MATC94 – **INTRODUÇÃO ÀS LINGUAGENS FORMAIS E TEORIA DA COMPUTAÇÃO**

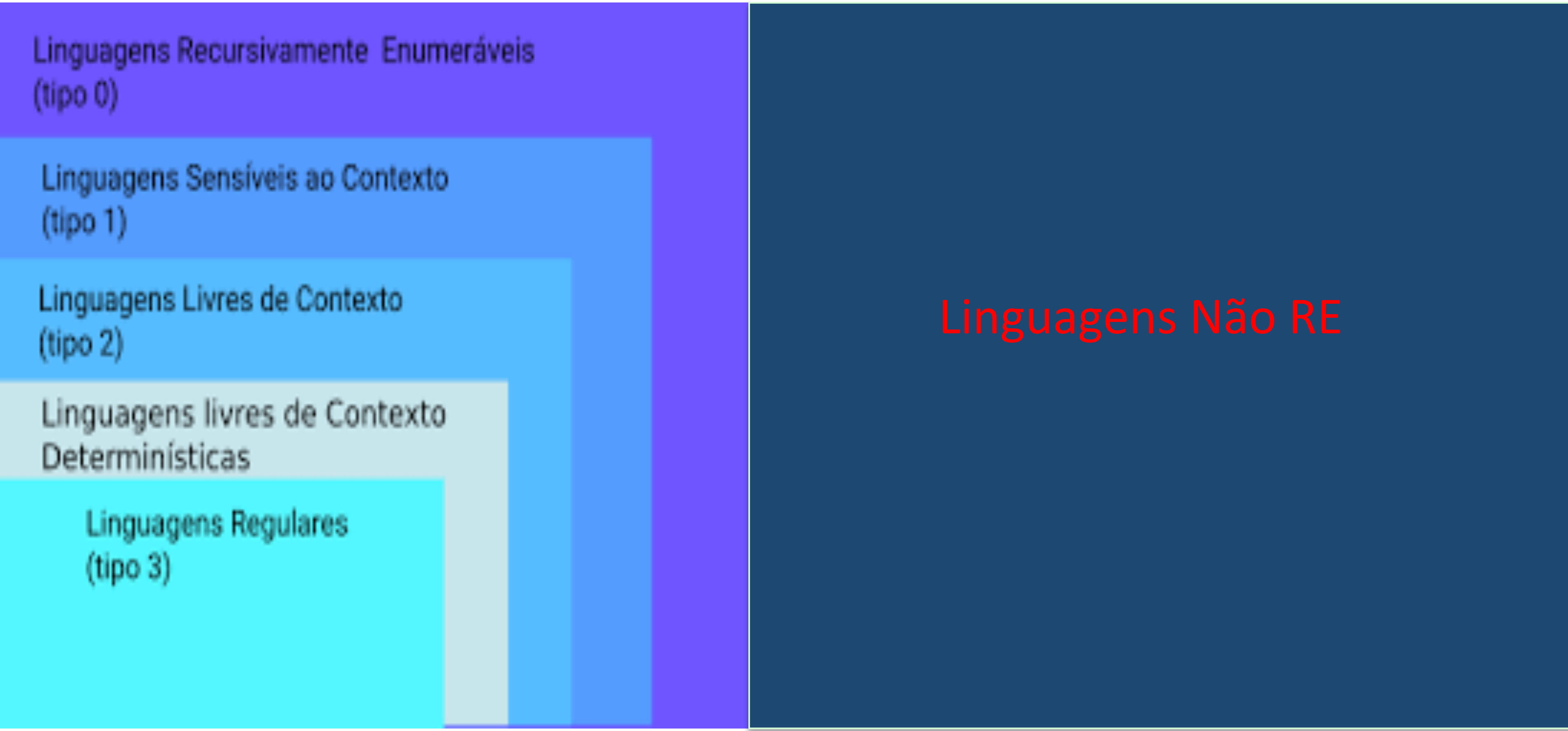
(Tese de Church / Máquina de Turing Universal / Indecidibilidade)

Docente: Laís Salvador

Tutores: Roberta Mércia e Luiz Gavaza

Universo das Linguagens

Hierarquia de Chomsky



Linguagens Recursivas e Recursivamente Enumeráveis

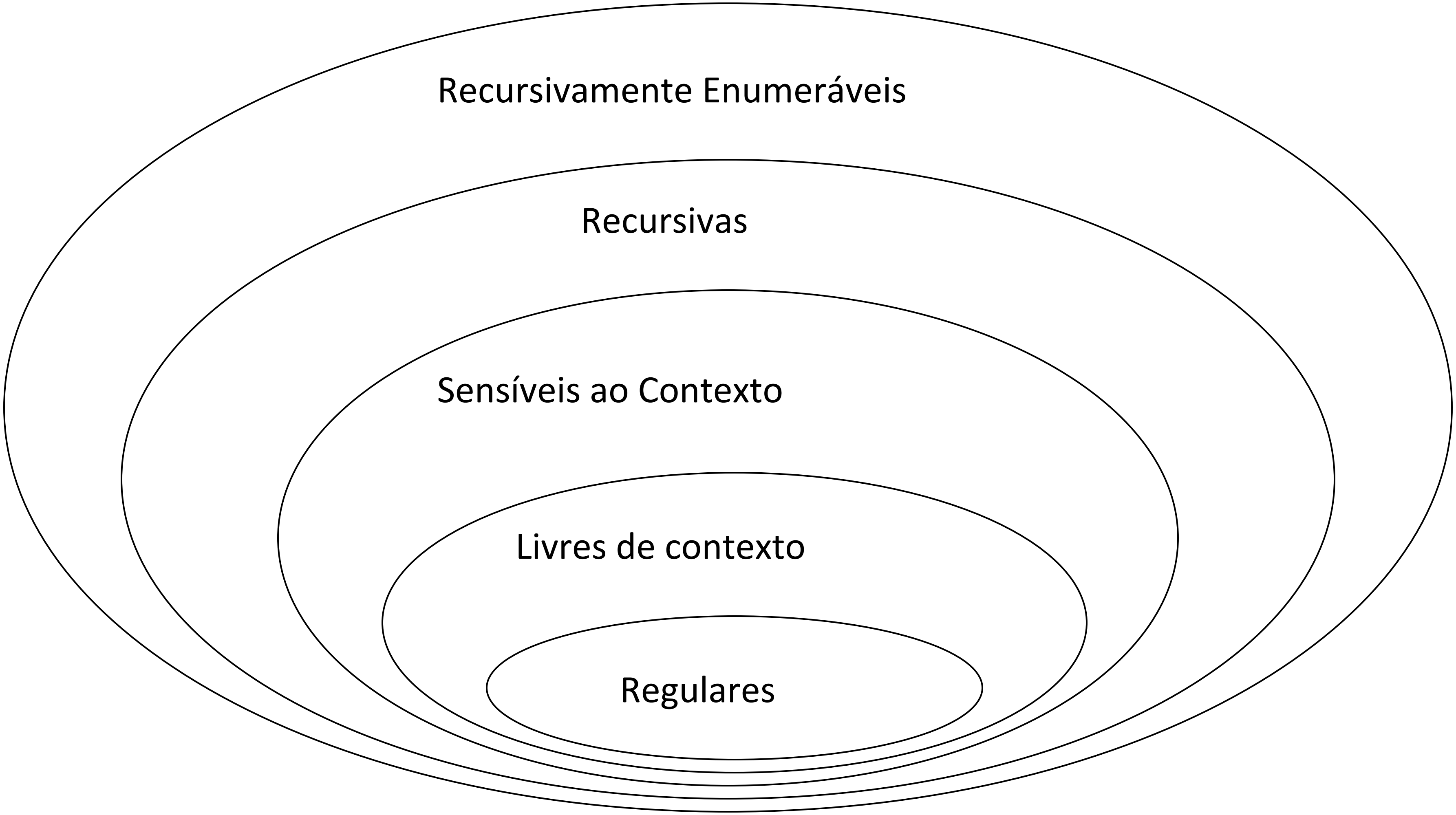
- **Uma linguagem estritamente recursivamente enumerável (r.e.) é uma linguagem reconhecida (semi-decidida) por uma Máquina de Turing (mT)**
  - **A mT pára quando analisa uma palavra pertencente à linguagem e entra em loop infinito quando analisa uma cadeia que não pertence à linguagem.**
  - **w $\in$ L se e somente se mT pára na entrada w.**
  - **Uma linguagem estritamente r.e. é uma linguagem r.e. que não é recursiva. O que é uma linguagem recursiva?**

Linguagens Recursivas e Recursivamente Enumeráveis

- **Uma linguagem recursiva é uma linguagem aceita por uma mT que sempre pára. Ela é decidida por uma mT**

A Hierarquia de Chomsky com outros detalhes

Linguagens Não Recursivamente Enumeráveis: não reconhecidas por Mts



Linguagens Recursivas e Recursivamente Enumeráveis

- **Se uma mT decide uma linguagem ou computa uma função, isso pode ser considerado como um algoritmo que executa correta e constantemente uma tarefa computacional**
- **Uma mT que semidecide uma linguagem L pode não ser proveitosa para dizer se w está em L, porque se w $\notin$ L, nunca saberemos quanto tempo esperar para obter uma resposta.**
  - **mTs que semidecidem linguagens não são algoritmos.**

Tese de Church

- **As máquinas de Turing que param em todas as entradas são versões formais da idéia intuitiva de algoritmo, e nada será considerado como um algoritmo se não puder ser reproduzido como uma máquina de Turing, cuja parada é garantida em todas as entradas.**
- **Por que esta afirmação é uma tese mas não é um teorema?**

Tese de Church

- **Porque não é um resultado matemático**

  Algoritmo – conceito informal

  X

  mT – conceito matemático

- **A Tese de Church pode ser desprovada?**
- **Sim, se alguém propuser um modelo de computação mais poderoso que a mT**
  - **Ninguém considera isso possível.**

Modelos Equivalentes à mTs

“ **Uma das razões para considerar a Máquina de Turing como o mais geral dispositivo de computação é o fato de que todos os demais modelos e máquinas propostos, bem como diversas modificações da Máquina de Turing, possuem, no máximo, o mesmo poder computacional da Máquina de Turing.**
” [Menezes 1998]

Uma pergunta:

Qual é o limite das mT ?

Elas resolvem todos os problemas computacionais?

Máquina de Turing Universal

Uma limitação das mTs:

Máquinas de Turing são “hardwired”

elas executam
apenas um programa

Computadores reais são re-programáveis

Máquina de Turing Universal

Solução: Máquina de Turing Universal

Atributos:

- Reprogramável

- Simula qualquer outra mT

Máquina de Turing Universal

mT Universal U

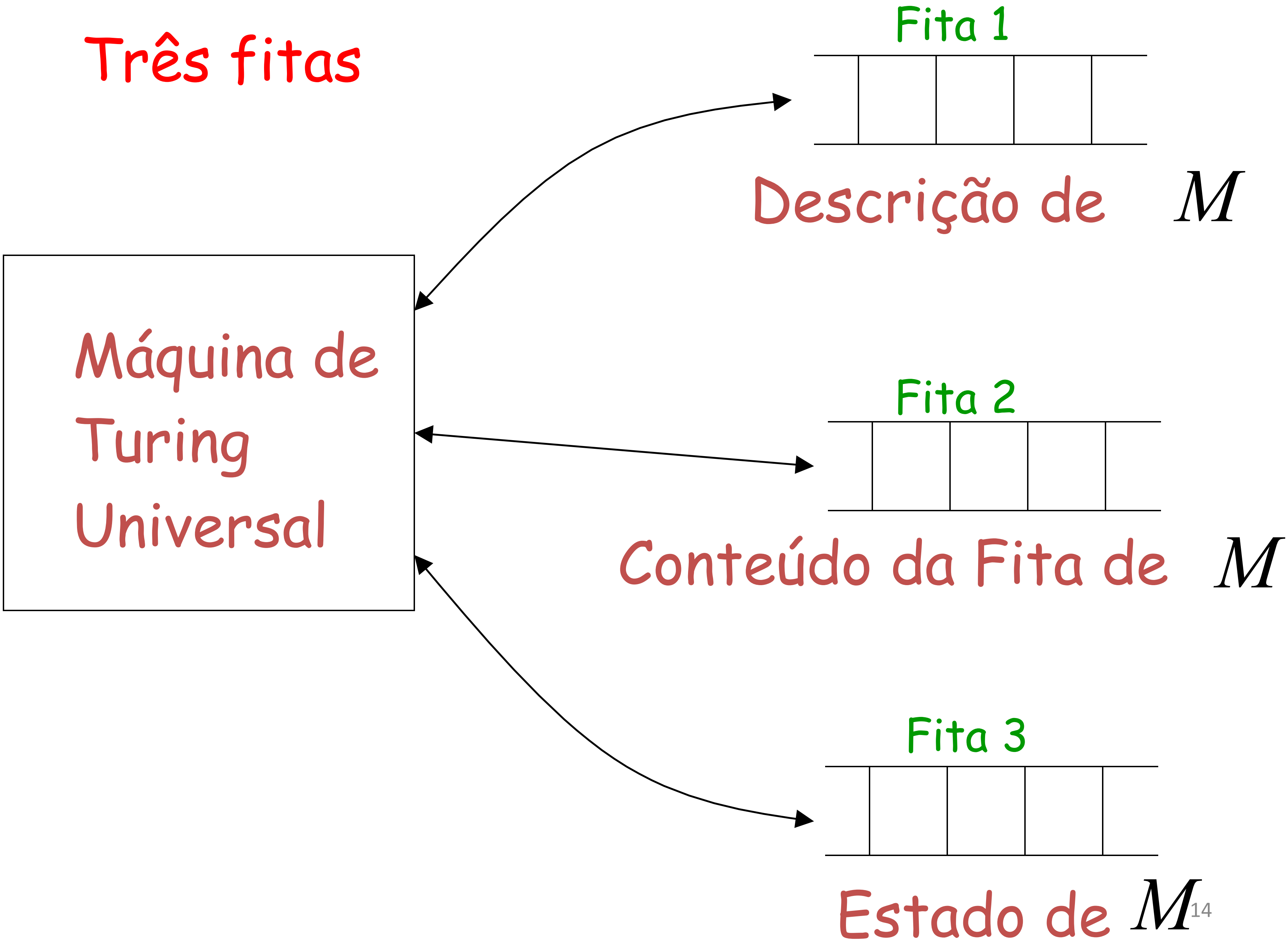
simula qualquer outra mT M

Ela recebe como entrada:

Descrição de M

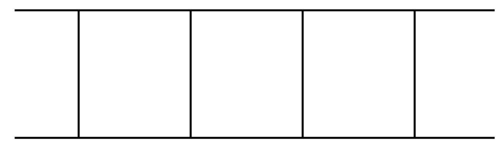
Conteúdo inicial da fita de M

Três fitas


Fita 1
Descrição de M
Máquina de Turing Universal
Fita 2
Conteúdo da Fita de M
Fita 3
Estado de M

Máquina de Turing Universal

Fita 1



Descrição de M

A mT M é descrita como uma cadeia de símbolos usando-se um alfabeto pré-determinado

Máquina de Turing Universal

Exemplo de codificação sobre $\sum = \{0,1\}$

Codificação do Alfabeto:

Símbolos:	a	b	c	d
Codificação:	111	1111	11111	111111

Máquina de Turing Universal

Codificação dos Estados:

Estados:	q_1	q_2	q_3	q_4
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Codificação:	1	11	111	1111

Máquina de Turing Universal

Codificação do movimento do cursor:

Movimento:	L	R
	$\downarrow$	$\downarrow$
Codificação:	1	11

Máquina de Turing Universal

Codificação da Transição

Transição: f(q1, a, L) → (q2, b)

Codificação: 10111010101101111

Separador 0 para distinguir elementos de uma tupla da função de transição

Máquina de Turing Universal

Codificação da Máquina

Transições:

f(q1, a,L) → (q2, b)

f(q2, a,R) → (q3, c)

Codificação:

101110101101111**00**110111011011101111

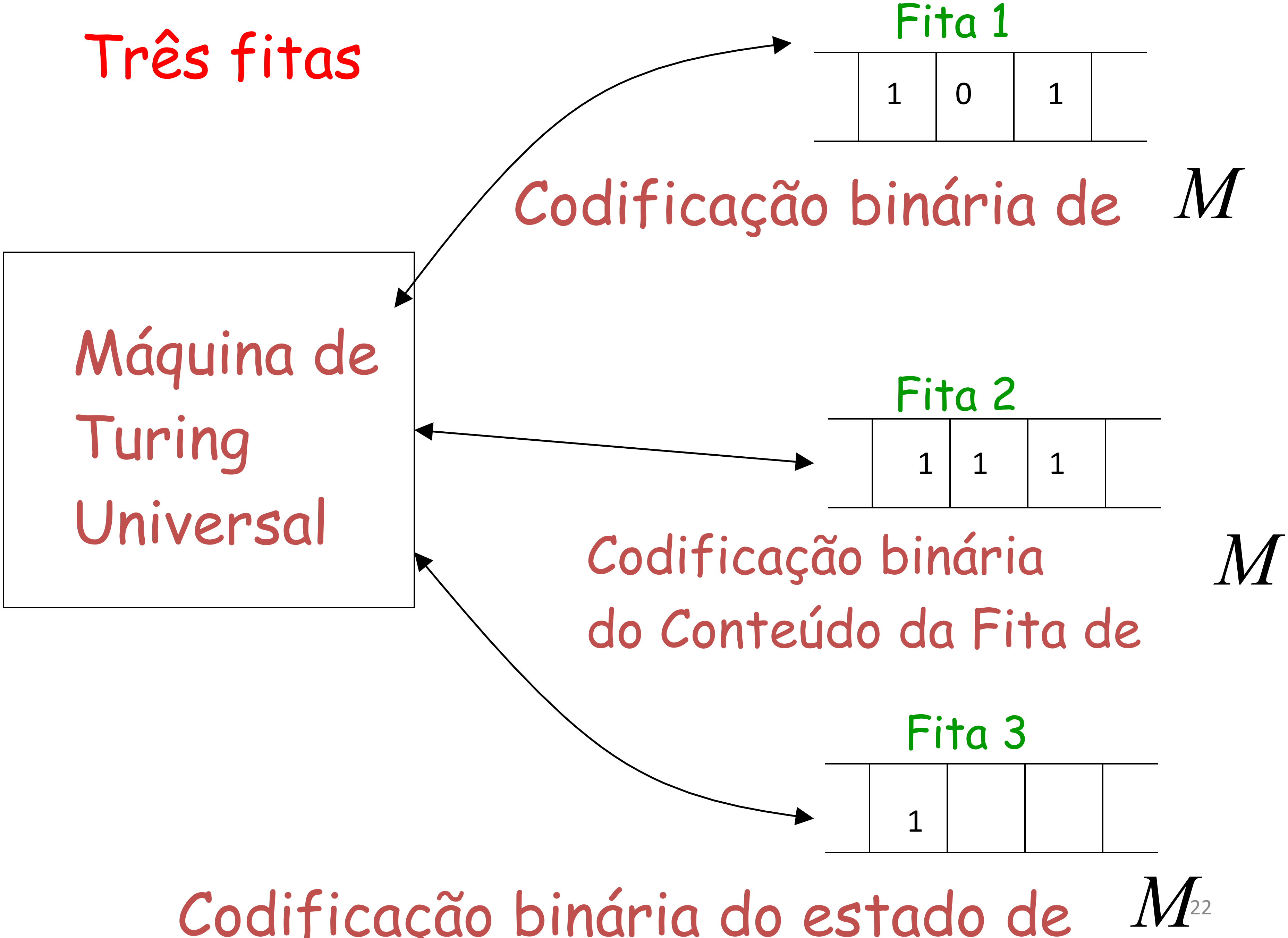
Separador 00 para distinguir as tuplas da função de transição

Máquina de Turing Universal

Conteúdo da Fita 1 da mT Universal:

codificação da máquina simulada M
como uma string de 0's e 1's

Três fitas

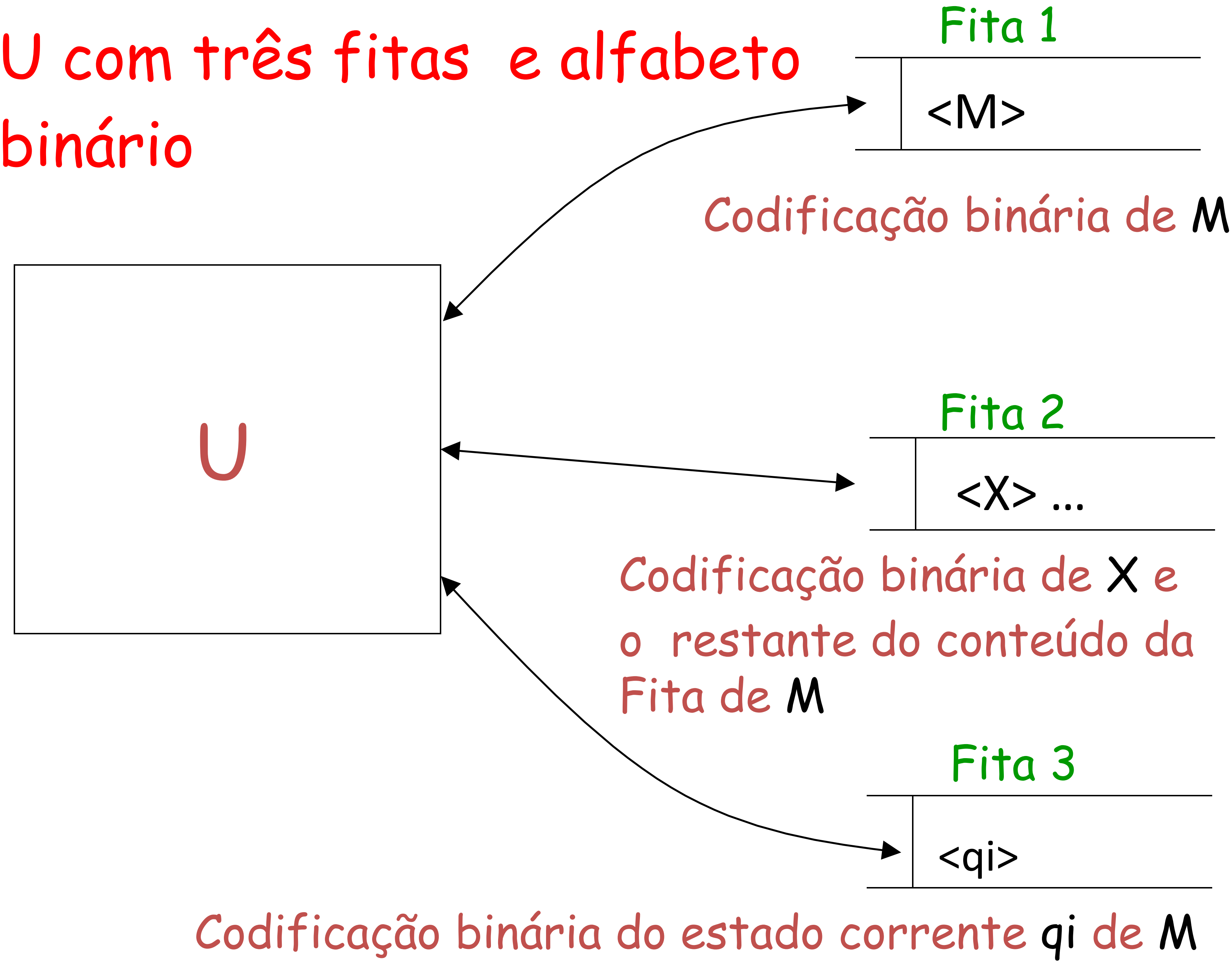

Fita 1
1 0 1
Codificação binária de M
Máquina de Turing Universal
Fita 2
1 1 1
Codificação binária do Conteúdo da Fita de M
Fita 3
1
Codificação binária do estado de M

Máquina de Turing Universal

Para a mt Universal (U) simular a computação de uma mT M qualquer, quando esta recebe como entrada a cadeia X, a idéia é a seguinte:

- U recebe uma entrada m na Fita1 e x na Fita2, onde m e x são as representações de M e X em algum alfabeto ∑;
- U simula M com entrada X;
- U aceita "m" "x" sse M aceita X.

U com três fitas e alfabeto binário



Máquina de Turing Universal

Mais especificamente:

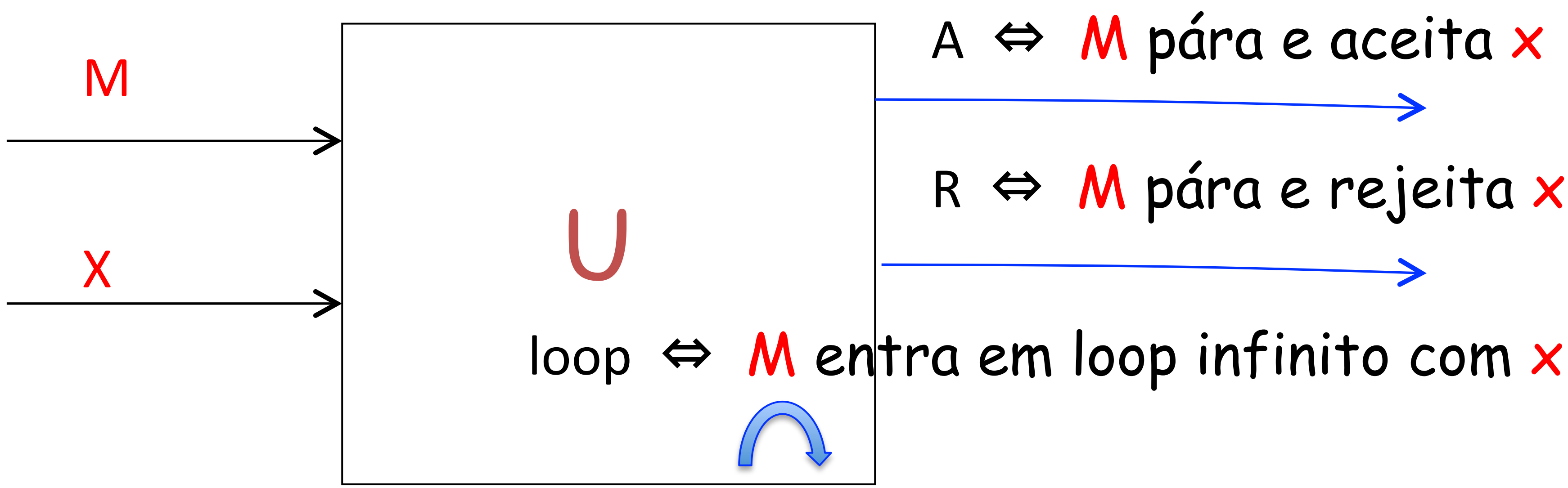
U simula M com a entrada x:

- U pára e aceita $\langle M \rangle x \Leftrightarrow$ M pára e aceita x ;
- U pára e rejeita $\langle M \rangle x \Leftrightarrow$ M pára e rejeita x ;
- U entra em loop infinito com $\langle M \rangle x \Leftrightarrow$ M entra em loop infinito com x ;

onde <M> é a representação de M em algum alfabeto Σ

$\Leftrightarrow$: se somente se

mTuring Universal



Setas (em azul) de saídas são alternativas de resultados de execução da máquina:

ou U pára e Aceita M e x

ou U pára e Rejeita M e x

ou U entra em loop infinito com as entradas M e x

Outra pergunta:

A mt Universal resolve tudo?

Ela resolve todos os problemas computacionais?

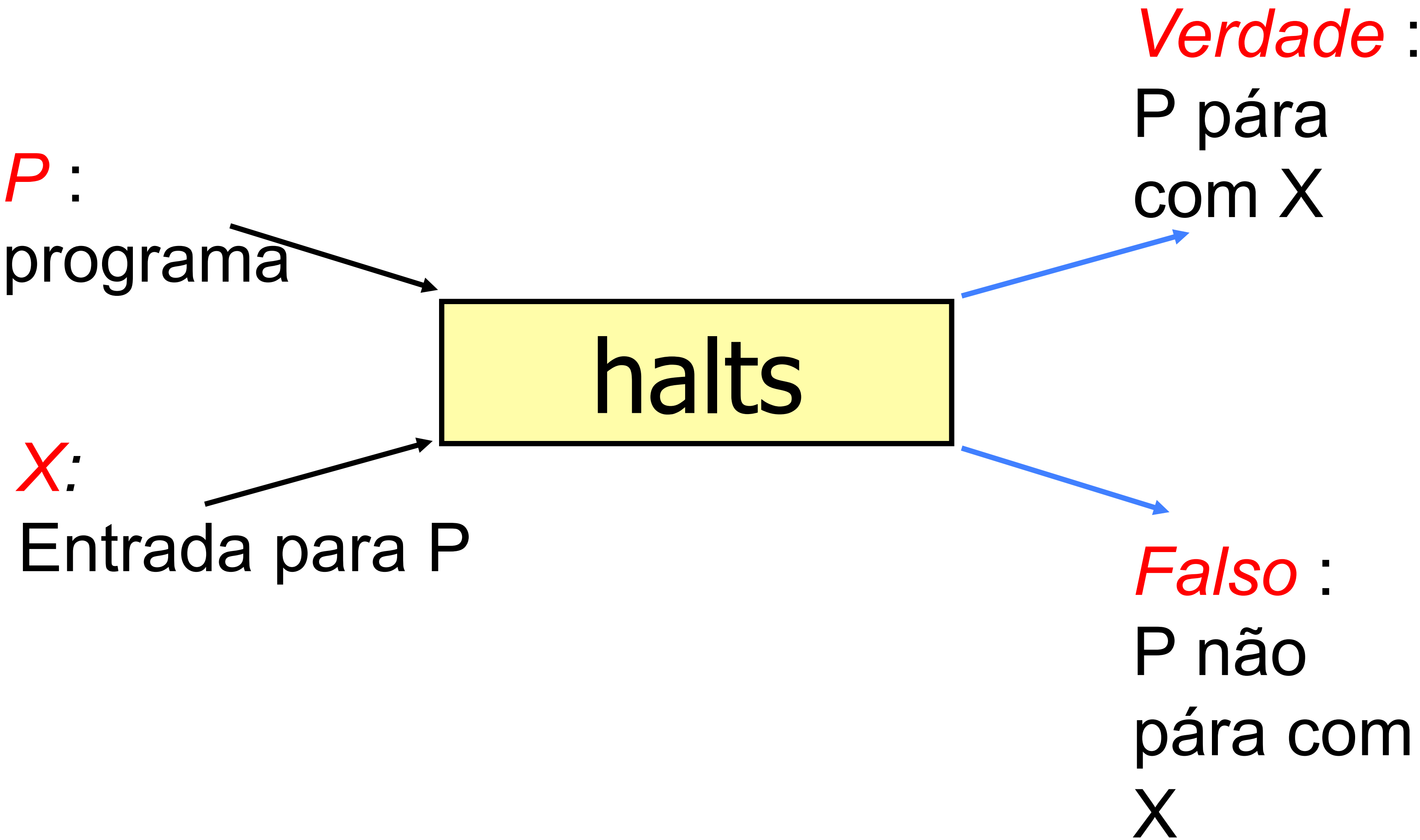
Problema da Parada

Assuma como hipótese a existência de um programa chamado **halts** escrito numa linguagem L, que realiza a seguinte proeza:

- Pega como entrada qualquer programa P em L e um entrada X de P.
- Realiza uma engenhosa análise e sempre determina corretamente se P irá parar na entrada X (**halts** retorna "true") ou se P entrará em loop infinito (**halts** retorna "false")

Este é o programa **halts(P,X)**

Problema da Parada



Problema da Parada

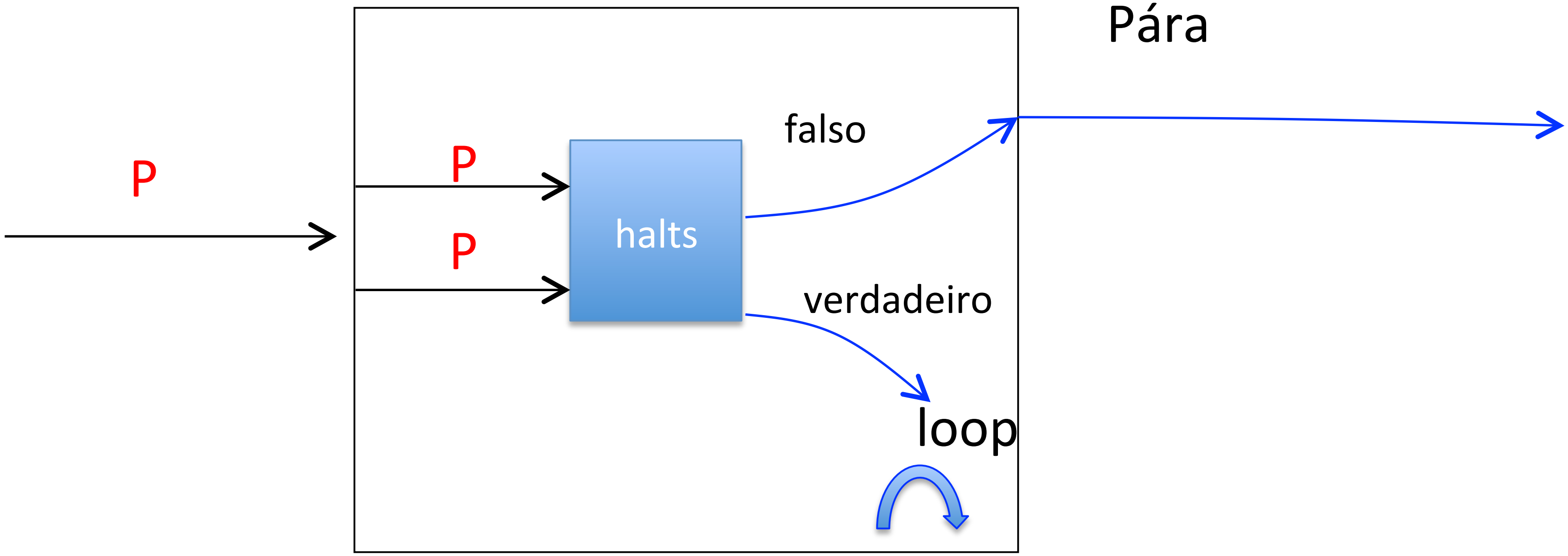
Observe este outro programa, chamado diagonal:

diagonal (P)

a: se halts (P,P) então goto a, senão pare

- O que faz diagonal (P)?
- Se halts decide que P pára com o próprio P como entrada,
  - então diagonal(P) entra em loop infinito
  - caso contrário, diagonal(P) pára

Diagonal



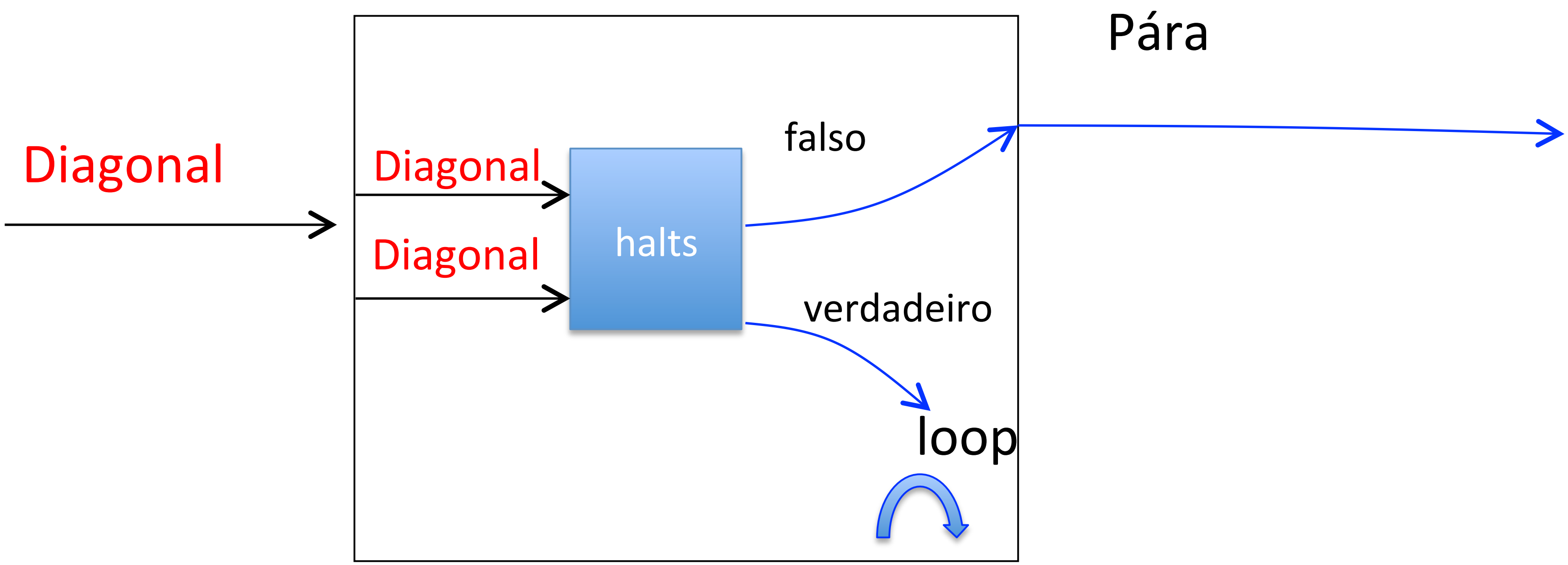
Problema da Parada

Agora, o que acontece quando diagonal é chamado com o seu próprio código?

diagonal (diagonal)

a: se halts (diagonal,diagonal) então goto a, senão pare

Diagonal



Pára
Diagonal
Diagonal
Diagonal
halts
falso
verdadeiro
loop

Problema da Parada

Agora, o que acontece quando diagonal é chamado com o seu próprio código?

- A função diagonal(diagonal) pára?

Problema da Parada

- Se diagonal(diagonal) **pára** (pela execução de halts) então diagonal(diagonal) **não pára**
- Se diagonal(diagonal) **não pára** (pela execução de halts) então diagonal(diagonal) **pára**

Problema da Parada

- Se diagonal(diagonal) **pára** (pela execução de halts) então diagonal(diagonal) **não pára**
- Se diagonal(diagonal) **não pára** (pela execução de halts) então diagonal(diagonal) **pára**

→ Chegamos numa contradição.

→ Logo a hipótese inicial é falsa: o programa **halts(P,X)** não existe, nem muito menos o programa **diagonal.**

Problema da Parada

Problema da parada é um problema de decisão que pode ser declarado informalmente da seguinte forma:

''Dadas uma descrição de um programa P e uma entrada X, decidir se o programa P termina de rodar ou rodará indefinidamente com a entrada X''

Problema da parada é um problema **indecidível**

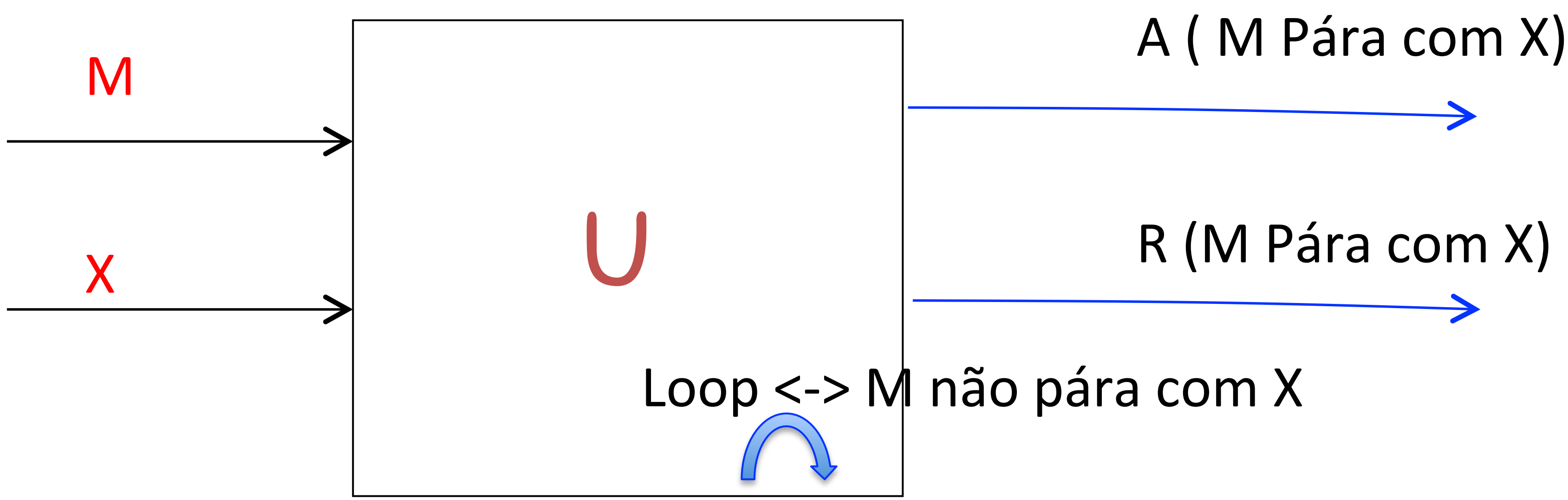
- **O que é problema de decisão?**
- **O que é problema indecidível?**

Uma linguagem não recursiva

H = {“M” “x” : a mT M pára na entrada X}

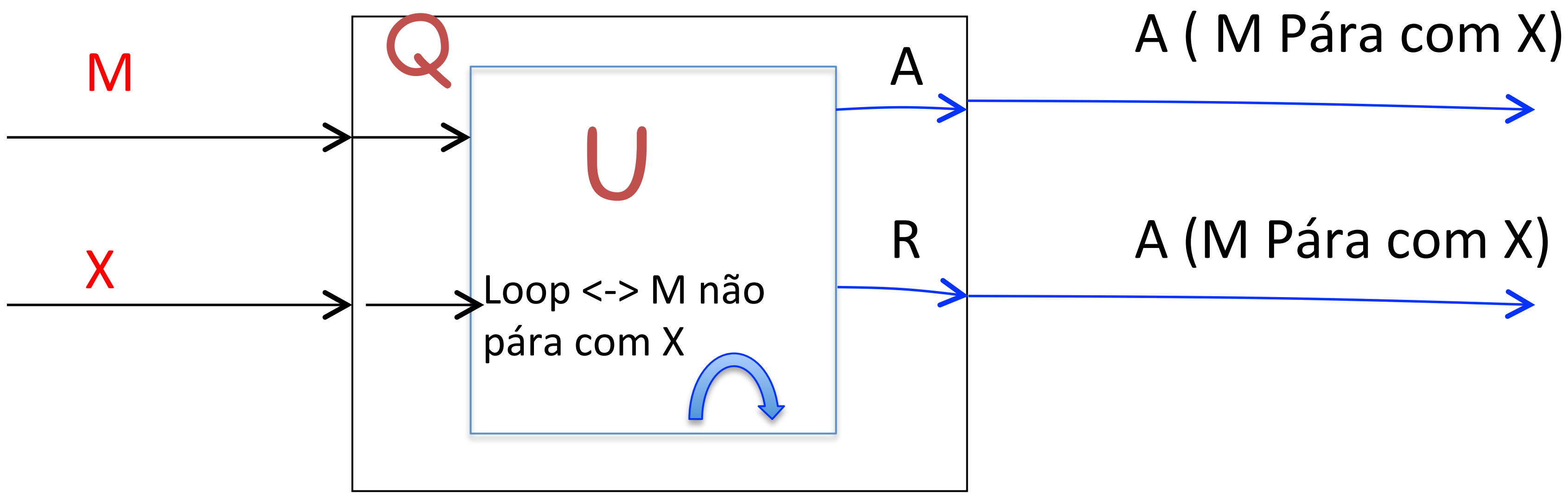
- Esta linguagem é reconhecida por alguma mT?
- H é uma linguagem **estritamente** r.e., ela é semidecidida com o apoio da mT U (universal)
- De fato, na entrada “M” “x”, U pára precisamente quando a entrada está em H.
- H é o **problema da parada** na forma de linguagem.

U como reconhecedora da Linguagem H: versão 1



U pode semidecidir H, mas há um problema no reconhecimento: U rejeita M, x qdo M pára com x

Q como reconhecedora da Linguagem H: versão 2



A máquina Q chama U e inverte a saída da rejeição

Problemas Indecidíveis

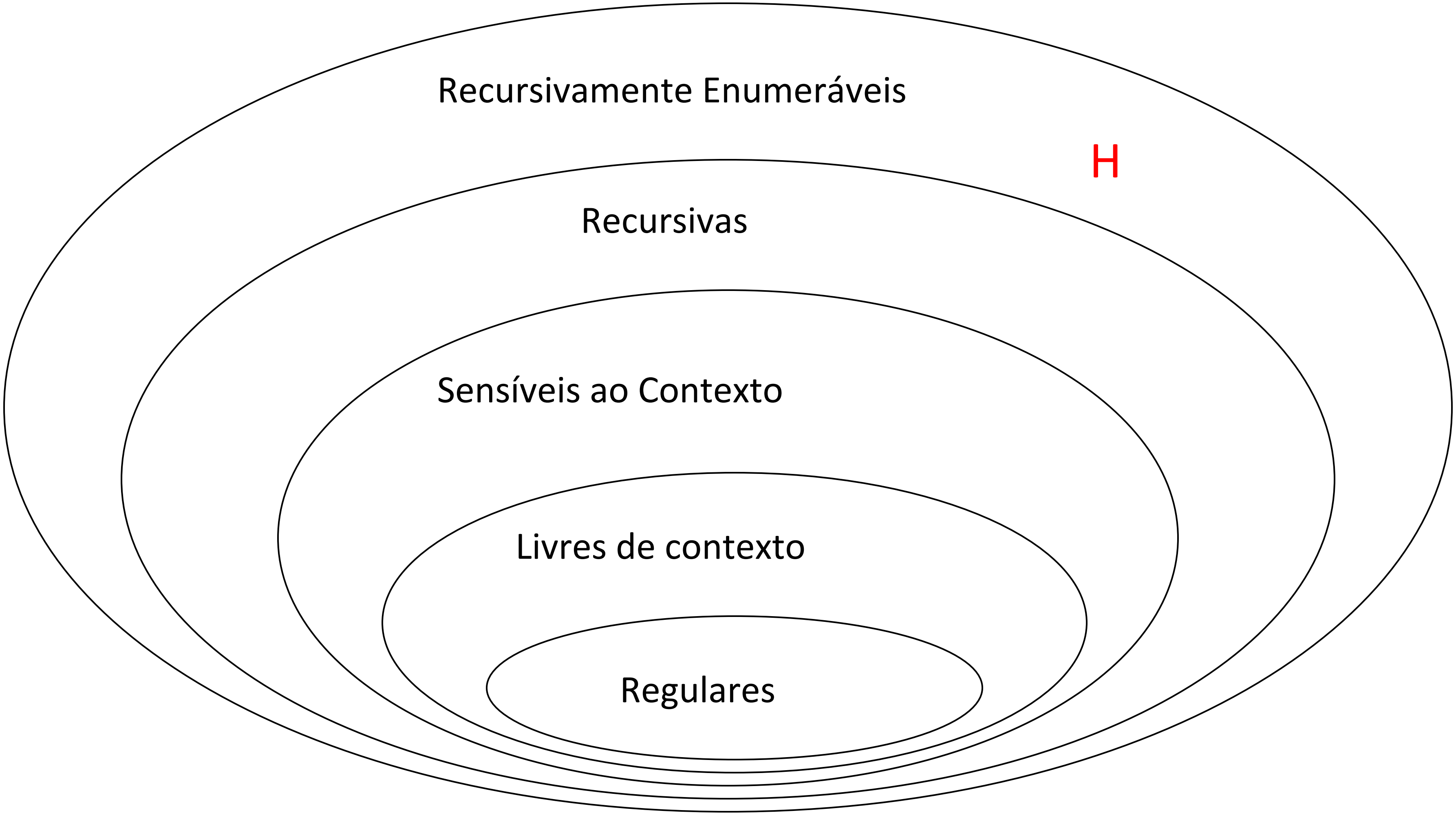
Como H **não é uma linguagem recursiva** e com base na tese de Church-Turing, concluímos que:

"Não há algoritmo que decida, para uma dada arbitrária mT M e uma string de entrada w, se M aceita w ou não." (Problema da Parada)

- Os problemas para os quais não existem algoritmos são chamados de **indecidíveis** ou **insolúveis**.

A Hierarquia de Chomsky com mais detalhes

Linguagens Não Recursivamente Enumeráveis:
não reconhecidas por Mts



Problemas Indecidíveis

Outros problemas não decidíveis:

- Dada uma mT M e uma string de entrada w, como fazer M parar na entrada w
- Dadas duas mTs M_1 e M_2, elas param na mesma entrada?
- Dadas duas gramáticas G_1 e G_2, determinar se $L(G_1) = L(G_2)$
  - Equivalência de Compiladores

Uma linguagem não recursivamente enumerável

$L = \{ x_i : x_i$ não é aceita por $M_i \}$

onde x_i é o código de M_i

Fato: L não é recursivamente enumerável.

Dem.: Vamos mostrar, por contradição, que L não é aceita por nenhuma máquina de Turing. Para isso, suponha que L é aceita pela mT M.

Como toda mT, M faz parte da enumeração das mT's, ou seja, para algum i, $M = M_i$, de forma que $L = L(M_i)$. Vamos verificar se $x_i \in L$:

Uma linguagem não recursivamente enumerável

- se tivermos $x_i \in L$, como $L = L(M_i)$, temos que M_i aceita x_i, e portanto $x_i \notin L$.
- se, alternativamente, tivermos $x_i \notin L$, ou seja, $x_i \notin L(M_i)$, naturalmente, M não aceita x_i, e portanto $x_i \in L$.

Estabelecida a contradição, concluímos que L não é aceita por nenhuma mT M, e que L não é recursivamente enumerável.

(L é conhecida como Linguagem da Diagonal, por que?)

Uma linguagem não recursivamente enumerável

Para a linguagem L (da diagonal) , portanto, não há:

- um procedimento reconhecedor de L
- um procedimento enumerador de L
- uma gramática, tipo 0 ou não, que gere L.

Hierarquia de Chomsky

Tipo	Nome das linguagens geradas	Restrições às regras de produção da gramática $X \rightarrow Y$	Máquinas que aceitam estas linguagens
0	De estrutura de frase = Recursivamente enumeráveis	X = qualquer cadeia com não terminais Y = qualquer cadeia	Máquinas de Turing
	Recursivas		Máquinas de Turing que terminam garantidamente
1	Sensíveis ao contexto	X = qualquer cadeia com não terminais Y = qualquer cadeia de comprimento maior ou igual ao comprimento de X	Máquinas de Turing com fita finita (tamanho proporcional à entrada)
2	Livres contexto	X = qualquer não terminal Y = qualquer cadeia	Autômatos de pilha
	Livres de contexto deterministas		Autômatos de pilha deterministas
3	Regulares	X = qualquer não terminal Y = tN ou Y=t, em que t é um terminal e N é um não terminal	Autômatos finitos

Referências

Esta aula é baseada em:

- Notas de aula do prof. José Lucas Rangel

http://www.inf.puc-rio.br/~inf1626/

- Linguagens Formais e Autômatos – Paulo Blauth Menezes. Editora Bookman.
- Elementos de Teoria da Computação – H. R. Lewis & C.H. Papadimitriou. 2ª. Edição. Editora Bookman.