Fundamentos Teóricos da Computação

CIÊNCIA DA COMPUTAÇÃO

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Sumário

* Máquinas de Turing
  * Introdução Informal
  * Definição Formal
    * Exemplos
* Hierarquia de Chomsky
* Decidibilidade
  * Introdução
  * Tese de Church-Turing
  * O Problema da Parada

Máquinas de Turing

* **Máquinas de Turing**
  * **Introdução Informal**

Limitações dos AFs e dos APs

* AFs e APs, apesar de muito importantes do ponto de vista prático, possuem limitações quanto às linguagens que podem ser reconhecidas
  * Algumas linguagens relativamente simples não podem ser reconhecidas por AFs ou APs, como:

  * $\{ xx \mid x \in \{a,b\}^* \}$
  * $\{ a^n b^n c^n \mid n >= 0 \}$
  * $\{ a^n b^k c^n d^k \mid n,k >= 0 \}$

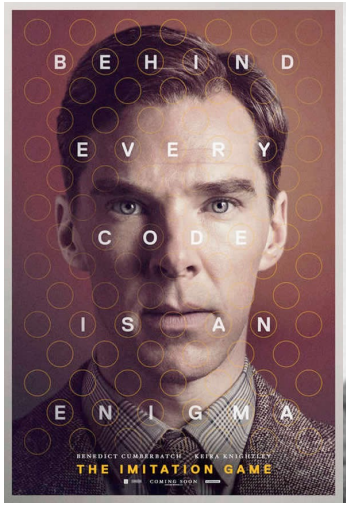
Observação

* Classe de máquinas proposta em torno de 1930
* Nenhum outro tipo de máquina proposta posteriormente tem maior poder computacional
  * Mesmo se considerados os computadores atuais!

* Foi a máquina que concebeu o estudo da computabilidade, criada pelo matemático inglês Alan Mathison Turing (1912-1954)

Curiosidade

* Alan Mathison Turing (1912-1954)
* Sugestão:
  * Veja o filme “The Imitation Game” (2014),
  * “O Jogo da Imitação”:
    * http://www.imdb.com/title/tt2084970/
  * Conheça o “Prêmio Alan Turing”:
    * http://amturing.acm.org/
    * Considerado o prêmio Nobel da Computação, dificilmente é dado à alguém
  * Conheça o “Teste de Turing”:
    * https://goo.gl/pp2T6r



Máquina de Turing

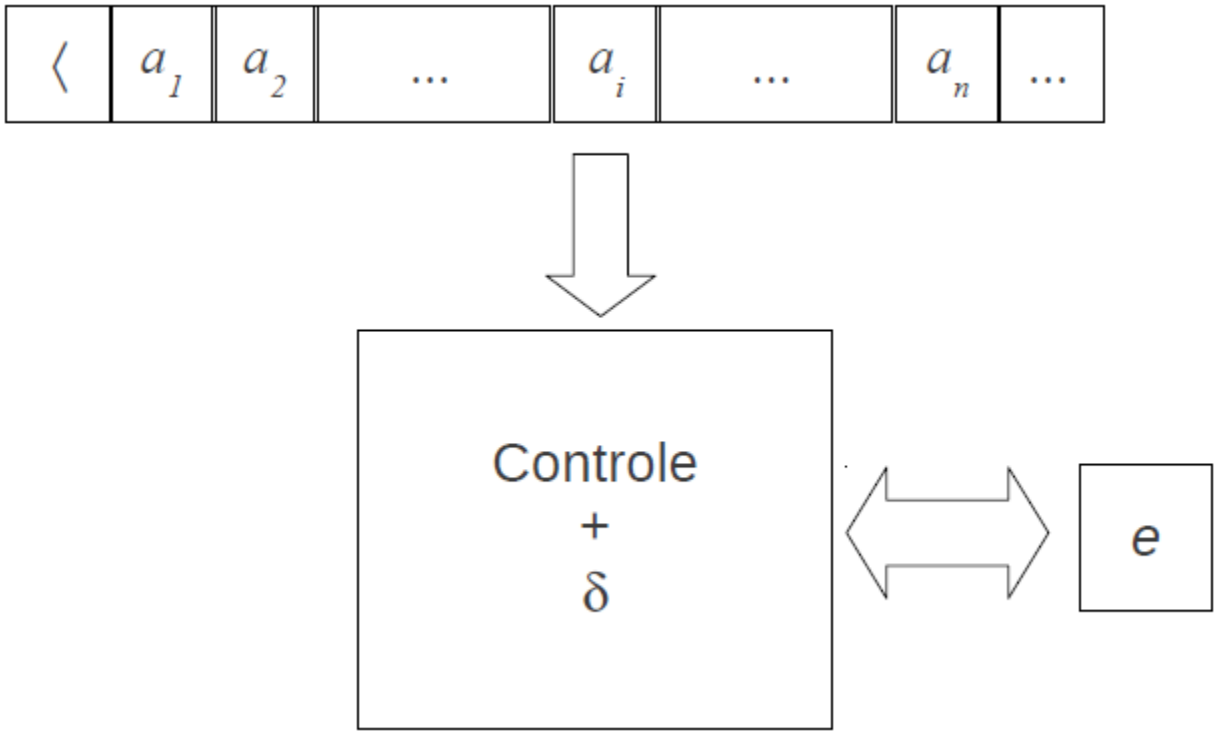
* Uma máquina de Turing é um dispositivo teórico, um modelo abstrato de um computador, com uma capacidade de memória infinita.

* Se restringe apenas aos aspectos lógicos do seu funcionamento (memória, estados e transições) e não à sua implementação física.

Máquina de Turing

* Uma MT pode ser vista como uma máquina que opera sobre uma fita na qual, além de ler, pode-se também escrever
  * A fita é dividida em células que suportam apenas um símbolo cada uma.
  * A fita é ilimitada à direita.
  * O cabeçote de leitura pode se mover para a direita e para a esquerda
  * Possui um registrador de estado atual
  * Um conjunto de instruções (transições)
  * Uma unidade de controle

Arquitetura de uma MT

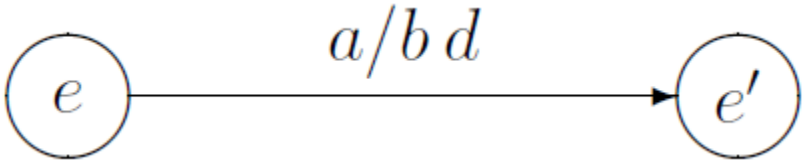


Símbolos adicionais

* $\langle$: Limita a fita à esquerda
  * Evita que o cabeçote de leitura se mova para a esquerda, além do limite da fita
* $\sqcup$: *Branco*, ou célula vazia
  * Denota que uma célula está vazia

* Os símbolos $\langle$ e $\sqcup$ não podem fazer parte do alfabeto de entrada!

Transições

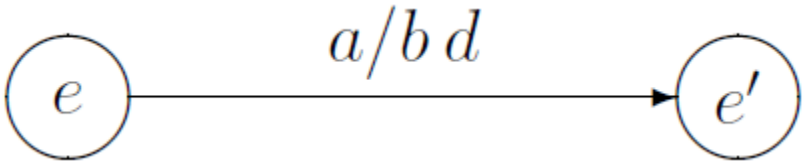


* Informalmente, a transição da máquina funciona assim:
* a/b {E,D} – Se tiver o símbolo a no cabeçote de leitura (condição), então troca esse símbolo por b e avança o cabeçote para a célula da esquerda (d = E) ou para a da direita (d = D).

Transições

* Observe que uma MT é determinística quando: para cada estado e e símbolo a há, no máximo, uma transição especificada pela função de transição.

* Formalmente, Uma transição $\delta(e,a) = [e', b, d]$, onde $d \in \{E,D\}$, será representada assim em um diagrama de estados:



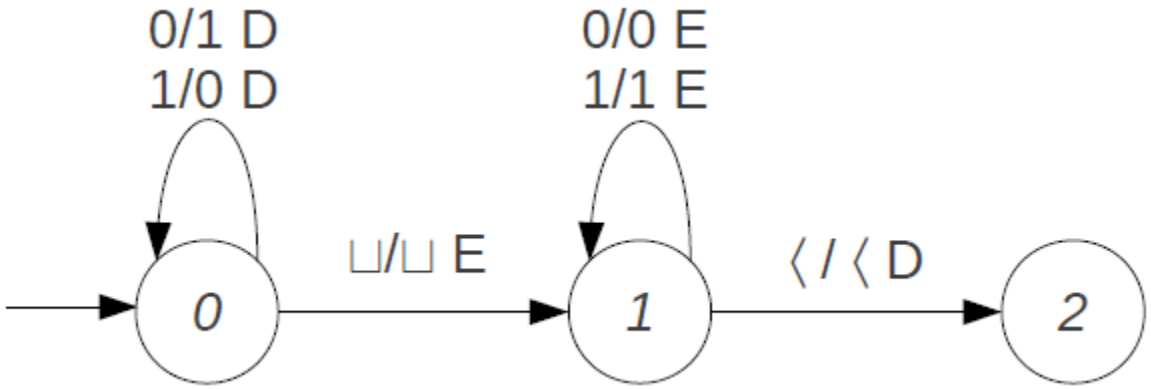
MTs transdutoras e reconhecedoras

* Uma MT pode ser usada como reconhecedora de linguagens e também como transdutoras. Neste último caso, a MT, recebendo na fita como entrada uma palavra w, produz na própria fita a saída respectiva.

* Apesar do nosso enfoque privilegiar o uso de MTs como reconhecedoras de linguagens, o primeiro exemplo, apresentado a seguir, trata de uma MT do tipo transdutora.

Exemplo 1

* MT que transforma a palavra de entrada em seu complemento. Depois, retorna o cabeçote para a posição inicial
  * Alfabeto: {0,1}
  * MT Transdutora: Traduz uma entrada em outra.



Exemplo 1

* Exemplo de execução:
* Palavra *w*: 101
* Fita:

⟨	1	0	1

* Posição inicial do cabeçote de leitura no símbolo 1;
* Estado 0 – cabeçote no símbolo 1, substitui na fita por 0 e avança na fita para a direita D.
* Estado 0 – cabeçote no símbolo 0, substitui na fita por 1 e avança na fita para a direita D.
* Estado 0 – cabeçote no símbolo 1, substitui na fita por 0 e avança na fita para a direita D.
* Fita:

⟨	0	1	0	Célula vazia

* Ir do Estado 0 para o Estado 1 – cabeçote na célula vazia, substitui na fita por célula vazia e avança na fita para a esquerda E.
* Estado 1 – cabeçote no último símbolo 0, substitui na fita por 0 e avança na fita para a esquerda E.
* Estado 1 – cabeçote no símbolo 1, substitui na fita por 1 e avança na fita para a esquerda E.
* Estado 1 – cabeçote no primeiro símbolo 0, substitui na fita por 0 e avança na fita para a esquerda E.
* Ir do Estado 1 para o Estado 2 – cabeçote no símbolo limitador, substitui na fita por símbolo limitador e avança na fita para a direita D.

Exemplo 1

* A MT transformou a palavra de entrada 101 em seu complemento 010. Depois, retornou o cabeçote para a posição inicial, símbolo 0.

Máquinas de Turing

* **Máquinas de Turing**
  * **Definição Formal**

Definição Padrão

* Uma MT é uma óctupla
* $M = (E, \Sigma, \Gamma, \langle, \sqcup, \delta, i, F)$ em que:

  * E é um conjunto finito de estados;
  * $\Sigma \subset \Gamma$ é o alfabeto de entrada;
  * Γ é o alfabeto da fita, que contém todos os símbolos que podem aparecer na fita;
  * $\langle$ é o primeiro símbolo da fita $(\langle \in \Gamma - \Sigma)$
  * $\sqcup$ é o símbolo *branco* $(\sqcup \in \Gamma - \Sigma, \sqcup \neq \langle)$
  * $\delta : E \times \Gamma \rightarrow E \times \Gamma \times \{E, D\}$ (Esquerda ou Direita) é a função de transição, uma função parcial
  * i é o estado inicial
  * $F \subseteq E$ é o conjunto de estados finais

Configuração Instantânea

* Seja uma MT M
* $M = (E, \Sigma, \Gamma, \langle, \sqcup, \delta, i, F)$

Uma configuração instantânea de M é um par $[e, x\underline{a}y]$, em que:

* $e \in E$ é o estado atual;
* $x \in \Gamma^*$ é a palavra situada à esquerda do cabeçote de leitura;
* $a \in \Gamma$ é o símbolo sob o cabeçote; e
* $y \in \Gamma^*$ é a palavra situada à direita do cabeçote de leitura até o último símbolo diferente de $\sqcup$.
    * Se não existir símbolo diferente de $\sqcup$, então y=λ.

Palavra Reconhecida

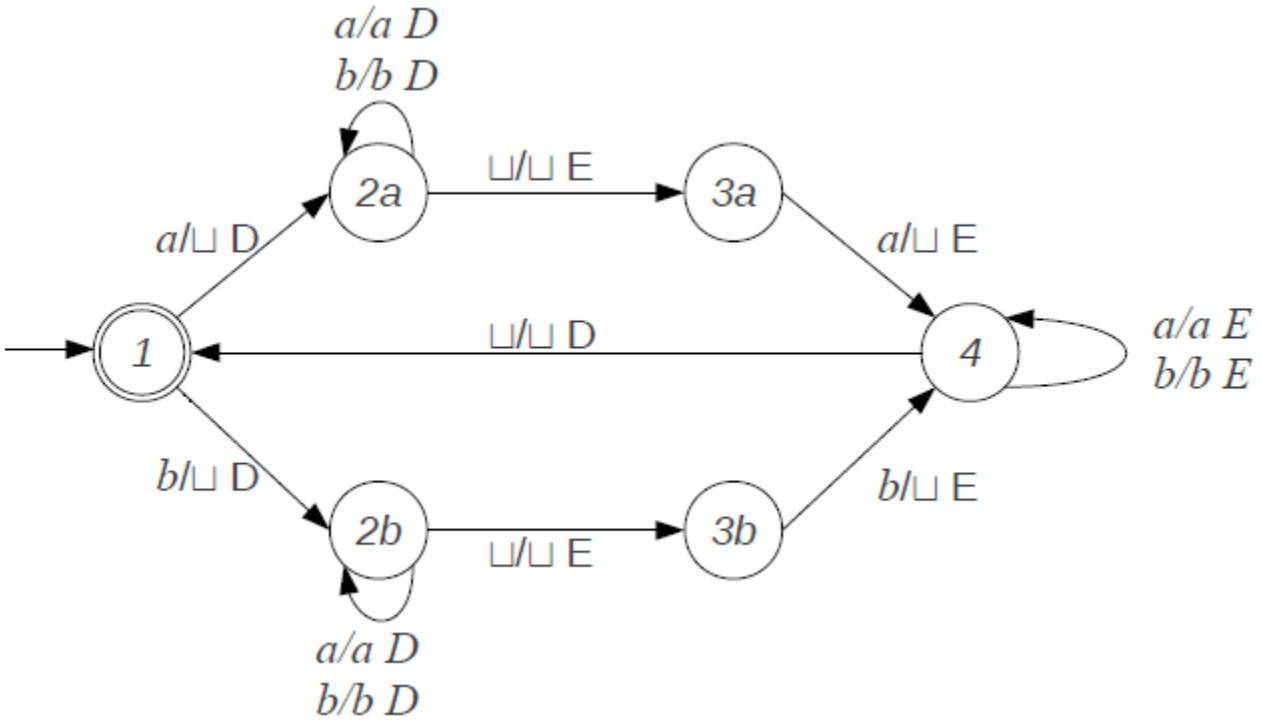
* A palavra $w \in \Sigma^*$ é dita reconhecida (aceita) pela MT quando:

  * $[\, i, \langle \underline{w} \,] \overset{*}{\vdash} [e, x\underline{a}y]$;
  * $\delta(e,a)$ é indefinido; e
  * e $\in F$.

  * Ou seja, a palavra w é reconhecida quando a máquina pára em um estado final.

Exemplo 2

* MT que reconhece palíndromos de tamanho par sobre o alfabeto $\{a,b\}$. Exemplos: $abba$, $baab$, $aabbaa$, $bbaabb$...



Exemplo 2

* Exemplo de execução:
* Palavra *w*: *abba*
* Fita:

⟨	*a*	*b*	*b*	*a*

* Posição inicial do cabeçote de leitura no símbolo *a*;
* Ir do estado inicial 1 para o estado 2a – cabeçote no símbolo *a*, substitui na fita por célula vazia e avança na fita para a direita D.
* Estado 2a – cabeçote no símbolo *b*, substitui na fita por *b* e avança na fita para a direita D.
* Estado 2a – cabeçote no símbolo *b*, substitui na fita por *b* e avança na fita para a direita D.
* Estado 2a – cabeçote no símbolo *a*, substitui na fita por *a* e avança na fita para a direita D.
* Fita:

⟨	*Célula vazia*	*b*	*b*	*a*	*Célula vazia*

* Ir do Estado 2a para o Estado 3a – cabeçote na célula vazia, substitui na fita por célula vazia e avança para a esquerda E.
* Ir do Estado 3a para o Estado 4 – cabeçote no último símbolo *a*, substitui na fita por célula vazia e avança para a esquerda E.
* Estado 4 – cabeçote no último símbolo *b*, substitui na fita por *b* e avança na fita para a esquerda E.
* Estado 4 – cabeçote no primeiro símbolo *b*, substitui na fita por *b* e avança na fita para a esquerda E.

Exemplo 2

* Continuação
* Fita Atual:

⟨	*Célula vazia*	*b*	*b*	*Célula vazia*	*Célula vazia*

* Ir do Estado 4 para o Estado 1 – cabeçote na primeira célula vazia, substitui na fita por célula vazia e avança para a direita D.
* Ir do Estado 1 para o Estado 2*b* – cabeçote no primeiro símbolo *b*, substitui na fita por célula vazia e avança para a direita D.
* Fita:

⟨	*Célula vazia*	*Célula vazia*	*b*	*Célula vazia*	*Célula vazia*

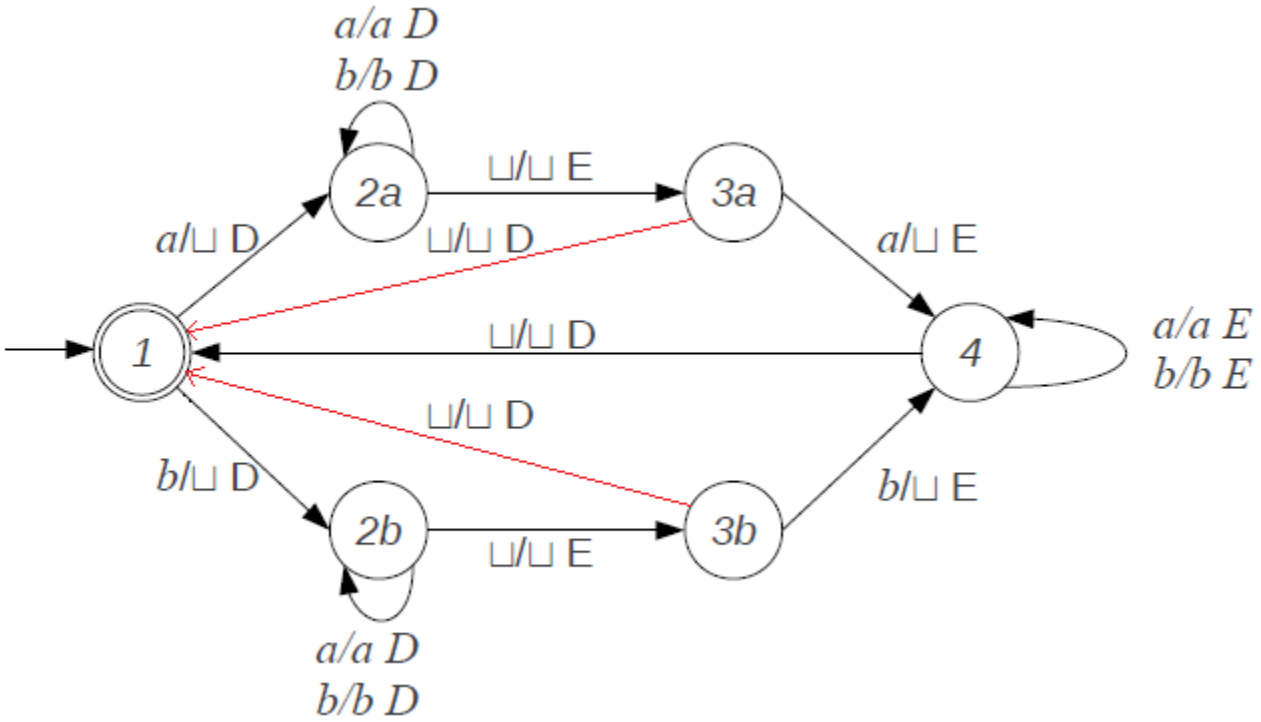
* Estado 2*b* – cabeçote no símbolo *b*, substitui na fita por *b* e avança na fita para a direita D.
* Ir do Estado 2*b* para o Estado 3*b* – cabeçote no penúltimo símbolo de célula vazia, substitui na fita por célula vazia e avança na fita para a esquerda E.
* Ir do Estado 3*b* para o Estado 4 – cabeçote no símbolo *b*, substitui na fita por célula vazia e avança na fita para a esquerda E.
* Fita:

⟨	*Célula vazia*	*Célula vazia*	*Célula vazia*	*Célula vazia*	*Célula vazia*

* Ir do Estado 4 para o Estado 1 – cabeçote no segundo símbolo de célula vazia, substitui na fita por célula vazia e avança na fita para a direita D.
* Cabeçote posicionado no primeiro símbolo de célula vazia, parou no estado 1 que é final. Reconheceu a word *abba*.

Exemplo 3

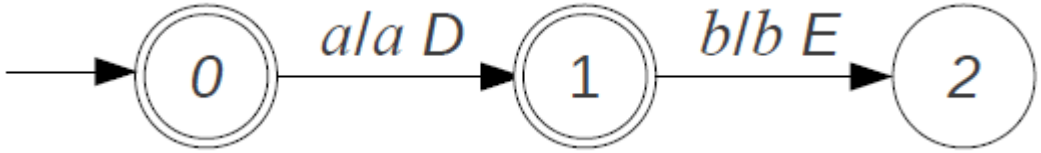
* MT que reconhece todos os palíndromos, seja de tamanho par ou seja de tamanho ímpar, sobre o alfabeto $\{a,b\}$. Exemplos: $abba$, aba, $baab$, bab, $aabbaa$, $bbbabbb$...



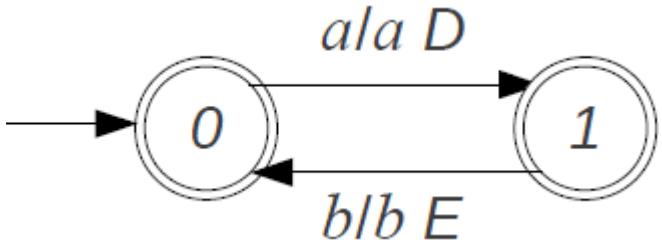
Exemplo 4

* MT que reconhece as palavras sobre o alfabeto $\{a,b\}$ que não se iniciam com '*ab*'.

* Máquina que pára sempre:



* Máquina que pára se aceita:



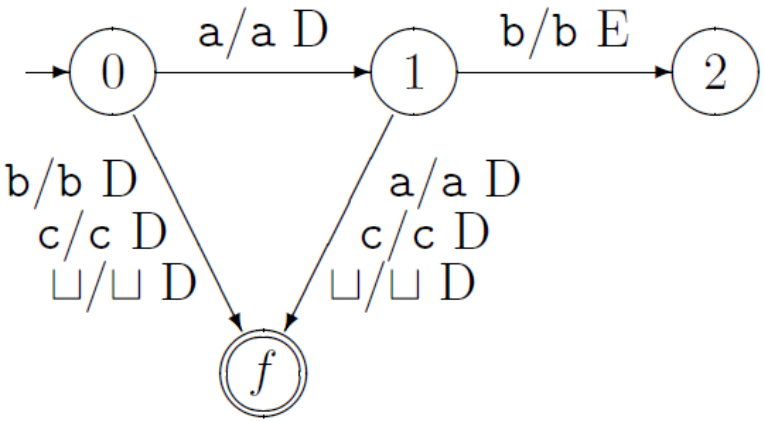
Exemplo 4

* O exemplo anterior ilustra, além do papel que a "não parada" tem com relação ao reconhecimento, uma faceta diferente das MT's com relação aos AF's e AP's: Não é necessário que uma palavra de entrada seja toda lida para que ela possa ser aceita ou rejeitada.

Exemplo 4

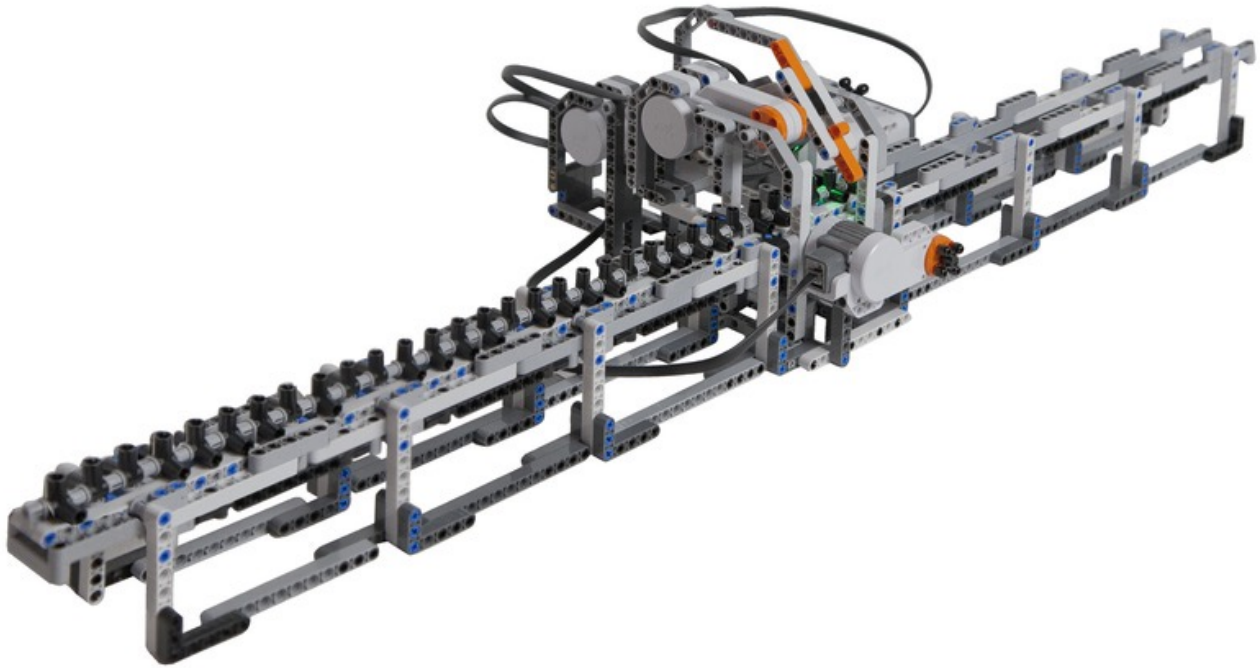
* MT que reconhece as palavras sobre o alfabeto $\{a,b,c\}$ que não se iniciam com '*ab*'.

* Máquina que reconhece por estado final:



MT do Lego

LEGO[tm] Turing Machine
in honor of the Alan Turing year 2012



MT do Lego

* Links
* http://legoofdoom.blogspot.com.br/
* https://www.youtube.com/watch?v=FTSAiF9AHN4

Exercícios

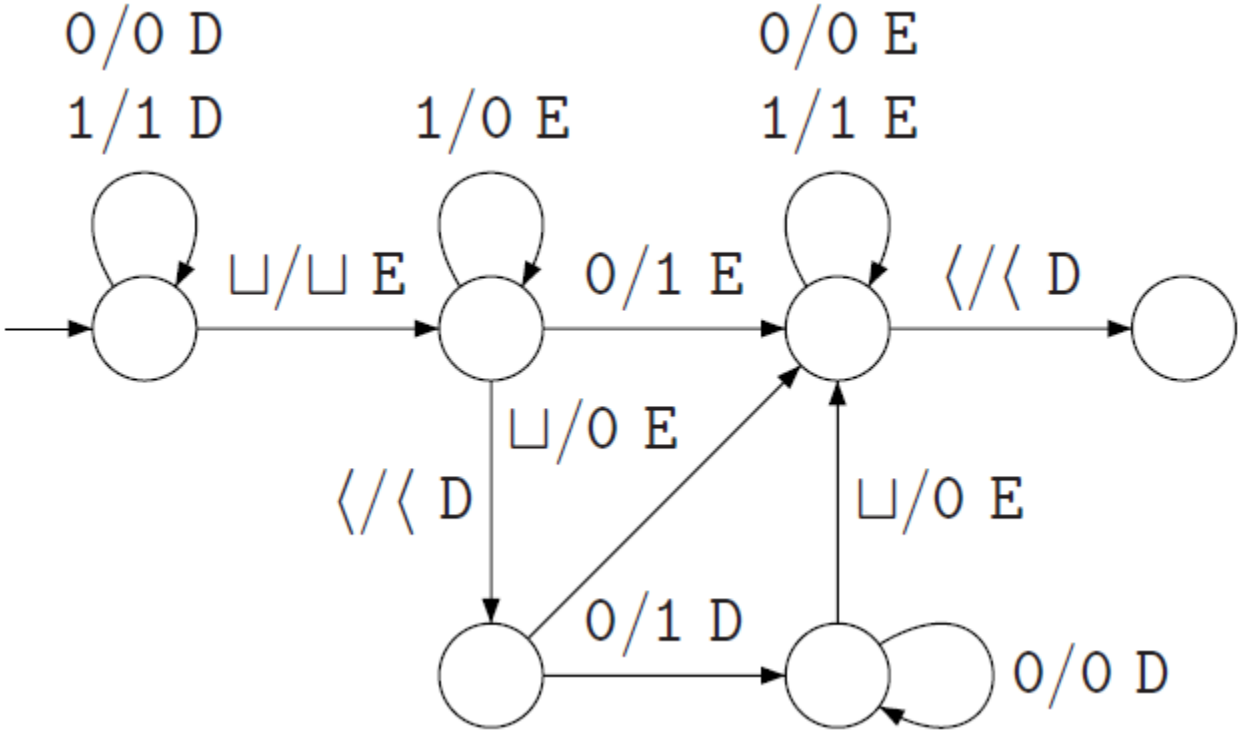
1) Construa uma MT que, recebendo como entrada um número binário, some 1 ao mesmo e retorne o cabeçote para a posição inicial. Se a palavra for λ, a MT deve escrever 0.

2) Faça uma MT que, recebendo uma palavra w sobre o alfabeto $\{a\}$, concatena a mesma palavra imediatamente à direita de w e retorne o cabeçote para a posição inicial. Por exemplo, se a configuração inicial for [i, ⟨$\underline{a}aa$], a configuração final deverá ser [i, ⟨$\underline{a}aaaaa$]

3) Projete uma MT com alfabeto de entrada $\{a\}$ que pare somente se a palavra de entrada w tiver tamanho par.

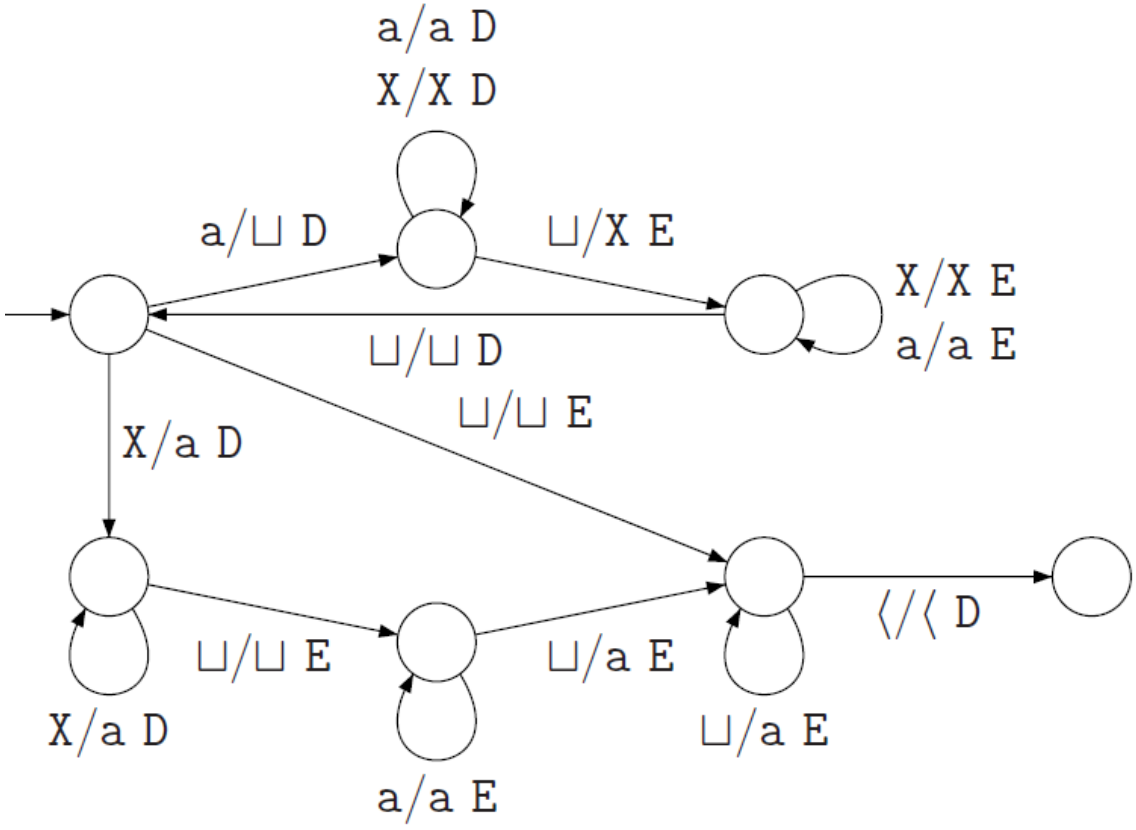
Respostas

1)



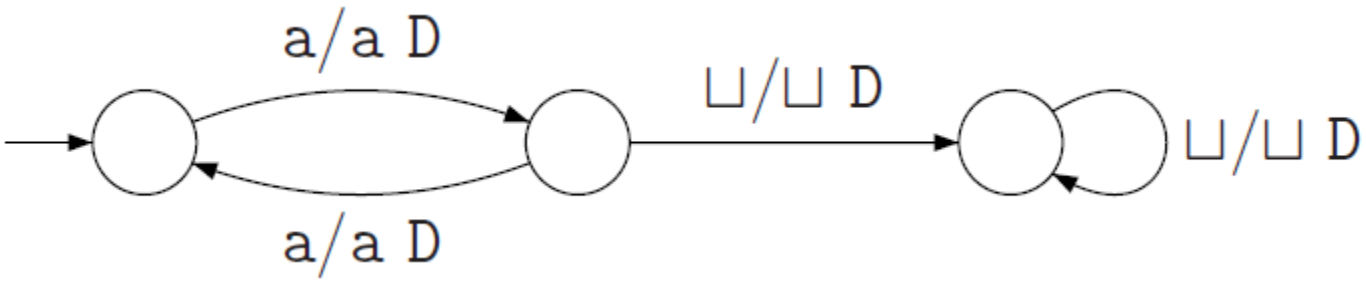
Respostas

2)



Respostas

3)



Hierarquia de Chomsky

* **Hierarquia de Chomsky**

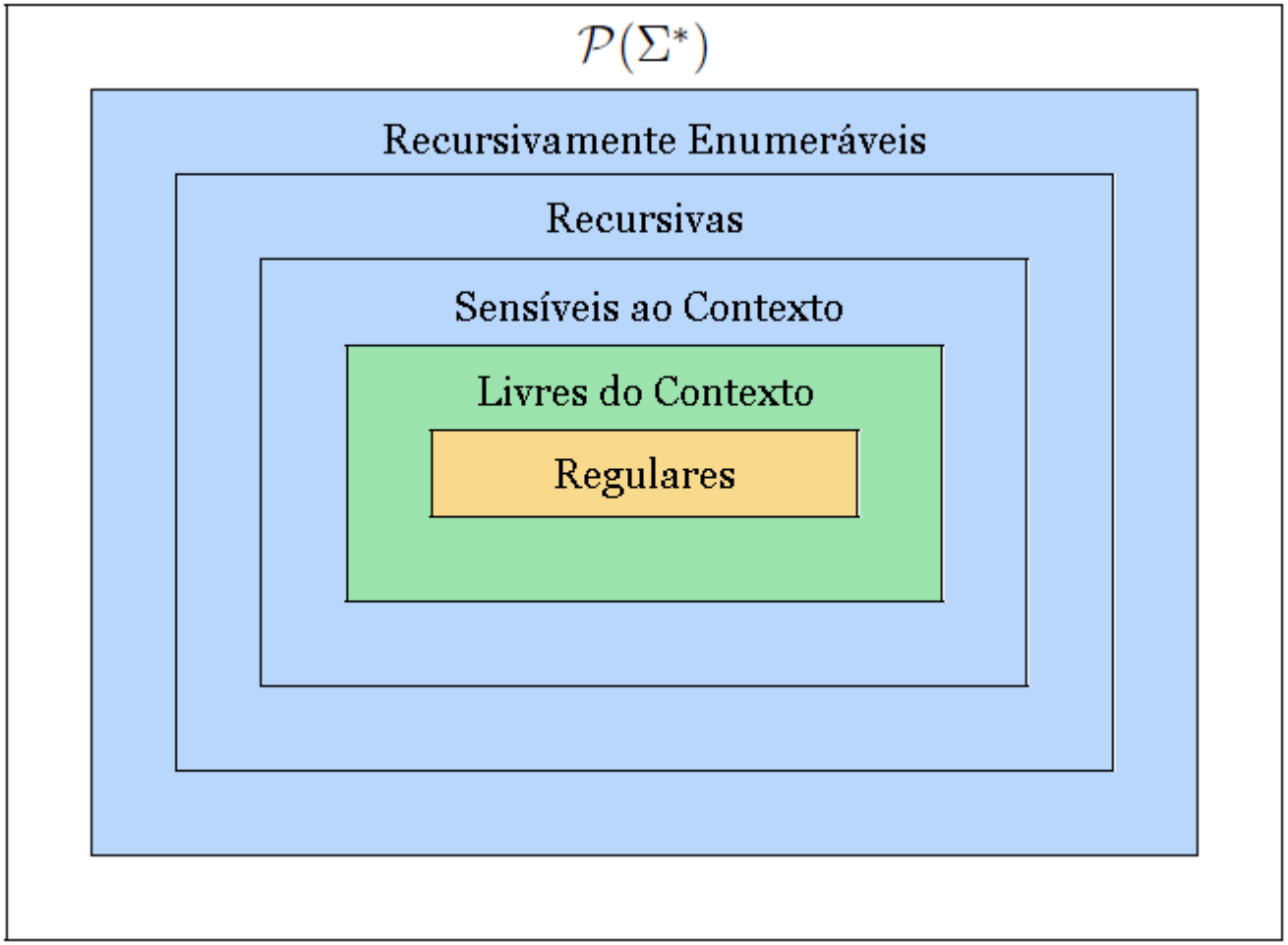
Hierarquia de Chomsky

* A classificação das gramáticas nos quatro tipos:

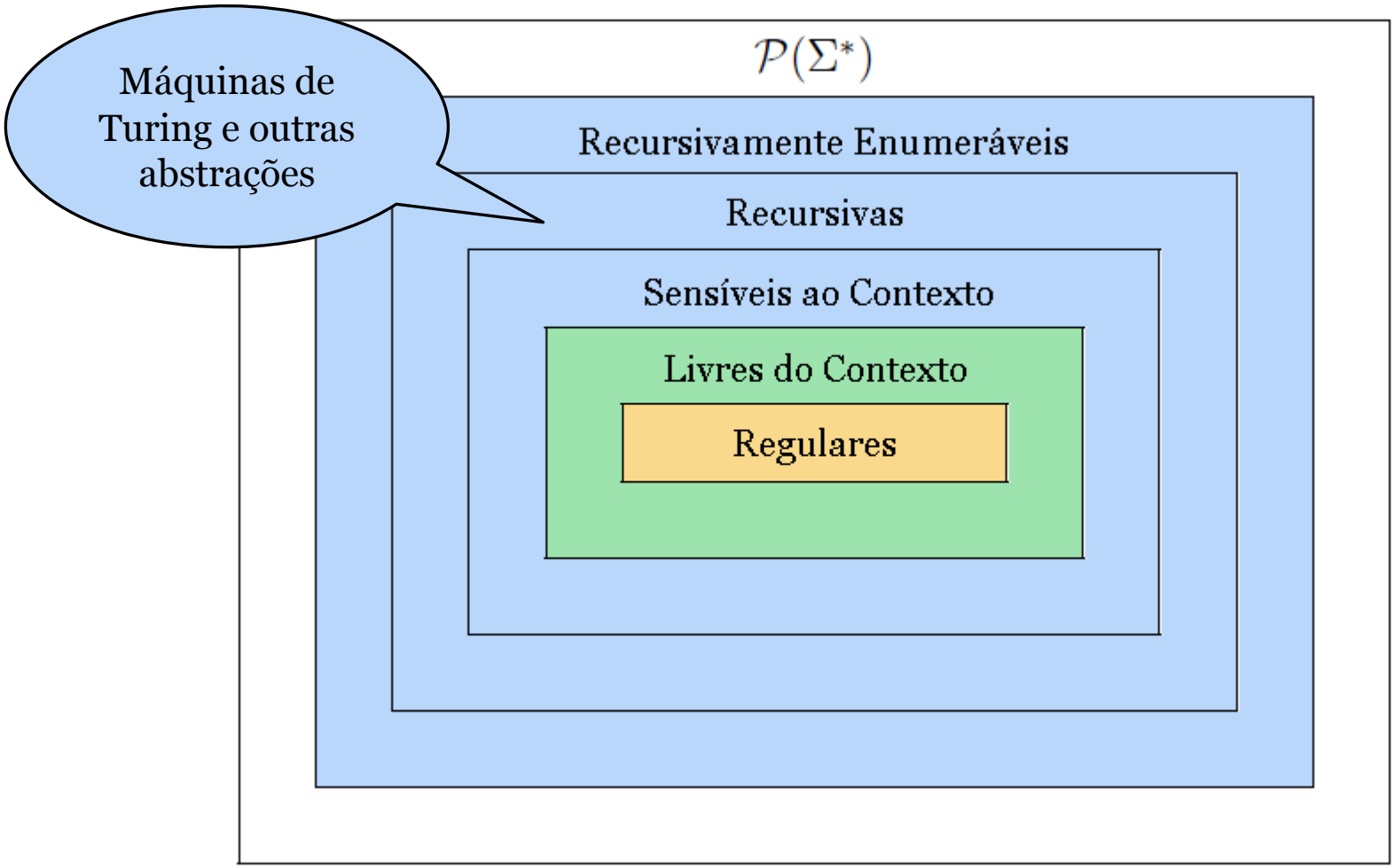
* Regulares;
* Livres do Contexto;
* Sensíveis do Contexto; e
* Irrestritas (Recursivamente Enumeráveis)

* é a denominada *hierarquia de Chomsky*

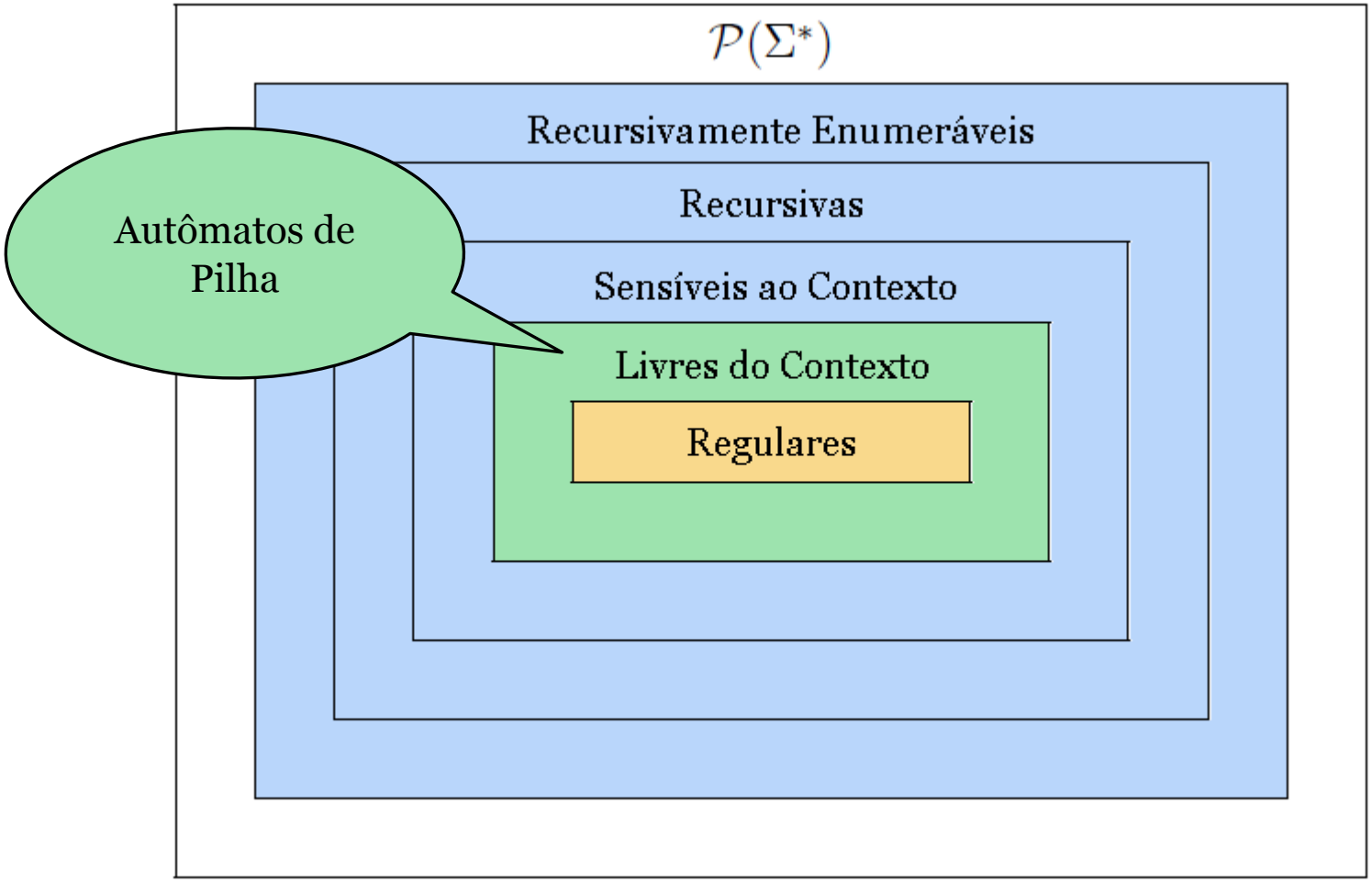
* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



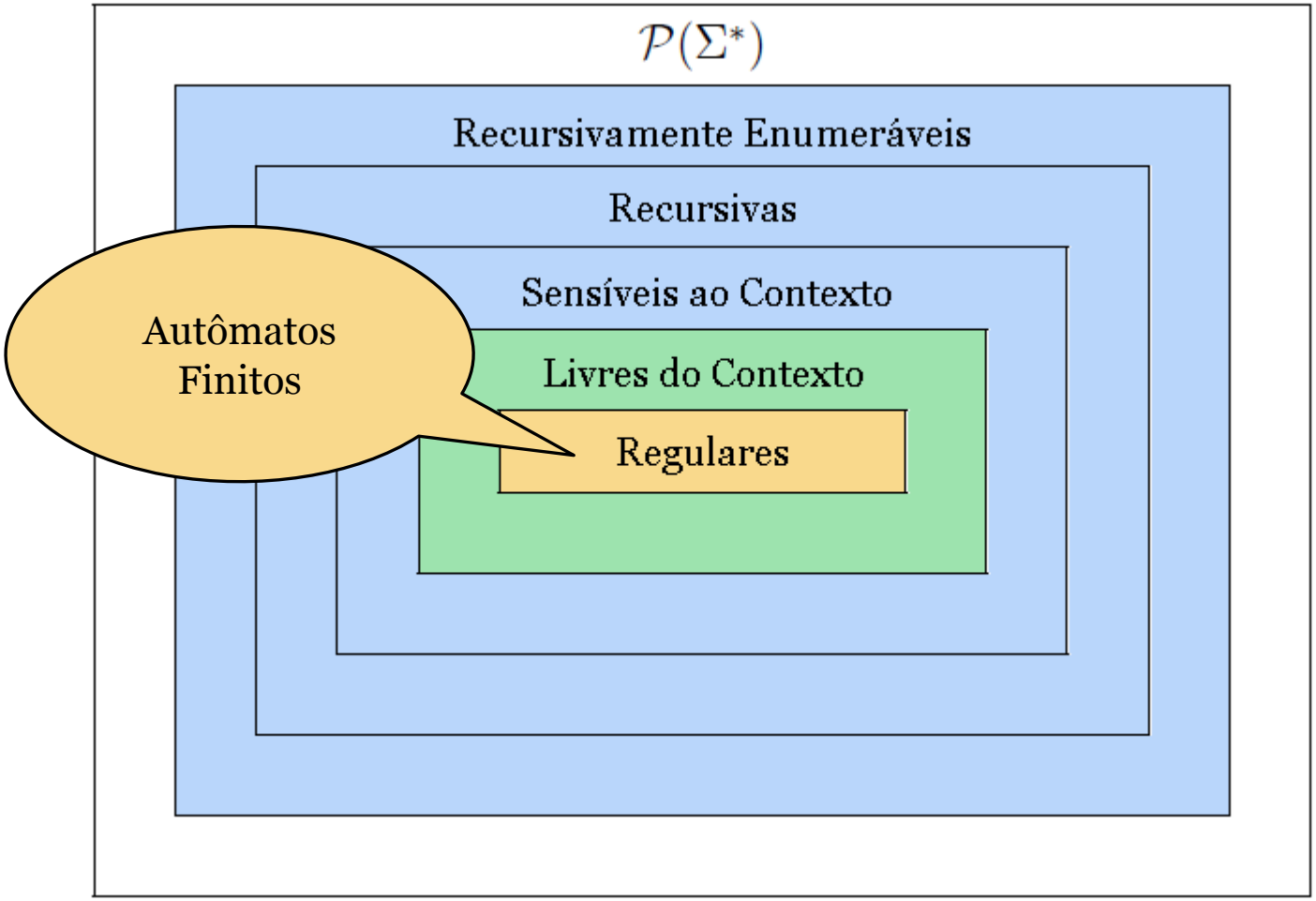
* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



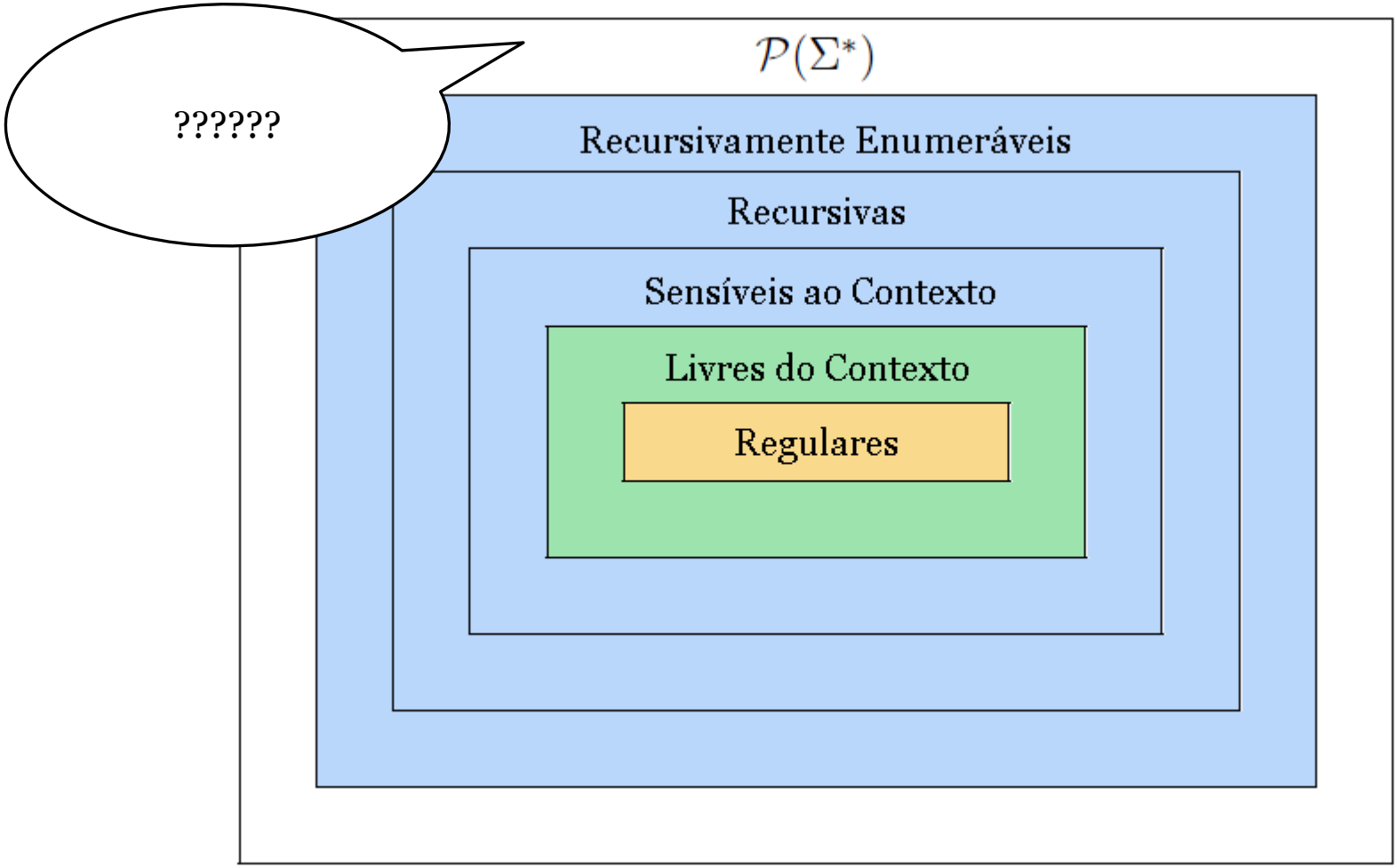
* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



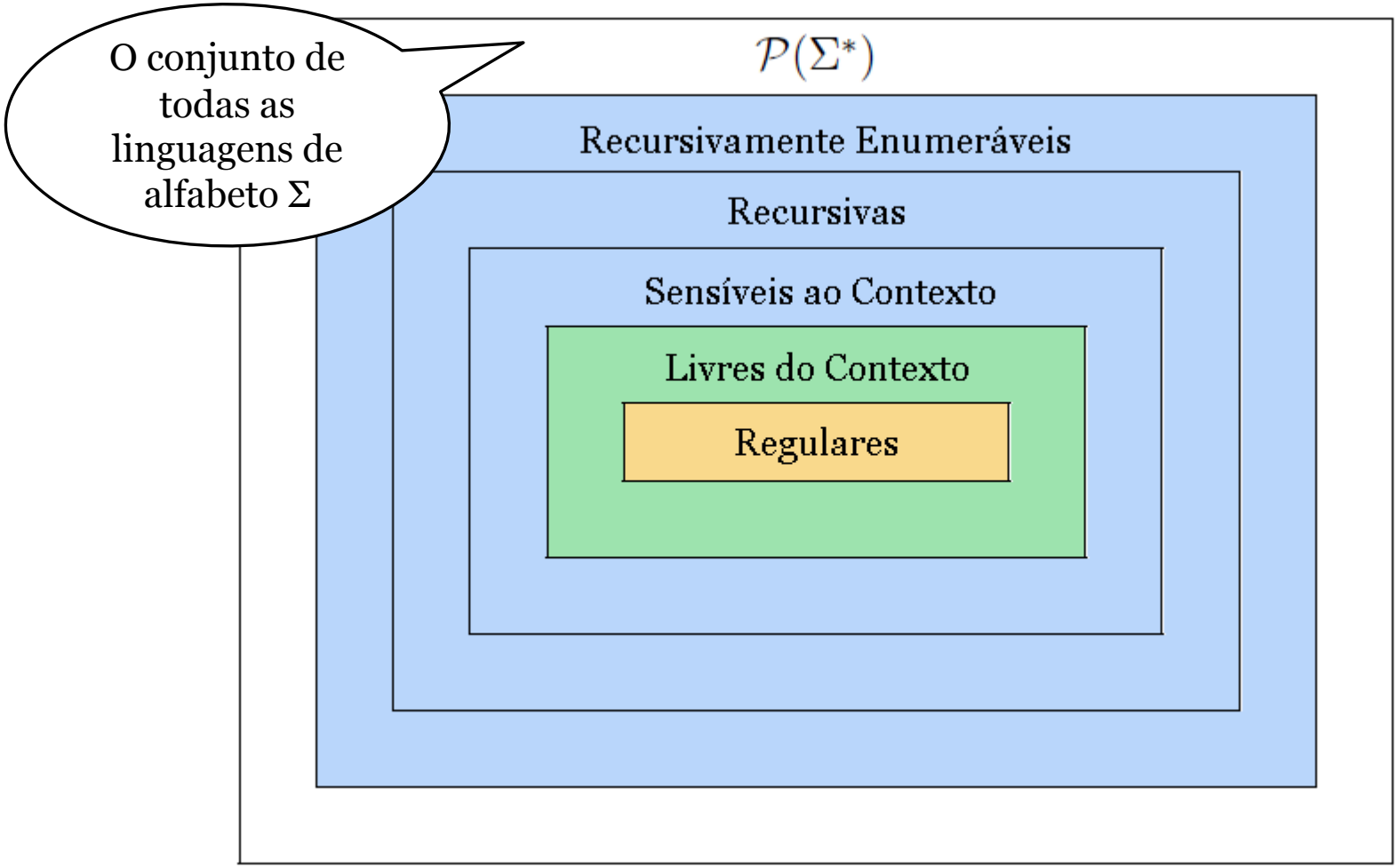
* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



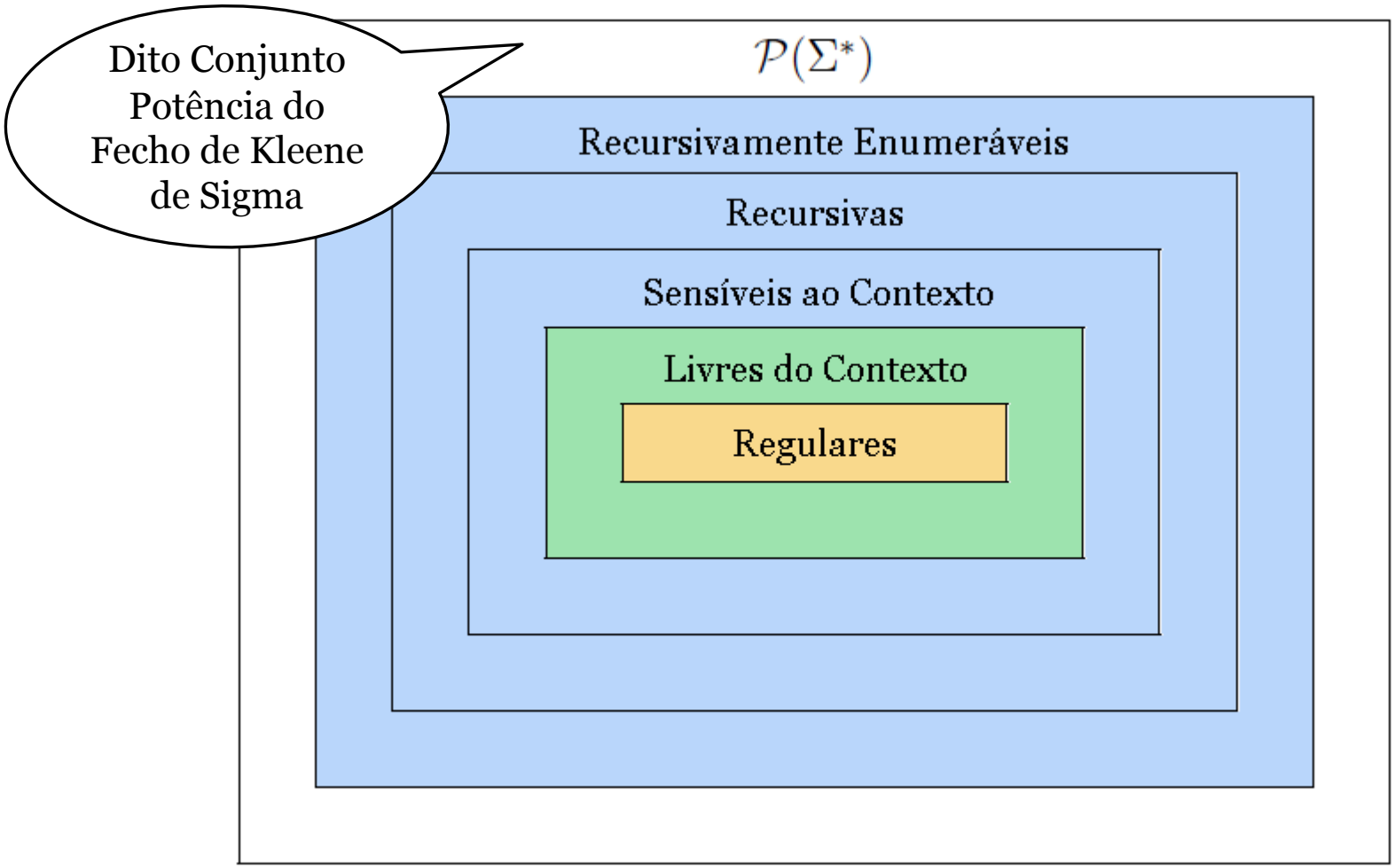
* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



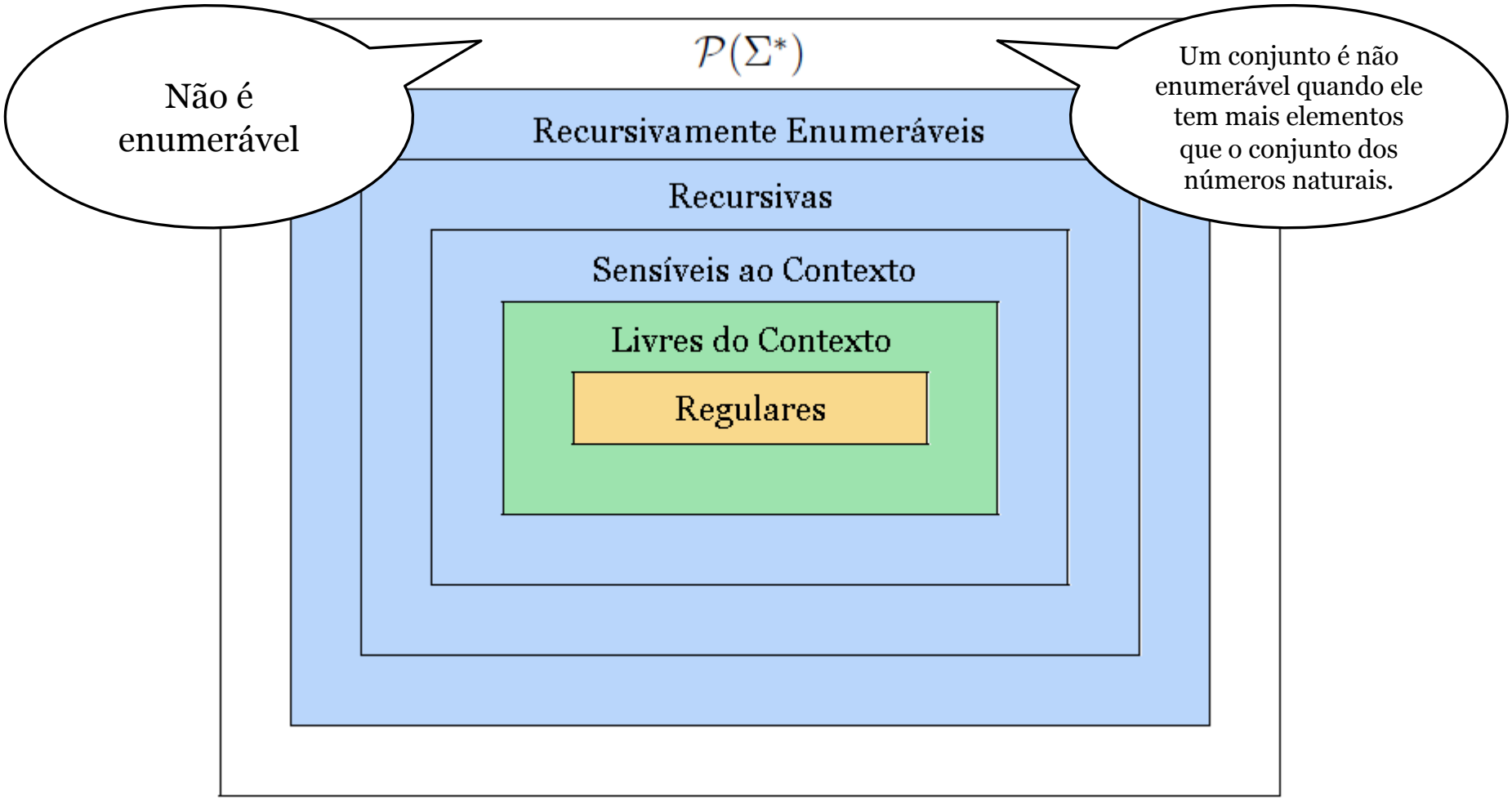
* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



* Hierarquia de Chomsky: Espaço das linguagens em $P(\Sigma^*)$



Decidibilidade

* **Decidibilidade**
  * **Introdução**
  * **Tese de Church-Turing**
  * **O Problema da Parada**

Decidibilidade

* O objetivo deste tópico é dar uma noção dos limites do conceito de computação.

* Isto é feito através da apresentação de alguns problemas de enunciado muito simples para os quais não existem soluções computacionais (algoritmos), dentro os quais o tradicional "*problema da parada*" (que é indecidível).

Tese de Church-Turing

* A tese, de 1936, leva os nome dos matemáticos Alonzo Church e Alan Turing.

* Alan Turing tentou capturar a noção de algoritmo (então chamado "computabilidade efetiva"), com a introdução de máquinas de Turing.

Tese de Church-Turing

* Os computadores digitais, tanto os mais antigos quanto os mais modernos, foram construídos com um conjunto de instruções que lhes dão um poder computacional idêntico ao das máquinas de Turing e ao de outros formalismos.

* Mais modernamente, as linguagens de programação de alto nível, como Java, C, Pascal etc., com o mesmo poder expressivo, apareceram no intuito de facilitar a tarefa de programação propriamente dita. Mas, não se deve perder de vista que tais linguagens não apresentam maior expressividade do que, por exemplo, as MTs, embora sejam, obviamente, muito mais adequadas do ponto de vista prático para a confecção de algoritmos.

Tese de Church-Turing

* Estritamente falando, o poder computacional associado aos formalismos matemáticos citados é maior que o de qualquer computador real (construído fisicamente), em virtude do fato de que um computador real tem uma memória limitada (4 GB, 8 GB, etc).

* No caso das MTs, por exemplo, podemos executar um algoritmo considerando a memória infinita.

Tese de Church-Turing

* Entre os diversos formalismos matemáticos propostos, a máquina de Turing é um dos mais aderentes aos computadores atuais, isto é, pode-se considerar que a máquina de Turing captura a parte "essencial", aquela responsável, em última análise, pelo poder computacional dos computadores atuais.

Tese de Church-Turing

* A tese de Church-Turing pode ser assim enunciada:
* *"Se uma função é efetivamente computável, então ela é computável por meio de uma máquina de Turing."*
  * Ou: Todo algoritmo pode ser expresso mediante uma MT

* Nunca foi definido formalmente a noção de *computação efetiva*
  * Portanto não há como provar a tese
  * Porém nenhum formalismo posterior conseguiu ter poder expressivo maior que ao da máquina de Turing

Tese de Church-Turing

* Por não ter sido provada, a tese de Church-Turing, apesar de amplamente difundida e aceita, poderia ser REFUTADA caso alguém descobrisse uma função que fosse universalmente aceita como um algoritmo efetivo mas que não pudesse ser computada por meio de uma MT.

* Ou seja, se você encontrar um algoritmo que NÃO possa ser expresso mediante uma MT, você terá grandes chances de ganhar o cobiçado Prêmio Alan Turing.

O Problema da Parada

* O célebre problema da parada não é decidível, tanto para máquinas de Turing, quanto para linguagens do tipo Pascal, C, Java, etc.

* O problema da parada para MT's pode ser assim enunciado:

"Dadas uma MT arbitrária M e uma palavra arbitrária w, determinar se a computação de M com a entrada w pára."

Regras para a vida

* Regras para a vida:

"O problema da parada para MT's é indecidível."

"O problema da parada para linguagens de programação procedurais comuns é indecidível."

O Problema da Parada

* O problema é determinar para uma dada entrada se o programa irá parar com ela.

* Neste tipo de abstração não há limitações de memória ou tempo necessário para execução de um programa, pois poderão ser necessários tempo e espaço arbitrários para o programa parar.

O Problema da Parada

* Partindo-se da configuração inicial:
  * Se a máquina parar e e for um estado final, a palavra de entrada é aceita.
  * Se a máquina parar e e não for um estado final, a palavra de entrada não é aceita.
  * Se a máquina não parar (entrar em loop infinito), a palavra de entrada não é aceita.

O Problema da Parada

* A questão é se o programa simplesmente poderá parar com uma certa entrada.

* O programa pode rodar por um tempo fixo e se ele não parar, não há um jeito de saber se o programa irá parar eventualmente ou se ele irá continuar rodando para sempre.

* Turing provou que não há um algoritmo que possa ser aplicado a qualquer programa arbitrário, com uma entrada, para decidir se o programa pára ou não com esta entrada.

Obrigado.

joaopauloaramuni@gmail.com
joaopauloaramuni@fumec.br

UNIVERSIDADE FUMEC
50 Anos