplicações...

Human Activity Discovery and Recognition Using Probabilistic Finite-State Automata

Kévin Viard, Maria Pia Fanti, *Fellow, IEEE*, Gregory Faraut, *Member, IEEE*, and Jean-Jacques Lesage, *Member, IEEE*

***Abstract*— Ambient assisted living and smart home technologies are a good way to take care of dependent people whose number will increase in the future. They allow the discovery and the recognition of human's activities of daily living (ADLs) in order to take care of people by keeping them in their home. In order to consider the human behavior nondeterminism, probabilistic approaches are used despite difficulties encountered in model generation and probabilistic indicators computing. In this article, a global method based on probabilistic finite-state automata and the definition of the normalized likelihood and perplexity is proposed to manage ADLs discovery and recognition. In order to reduce the computational complexity, some results about a simplified normalized likelihood computation are proved. A real case study showing the efficiency of the proposed method is discussed.**

I. Introduction

ACCORDING to several demography studies, the dependence rate of the world population is continuously increasing since 2010. In 2050, the part of the population aged 60 or more will rise to 30% in the majority of countries [1]. This societal evolution is becoming an important human and economic issue for next years. In fact, current health and welfare institutions will not be sufficient to treat this proportion of elderly people. Hence, severe pressure on the public healthcare sector and lack of adequate facilities are driving the way in which health services are delivered to the patients [2], [3]. Therefore, alternative solutions have to be found and rapidly

Modeling and Verification of Hospital Intelligent Diagnosis and Treatment Service Based on Timed Automata in Internet of Things

Lei Yu [1,2,*], Yubo Yan [3], Yang Lu [4], Benhong Zhang [4], Ya Li [1], Fangliang Huang [1] and Yulian Shen [5]

1 School of Medical Information Technology, Anhui University of Chinese Medicine, Hefei 230012, China;
2 Institute of Computer Application in Traditional Chinese Medicine, Anhui Academy of Chinese Medicine, Hefei 230012, China;
3 School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China;
4 School of Computer and Information, Hefei University of Technology, Hefei 230601, China;
5 First affiliated hospital, Anhui University of Chinese Medicine, Hefei 230038, China;
** Corresponding author: fishstonehfut1006@163.com*

***Abstract*—Intelligent diagnosis and treatment service is the core content of hospital Internet of Things. Due to the complexity of the hospital environment, the disregard of the system design leads to the uncertainty of data and the instability of system. Timed automata provides an effective method for formal modeling of Internet of Things systems. Therefore, based on the theory of timed automata, introducing the related concepts such as environmental entities, the interaction between Internet of Things services and external**

ensure its correctness and reliability, it is very necessary to conduct sufficient testing and inspection before the system is deployed. This has become a critical problem that the Internet of Things project needs to solve urgently. Formalization method [4] is an effective method to verify the system. Due to its low cost and short cycle, it provides better support for the pre-deployment inspection of the Internet of Things system.

Service modeling is the core content of Internet of

Physica A 517 (2019) 1–12

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

High-order Hidden Markov Model for trend prediction in financial time series

Mengqi Zhang [a,b], Xin Jiang [a,b,*], Zehua Fang [b], Yue Zeng [b], Ke Xu [a,c]

[a] Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), China
[b] LMIB & School of Mathematics and Systems Science, Beihang University, 100191, Beijing, China
[c] School of Computer Science and Engineering, Beihang University, 100191, Beijing, China

HIGHLIGHTS

- We give a detailed description of some financial indicators used in evaluating the performance of the trading strategies.
- We demonstrate how the dynamic trading strategy works in each time window and generate the trading signals according to the observation sequence.

 ISBN: 978-979-1421-19-5

Stochastic Regular Grammar-based Learning for Basic Dance Motion Recognition

Yaya Heryadi[1]
[1]*School of Computer Science*
Binus International – Binus University
Jakarta, Indonesia
Email: [1]yayaheryadi@binus.edu

Mohamad Ivan Fanany[2], Aniati Murni Arymurthy[3]
[2,3]*Fakultas Ilmu Komputer*
Universitas Indonesia
Depok, Indonesia
Email: [2]ivan@cs.ui.ac.id, [3]aniati@cs.ui.ac.id

***Abstract*—this paper presents a simple and computationally efficient framework for 3D dance basic motion recognition based on syntactic pattern recognition. In this research, a class of basic dance motions is modeled by a stochastic regular grammar (SRG), inferred from training dataset, in which key body poses that are learned from training dataset are selected as gesture primitives. To represent a dance motion, body pose of a dancer is**

the National Synergy Research Grant SINas, contract No. RT-2012-1117 and No. KP-2013-1679

The challenges in solving this problem, however, are multifold due to complexity of human motions thanks to the flexibility of human body, the spatial and temporal variations exhibited due to differences in duration of different actions performed, and the changing spatial characteristics of the human form in performing each

2019 5th International Conference for Convergence in Technology (I2CT)
Pune, India. Mar 29-31, 2019

Instant Bi-Lingual Captions

Mahesh Dhumal, Hemant Kumar Kushwaha, Vaibhav Gupta, Sona R. Pawara
Department of Computer Engineering
STES's Sinhgad Academy of Engineering
Pune, Maharashtra, India
mahesh90010@gmail.com, hkushwaha345@gmail.com, vaibhavgupta710@gmail.com, srpawara.sae@sinhgad.edu

***Abstract* — With rapid development of the internet, the tremendous growth has been observed in video sharing platform. Millions of users have been using this platform as source of knowledge, entertainment, self-development, but language barrier has confined the reach of this platform to certain amount of population. Also the hearing-impaired people could not make satisfactory use of this platform. Subtitles play important role in such scenarios, with the help of which people can understand the content of the video. Various applications have been developed for subtitle generation to resolve issue of language barrier and hearing impairment. In this paper we have surveyed different methods for speech recognition and bilingual subtitle generation and proposed a system for instant bi-lingual subtitle generation based on MFCC and HMM.**

will generate the subtitles in the language specified by user without any delay.

This paper is bases on the comparisons between various algorithms to understand merits and demerits of each and develop an application providing higher accuracy in results. The survey done in this paper composed of the researches done by various analysts, their methodologies, theoretical formulae and the conclusions they have arrived at.

II. LITERATURE SURVEY

For recognition of speech signal and generation of subtitle Su Myat Mon and Hla Myo Tun in 2015 [1], implemented a Speech-to-Text conversion system using MFCC for feature extraction and HMM as the recognizers.

Autômatos finitos

- Para não nos preocupar com detalhes de negócio, vamos trabalhar apenas com exemplos “simples”
- Ex: reconhecer strings binárias (compostas por 0’s e 1’s) que terminem com 1, com tamanho pelo menos 1
  - 1, 01, 11, 001, 00000001, 1101111, ...

Círculos: estados

Círculos duplos: estados finais ou de aceitação

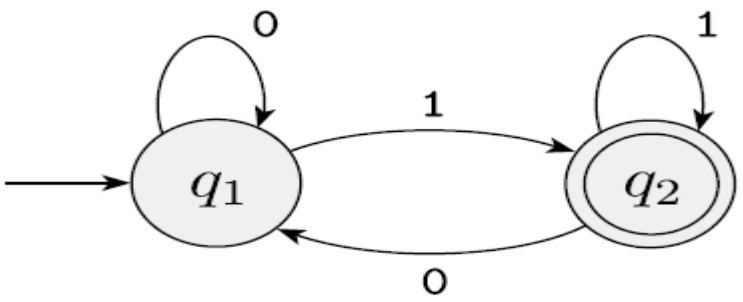
Seta sem início (somente uma): indica quem é o estado inicial

Setas entre estados: define a transição entre dois estados após a leitura do símbolo que está sobre a seta

- Ex: reconhecer strings binárias (compostas por 0's e 1's) que terminem com 1, com tamanho pelo menos 1
  - 1, 01, 11, 001, 00000001, 1101111, ...

Diagrama de estados:

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

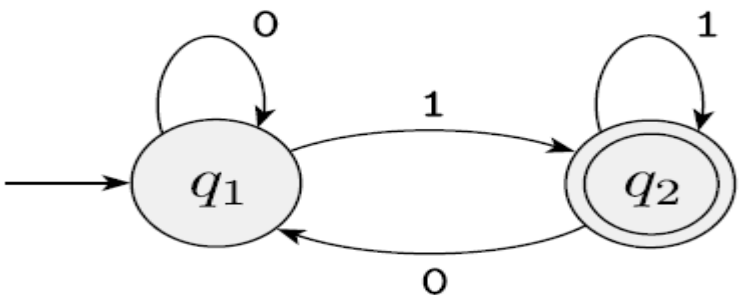
Exemplos de cadeias w aceitas pelo autômato M1:
1, 01, 11, 001, 00000001, 1101111, …

Exemplos de cadeias w NÃO aceitas pelo autômato M1:
0, 10, 110, 000, 0001010, 1111110, ε, ... ← Cadeia vazia ("")

Autômatos finitos

- Diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

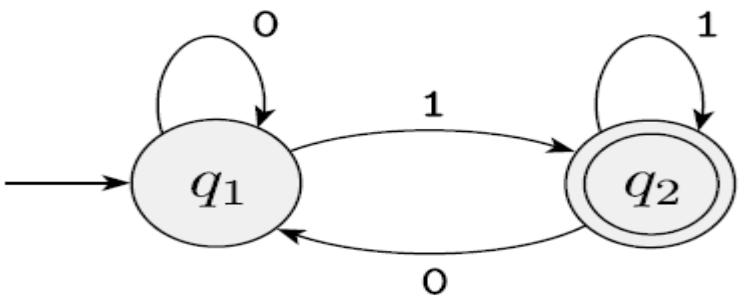
Que tipo de cadeia esse autômato M1 aceita?

Sequência binária (de tamanho >= 1) que termine em 1

Autômatos finitos

- Diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

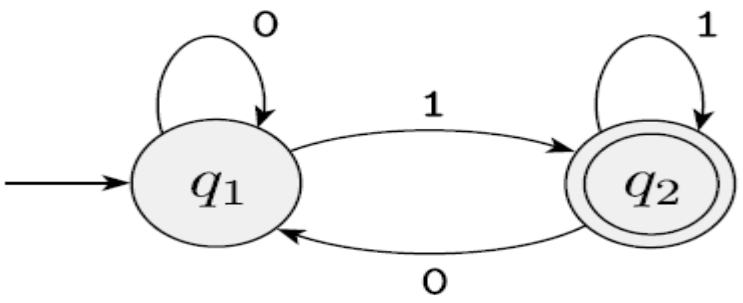
Que tipo de cadeia esse autômato M1 aceita?

Sequência binária (de tamanho >= 1) que termine em 1

Autômatos finitos

- Diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

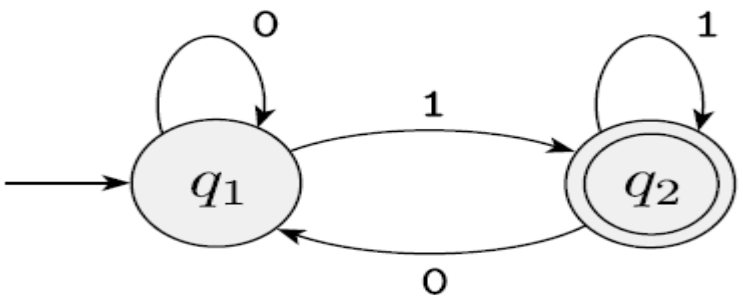
Qual o CONJUNTO de cadeias esse autômato M1 reconhece?

O conjunto de todas as sequência binárias (de tamanho >= 1) que terminam em 1

Autômatos finitos

- Diagrama de estados

M1



Processo de reconhecimento:

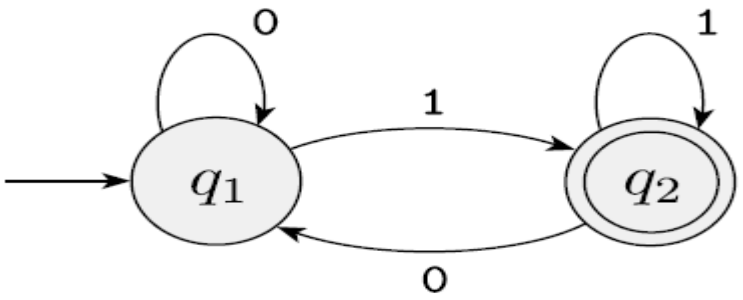
Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

Qual a LINGUAGEM que esse autômato M1 reconhece?

O conjunto de todas as sequência binárias (de tamanho >= 1) que terminam em 1

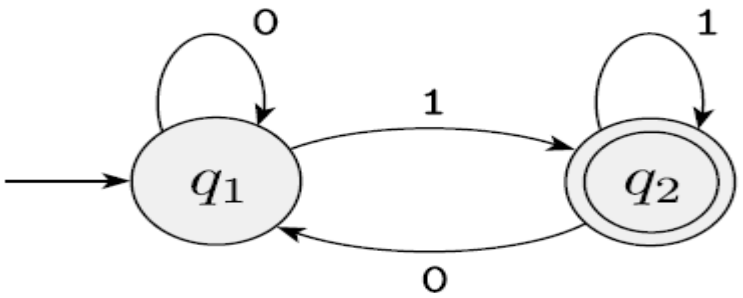
Alfabeto



Um **alfabeto** Σ é um conjunto de símbolos (que aparecem nas cadeias de interesse…)

- Σ = ?

Alfabeto



Um **alfabeto** Σ é um conjunto de símbolos (que aparecem nas cadeias de interesse…)

- Σ = {0, 1}, portanto Σ* é o conjunto de **todas** as sequências binárias

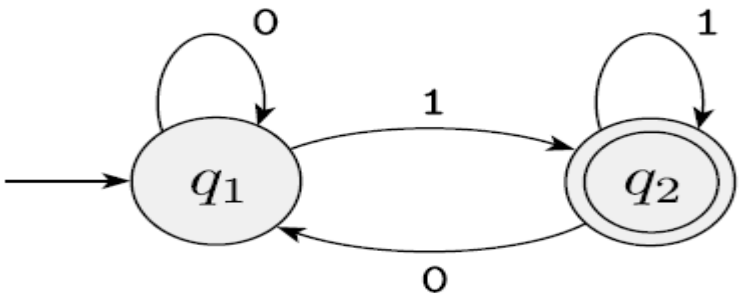
Σ = Σ^1: um símbolo de Σ (ex: 0, 1)

Σ^2: concatenação de dois símbolos (quaisquer) de Σ (ex: 00, 01, 10, 11)

Σ^*: concatenação de 0 ou mais símbolos (quaisquer) de Σ (ex: ε, 1, 00, 101, 0011….)

Operação * : **fecho de Kleene**

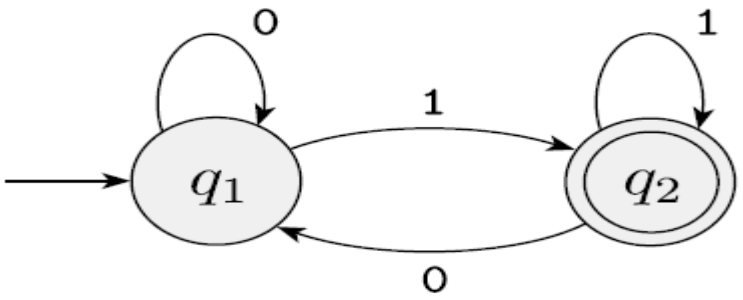
Cadeia



Uma **cadeia** w é uma sequência de símbolos de um alfabeto Σ, incluindo a **cadeia vazia ε** (w ϵ Σ*)

- Σ = {0, 1}, portanto Σ* é o conjunto de **todas** as sequências binárias
- w ?

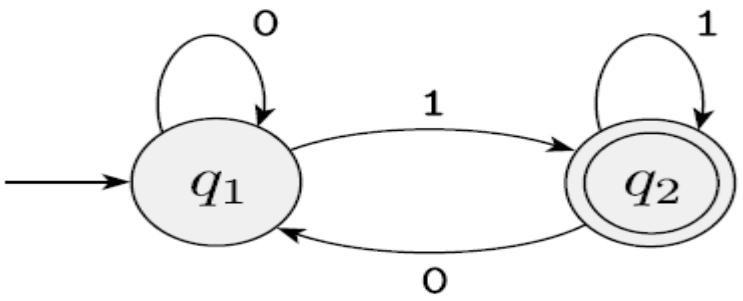
Cadeia



Uma **cadeia** w é uma sequência de símbolos de um alfabeto Σ, incluindo a **cadeia vazia ε** (w ϵ Σ*)

- Σ = {0, 1}, portanto Σ* é o conjunto de **todas** as sequências binárias
- w é a cadeia vazia ou uma sequência binária (de 0's e 1's)

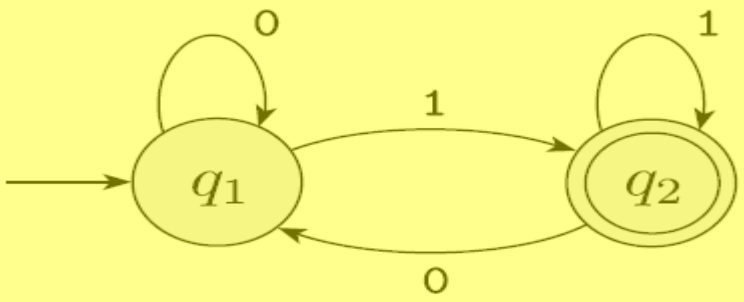
Linguagem



Uma **linguagem** L é um **conjunto de cadeias** sobre um alfabeto Σ (L é subconjunto de Σ*)

- Σ = {0, 1}, portanto Σ* é o conjunto de **todas** as sequências binárias
- w é a cadeia vazia ou uma sequência binária (de 0's e 1's)
- Uma linguagem L pode ser o conjunto vazio ou um conjunto de sequências binárias

Linguagem



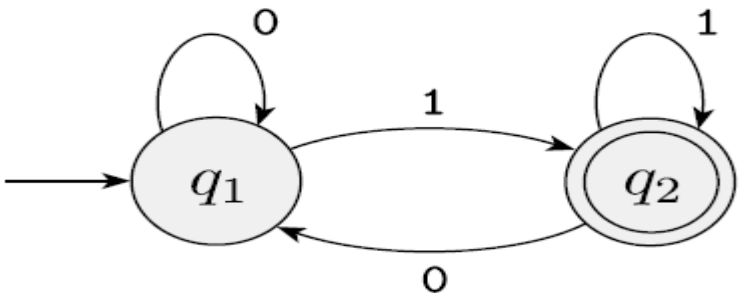
Uma **linguagem** L é um **conjunto de cadeias** sobre um alfabeto Σ (L é subconjunto de Σ*)

- Σ = {0, 1}, portanto Σ* é o conjunto de **todas** as sequências binárias
- w é a cadeia vazia ou uma sequência binária (de 0's e 1's)
- Uma linguagem L pode ser o conjunto vazio ou um conjunto de sequências binárias

Uma **linguagem** L é um **conjunto de cadeias**

sobre um alfabeto Σ (L é subconjunto de Σ*)

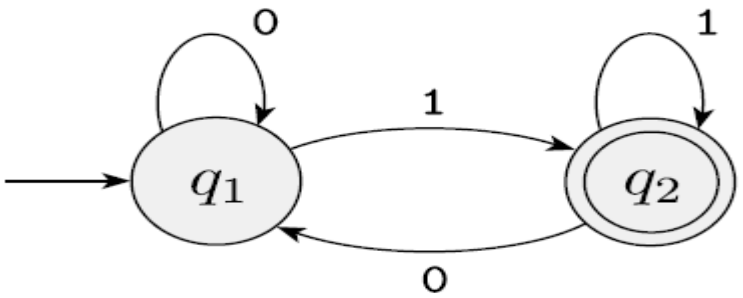
Linguagem reconhecida por um autômato



A **linguagem** L reconhecida por um autômato M (denotado L(M)) é o conjunto das **cadeias** (de símbolos de entrada) aceitas pelo autômato

- Σ = {0, 1}
- w é a cadeia vazia ou uma sequência binária (de 0's e 1's)
- L é o conjunto vazio ou um conjunto de sequências binárias
- L(M1) = ?

Linguagem reconhecida por um autômato



A **linguagem** L reconhecida por um autômato M (denotado L(M)) é o conjunto das **cadeias** (de símbolos de entrada) aceitas pelo autômato

$\Sigma = \{0, 1\}$

- w é a cadeia vazia ou uma sequência binária (de 0's e 1's)
- Uma **linguagem** L sobre Σ é o **conjunto** vazio ou um **conjunto** de sequências binárias
- L(M1) = é o conjunto de todas as sequências binárias que terminam com 1, ou formalmente: $L(M1) = \{ w \in \{0, 1\}^* \mid w \text{ termina com } 1\}$

Resumo

- Um **alfabeto** Σ é um conjunto de símbolos
- Uma **cadeia** w é uma sequência (concatenação) de símbolos de um alfabeto Σ, incluindo a **cadeia vazia ε** (w ϵ Σ*)
- Uma **linguagem** L é **um conjunto de cadeias** sobre um alfabeto Σ (L é subconjunto de Σ*)
- A **linguagem** L reconhecida por um autômato M (denotado L(M)) é o conjunto das **cadeias** (de símbolos de entrada) aceitas pelo autômato

Exercício

Projete um autômato que reconheça cadeias binárias (compostas por 0’s e 1’s) que comecem e terminem com zero, com tamanho pelo menos 1

0, 00, 010, 000000, 0101110, ...