Teoria da Computação

Universidade Federal de Campina Grande – UFCG
Centro de Engenharia Elétrica e Informática – CEEI
Departamento de Sistemas e Computação – DSC
Professor: Andrey Brito Período: 2023.2

O que é Teoria da Computação?

- Teoria
  - Estudo matemático, formal, baseado em provas (positivas ou negativas)
  - Geral, independente de tecnologia
  - Abstrato, ignora detalhes “secundários” (assim como programação ignora eletrônica)
- Computação
  - Execução de um processo que segue um modelo bem definido
  - Ato de calcular

Para que serve?

- O que pode ser feito por um computador?
  - Independente de linguagem
  - Independente de desempenho, velocidade
- No dia-a-dia pensar na solução de um problema significa pensar em uma linguagem, uma biblioteca, uma máquina
- Em teoria da computação, transformamos isso em...
  - É possível resolver o problema (de forma “automatizável”)?
  - É viável? Precisamos de quanto espaço ou quanto tempo?
  - Abstrair detalhes é útil para olhar para sistemas de um jeito diferente

Para que serve? (2)

- E se não for solúvel? Ou não for viável?
  - Modificar o problema
  - Procurar respostas aproximadas
  - Ter um problema difícil pode ser o alvo (ex., blockchain, criptografia)

- Embora o grande objetivo seja “entender os limites da computação”, existem muitas aplicações de conceitos vistos aqui...
  - Modelar um pedaço de hardware ou uma comunicação (protocolos)
  - Descrever padrões (ex., para procurar strings ou validar entradas)
  - Descrever o que é válido em uma linguagem (ex., em compiladores)

Então... o que é Teoria da Computação?

- Estudo de propriedades fundamentais dos computadores

- Como assim? O que poderia ser ignorado?
  - Consideramos uma “máquina que reage à uma entrada”
  - Sem distinção hardware/software/aplicações
  - E retirando todos os “detalhes tecnológicos” de um computador (inclusive o tipo de resultado de uma computação)
    - Problemas de decisão
    - Verdadeiro/Falso, Sim/Não

- Objetivo: entender as limitações dos computadores
  - TC é frequentemente estudada de 3 perspectivas...

Complexidade

- Alguns problemas são mais difíceis que outros: máximo-valor, ordenação, escalonamento, fatoração, caixeiro viajante
  - Você consegue ordenar milhões de números rapidamente (100 TB em 1 minuto)
  - Mas levaria décadas para achar o par de uma chave assimétrica de 256 bytes

Complexidade (2)

- Dois algoritmos para solução de sistemas de equações
  - Cramer (determinantes)
  - Gauss (escalonamento de matrizes)

http://www.decom.ufop.br/moreira/site_media/uploads/arquivos/01_selas_new.pdf
http://wiki.icmc.usp.br/images/d/de/Analise_complexidade.pdf

Complexidade (3)

- Dois algoritmos para solução de sistemas de equações
  - Cramer (determinantes)
  - Gauss (escalonamento de matrizes)
- O tempo de solução depende da ordem da matriz (número de equações) e da velocidade do processador (assumindo 3 GHz)

Ordem	Tempo - Cramer	Tempo - Gauss
2	4 ns	2 ns
3	12 ns	8 ns
4	48 ns	20 ns
5	240 ns	40 ns
10	7.3 ms	330 ns
20	152 anos	2.7 ms

http://www.decom.ufop.br/moreira/site_media/uploads/arquivos/01_selas_new.pdf
http://wiki.icmc.usp.br/images/d/de/Analise_complexidade.pdf

Complexidade (4) – Outro exemplo

Execution times of a machine that executes 10^9 steps by second (~ 1 GHz), as a function of the algorithm cost and the size of input n:

Size	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
10	3.322 ns	10 ns	33 ns	100 ns	1 μs	1 μs
20	4.322 ns	20 ns	86 ns	400 ns	8 μs	1 ms
30	4.907 ns	30 ns	147 ns	900 ns	27 μs	1 s
40	5.322 ns	40 ns	213 ns	2 μs	64 μs	18.3 min
50	5.644 ns	50 ns	282 ns	3 μs	125 μs	13 days
100	6.644 ns	100 ns	664 ns	10 μs	1 ms	$40 \cdot 10^{12}$ years
1000	10 ns	1 μs	10 μs	1 ms	1 s	
10000	13 ns	10 μs	133 μs	100 ms	16.7 min	
100000	17 ns	100 μs	2 ms	10 s	11.6 days	
1000000	20 ns	1 ms	20 ms	16.7 min	31.7 years	

http://wiki.icmc.usp.br/images/d/de/Analise_complexidade.pdf

Complexidade (5)

- Em que ajuda saber que um problema é difícil?
  - Isolar qual a dificuldade real e assim mudar levemente o problema pode ajudar
  - Uma solução ótima pode não ser necessária, pior caso pode ser bem diferente do caso comum
  - E, de novo: você pode também estar a procura de problemas difíceis

Solubilidade

- Resolver um problema pode ser impossível
  - “Decidibilidade” vs. “Indecibilidade”
  - (Resposta garantida ou não)

- Quais problemas podem ser resolvidos por um computador?
  - Nem sempre existe uma solução em algoritmos
  - Exemplo de problema indecidível (ou insolúvel)
    - Problema da parada: ver se um programa termina (ex., não “trava”)
    - Construir um algoritmo que, dado um programa, avalie se ele pode alcançar um ponto onde fica preso em um loop infinito...
    - ... Ou se ele termina para qualquer entrada.

Teoria dos autômatos

- Tudo começa simplificando o ambiente, veremos vários “modelos computacionais”
- Um modelo tem um conjunto de capacidades
  - Quão “robustos” eles são em relação às capacidades? Ele perde/ganha poder?
  - Que tipos de problemas eles conseguem resolver?
- 3 modelos principais de máquinas: autômatos finitos, autômatos de pilha, máquinas de Turing
  - Começar com o modelo mais simples possível (e ainda útil)
  - Terminar com um modelo “completo” (mesmo ele ainda é muito simples)
  - Neste modelo, fica mais razoável de ver que alguns problemas não têm solução...
  - ... e que essas máquinas e programas estão relacionados

Teoria da computação aqui

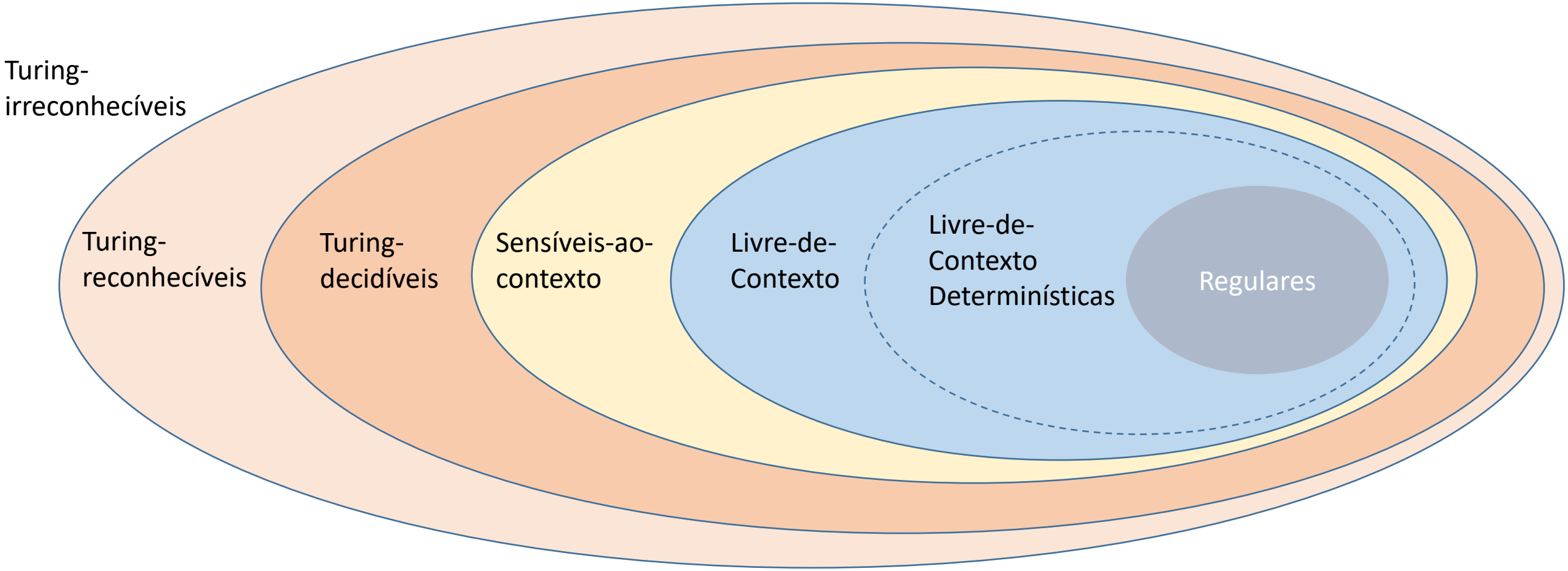
- Entender as limitações das máquinas

Teoria da computação aqui

- Entender as limitações das máquinas

- É computacionalmente possível verificar...?
  - Se uma cadeia binária tem um número par de dígitos “1”? (checksum)
  - Se um programa Java é válido? (compilador)
  - Se um programa Java pode entrar em loop? (testador)

Hierarquia de problemas



Funcionamento da disciplina

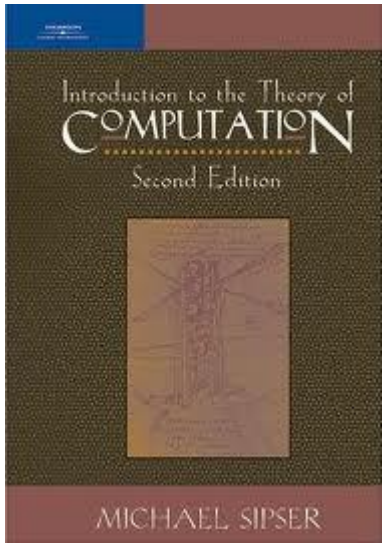
Diversos

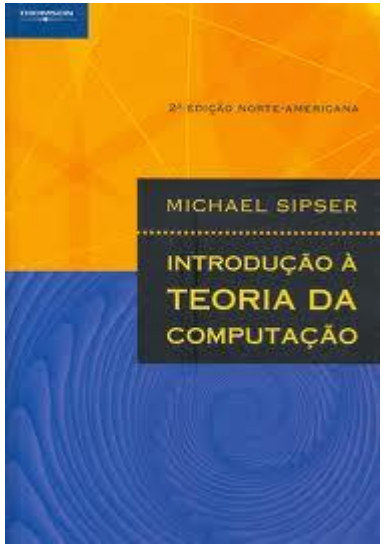
- Comunicação: PVAE (Moodle@UFCG) + Controle Acadêmico
- Avaliação: 3 provas (e uma reposição)
- Estudos dirigidos
  - Compensação de pontos facultativos ou outros eventos extraordinários
  - Exercícios ou resumos de assuntos que não serão priorizados em sala

Material

- PVAE: slides, anúncios, links, plano de aulas, listas de exercícios

- Bibliografia
  - **Introdução à teoria da computação, Michael Sipser (tradução)**
  - Introduction to the Theory of Computation, Michael Sipser
  - Introdução à Teoria de Autômatos, Linguagens e Computação, John Hopcroft, Jeffrey Ullman, Rajeev Motwani
  - Introdução aos fundamentos da Computação, Newton José Vieira.





Autômatos finitos

Começando...

- Como começar a entender os limites do que os computadores podem fazer?

Começando…

- Como começar a entender os limites do que os computadores podem fazer?
  - Vamos considerar um computador “bem” simplificado
  - Tirar o máximo possível de detalhes
  - E aí vemos que tipo de problemas conseguimos resolver

Teoria dos autômatos

- “Máquina ou engenho composto de mecanismo que lhe imprime determinados movimentos (p.ex., um relógio, certos tipos de brinquedo etc.)”

  (Houaiss, http://houaiss.uol.com.br)

- “Automaton: *a machine or control mechanism designed to follow automatically a predetermined sequence of operations or respond to encoded instructions*”

  (Merriam-Webster, www.m-w.com)

- Estudo de máquinas abstratas simples e o que pode ser processado por elas

Teoria dos autômatos

Você pode olhar para um autômato como uma máquina mecânica ou como um programa escrito em uma linguagem de programação sem muitos recursos, não precisamos fazer essa distinção.

- "Máquina ou engenho composto de mecanismo que lhe imprime determinados movimentos (p.ex., um relógio, certos tipos de brinquedo etc.)"

  (Houaiss, http://houaiss.uol.com.br)

- "Automaton: *a machine or control mechanism designed to follow automatically a predetermined sequence of operations or respond to encoded instructions*"

  (Merriam-Webster, www.m-w.com)

- Estudo de máquinas abstratas simples e o que pode ser processado por elas

Teoria de autômatos finitos

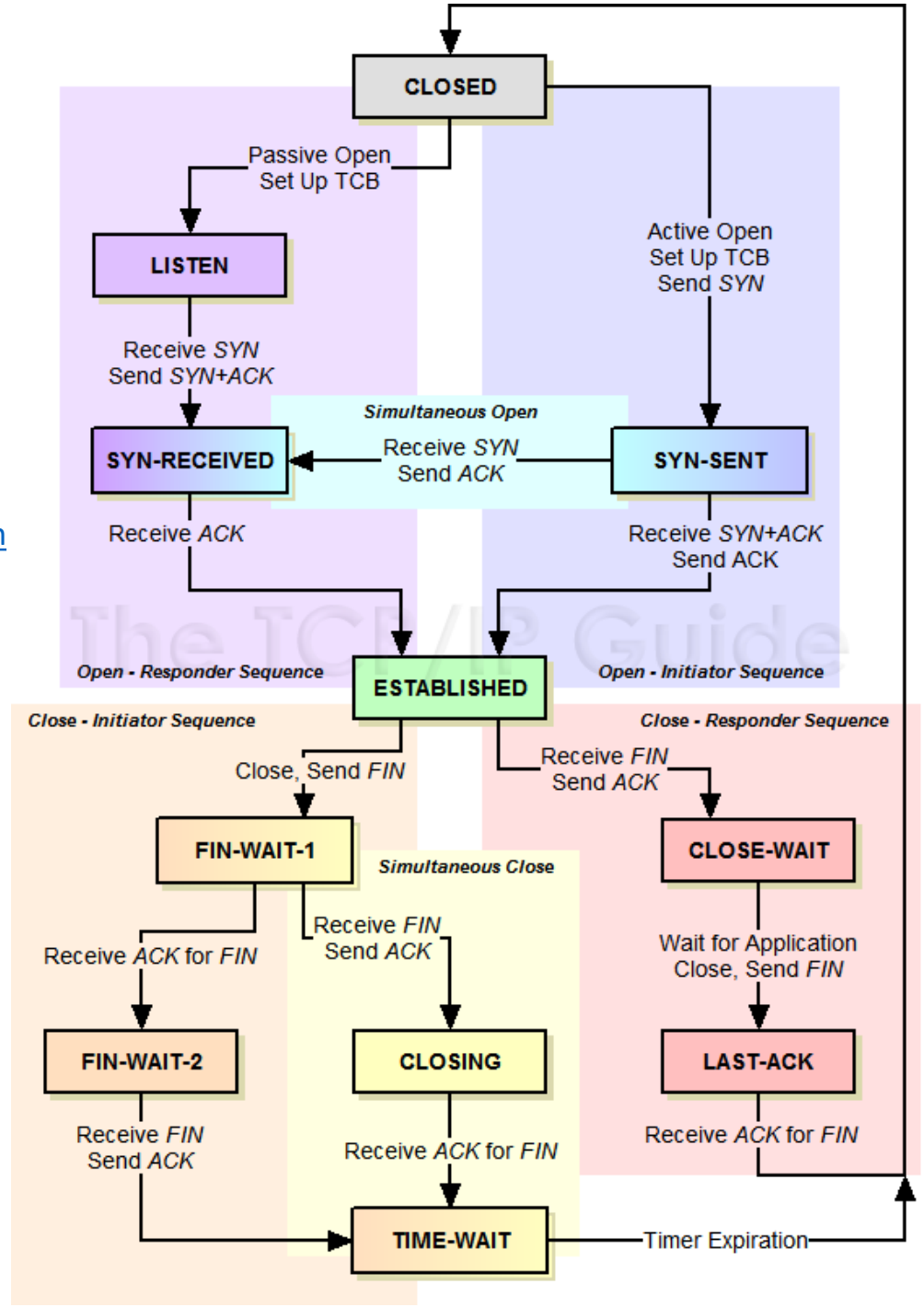
- Computador vs. modelo computacional
  - Computador: máquina (real) que realiza computações
  - Computação: execução de um processo, ato de calcular
  - Sistema real: complexo, muitas variáveis, aspectos fora do controle
  - Modelo: máquina (abstrata), simplificado, realista em alguns aspectos, irrealista em outros, sob controle
- Outro exemplo de simplificação: foco problemas de decisão
- Um modelo simples para um computador: máquina de estados finitos

Onde máquinas de estado são usadas?

- Computadores embarcados simples (eletrodomésticos, sistemas de automação)
- Protocolos de comunicação
- Detecção e definição de padrões

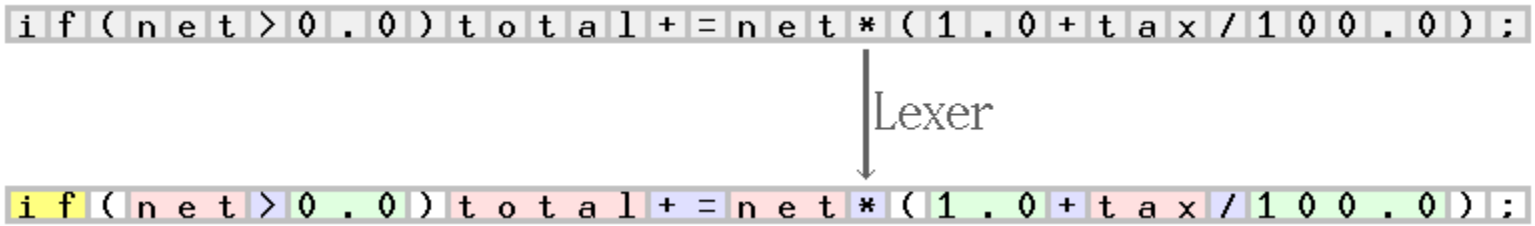
Protocolos de comunicação

Fonte: http://tcpipguide.com/free/t_TCPOperationalOverviewandtheTCPFiniteStateMachineF-2.htm



Analisador léxico

- Parte de um compilador
- Quebra palavras considerando os separadores



E como funcionam?

- Apenas reagem às entradas

- Lembram apenas de como funcionam
  - Sua especificação ou “código”
  - E, durante a execução, lembram em que posição atual dentro desta especificação

Nossa primeira máquina

- Um interruptor eletrônico
  - Como o de uma TV ou eletrodoméstico mais sofisticado
  - Não existe um contato mecânico permanente, ele “lembra” qual a sua situação
  - O interruptor recebe um pulso e alterna entre ligado e desligado
- Uma máquina de estado: recebe entradas e reage a estas entradas de alguma forma
- Já ouviram falar de circuitos sequenciais? (ex., em circuitos digitais, como flip-flops)

Nossa primeira máquina (2)

- O que é a entrada da nossa máquina?

- O que ela precisa lembrar?

Nossa primeira máquina (2)

- O que é a entrada da nossa máquina?

- O que ela precisa lembrar?
  - Quantas vezes o botão foi apertado?

Nossa primeira máquina (2)

- O que é a entrada da nossa máquina?
  - Uma sequência de comandos “BOTÃO-PRESSIONADO”

- O que ela precisa lembrar?
  - Quantas vezes o botão foi apertado? Não.
  - Basta lembrar se o interruptor está ligado ou desligado
  - (Se o botão foi apertado um número par ou ímpar de vezes)

- A máquina é representada com um **diagrama de estados**, que deve rastrear a informação que deve ser lembrada

Estado

- “Qual a situação atual?”
- Porção relevante da história da máquina
  - Ou seja, o que ela de fato precisa lembrar para resolver o problema
- Máquina de estados finita → memória limitada aos seus estados
  - Lembrar de quantas vezes o botão do interruptor foi pressionado precisa de uma quantidade **ilimitada** de memória, ou seja, infinitos estados
  - Lembrar se está **ligado ou desligado** exige apenas dois estados

Estado inicial

- Onde o sistema começa
- O interruptor começa no estado “desligado”

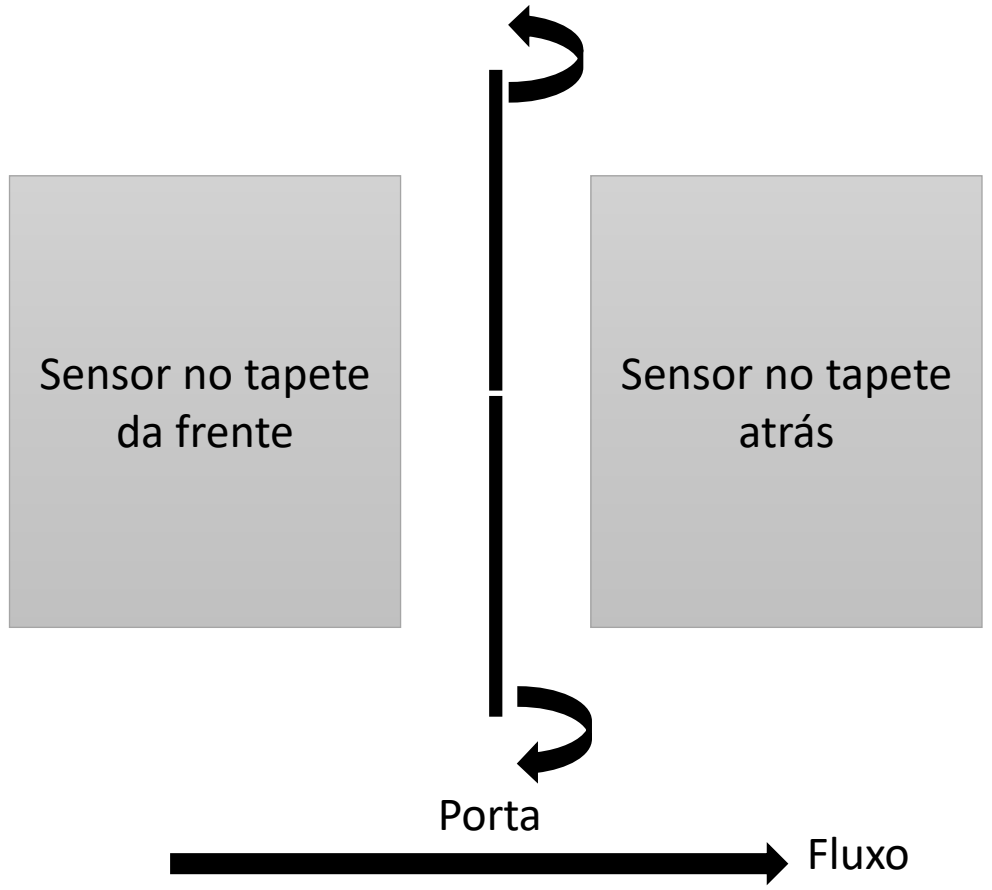
Entradas – Interruptor

- O problema: como estará o interruptor depois de um conjunto de comandos?
  - Tipo de entrada: BOTÃO-PRESSIONADO (P)
  - Exemplo de entrada: PPPPP
  - Resultado depois da entrada acima: estado “ligado”

- Formalmente:
  - Alfabeto de entrada: tipos de “unidades” que podem aparecer como parte de uma entrada (aqui: P)
  - Cada unidade é chamada de símbolo
  - Palavra/cadeia de entrada: uma entrada a ser processada (aqui: PPPPP)

Outro exemplo: Porta automática

- Movimento em uma só direção
- Sensores lidos periodicamente
- Outra forma de descrever o diagrama de estados: tabela de transições de estado

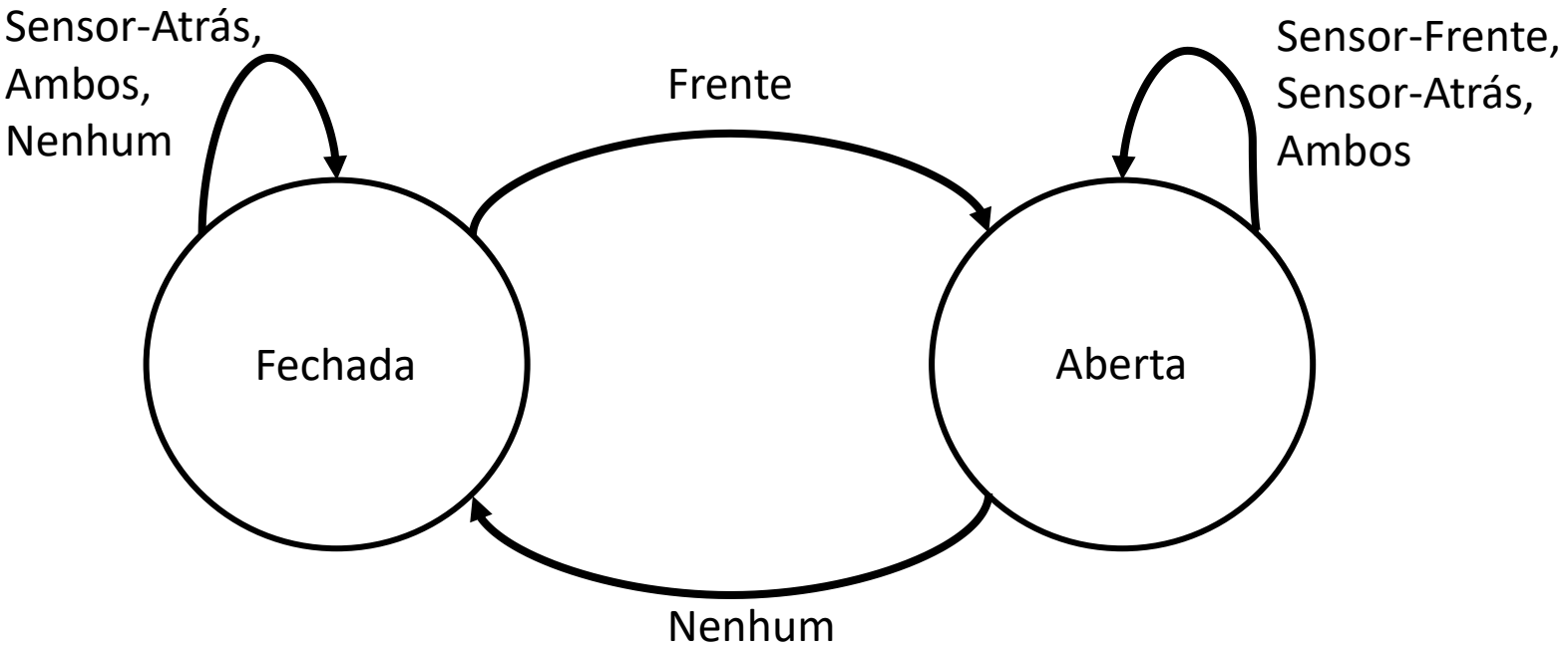


Outro exemplo: Porta automática

- Como é o diagrama de estados?

Outro exemplo: Porta automática

- Como é o diagrama de estados?



Outro exemplo: Porta automática

- Outra forma de descrever o diagrama de estados: tabela de transições de estado

	Nenhum	Frente	Atrás	Ambos
FECHADA	Fechada	Aberta	Fechada	Fechada
ABERTA	Fechada	Aberta	Aberta	Aberta

Outro exemplo: Contadora de pontos

- Suponha um esporte onde dois jogadores se enfrentam e cada jogo tem pelo menos 3 pontos

- Um set acaba quando um dos jogadores faz 3 pontos...
  - ... a não ser quando os jogadores empatam em 2 a 2
  - Neste caso, o jogo continua até que um deles faça dois pontos seguidos (como em um jogo de tênis): “iguais”, “vantagem para A”, “dois seguidos para B”

- Como seria uma máquina de estados que acompanha a pontuação?

A máquina contadora de pontos

- O que a máquina de estados precisa lembrar?
- Onde ela começa?
- O que ela recebe como entrada? Como ela muda de estado?

A máquina contadora de pontos

- O que a máquina de estados precisa lembrar?
  - A pontuação de cada jogador, p. ex.: 0-0, 1-0, 2-0, 2-1, 3-0
- Onde ela começa?
  - No estado “0-0”
- O que ela recebe como entrada? Como ela muda de estado?
  - A entrada é quem fez o ponto, p.ex., jogador 1 ou jogador 2
  - Ela precisa considerar o pontuador para atualizar o estado

Máquina de estados como um autômato

- Nosso objetivo não é só rastrear uma execução, mas sim implementar soluções para problemas de decisão
- Um autômato finito pode ser descrito como uma máquina de estados que diz sim ou não
  - Processa a cadeia de entrada, gerando uma resposta no final
  - Estado final: marca onde a máquina precisa terminar para que a resposta seja sim
- Como a máquina contadora de pontos pode ser transformada em uma “validadora de placar”?
  - Deve responder sim se é um placar válido e não caso contrário