Linguagens Formais e Autômatos

Decidibilidade

Andrei Rimsa Álvares
andrei@cefetmg.br

Sumário

- Introdução
- A tese de Church-Turing
- Máquinas de Turing e problemas de decisão
- Máquina de Turing Universal
- O problema da parada
- Redução de um problema a outro
- Teorema de Rice

INTRODUÇÃO

Introdução

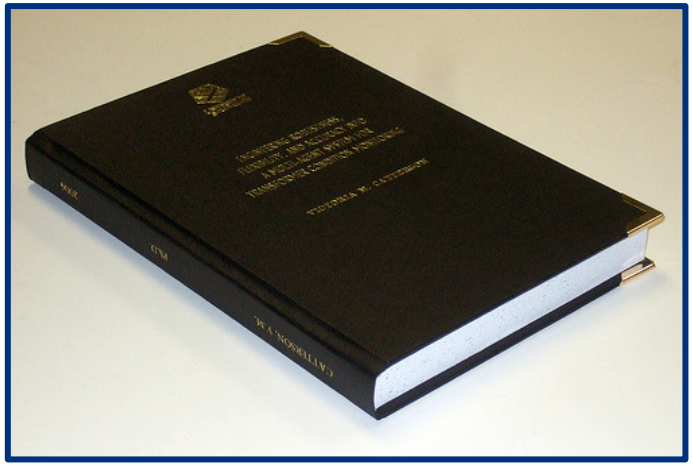
- Máquina de Turing, segundo a tese de Church-Turing, é um formalismo que captura a noção de **computação efetiva**
  - Uma instância de um problema de decisão pode ser codificada (representada) de forma que ela possa ser alimentada como entrada em uma Máquina de Turing
  - Quaisquer Máquinas de Turing podem ser codificadas e apresentadas como entrada para uma Máquina de Turing (chamada de **Máquina de Turing Universal**), que simula qualquer Máquina de Turing fornecida como entrada

Introdução

- Um problema de decisão indecidível é o **Problema da Parada** para Máquinas de Turing - será mostrada uma linguagem que não é LRE, assim como uma linguagem que é LRE mas não é recursiva

- Uma técnica de redução de problemas pode ser usada para mostrar que uma série de problemas são indecidíveis

A TESE DE CHURCH-TURING



Máquinas de Turing

- As Máquinas de Turing (MT) podem ser usadas para
  - Reconhecimento de linguagens (problemas de decisão)
  - Implementação de funções em geral (saída na fita)

- Algumas considerações a respeito do poder computacional das MTs com relação à computação de funções em geral serão discutidas
  - Para o propósito de introdução aos problemas insolúveis, serão consideradas apenas funções que retornam **Sim** ou **Não** (Problemas de Decisão), as quais podem ser *programadas* por meio de MT

Computação Efetiva

- Mesmo antes do aparecimento dos primeiros computadores, os matemáticos já se preocupavam com a noção de **computação efetiva**
  - Mas trabalhavam a partir de uma noção imprecisa (informal), onde especificavam uma lista de atributos desejáveis como a possibilidade de execução mecânica, produção da mesma saída para as mesmas entradas, execução em tempo finito, ...

- A partir de uma caracterização formal, seria possível mostrar que certos problemas são **computáveis**, ou seja, existem **algoritmos** para eles, e que outros não são computáveis

Computação Efetiva

- Vários formalismos foram propostos no intuito de capturar, de forma precisa (formal), o conceito de computação efetiva; todos revelam a mesma expressividade, apesar de suas diferentes *aparências*
  - Máquinas de Turing
  - Sistemas de Post
  - Funções μ-recursivas
  - λ-cálculo

A noção de **computação** é o que existe de comum entre todos eles; ou que cada um deles apresenta uma abordagem diferente para o conceito de **computabilidade**

Poder Computacional

- Os computadores digitais foram construídos com um conjunto de instruções que lhes dão um poder computacional idêntico ao das Máquinas de Turing (e demais formalismos)
  - A máquina de Turing captura a parte "essencial", aquela responsável pelo poder computacional dos computadores atuais

- As linguagens de programação de alto nível (como Java, C, Pascal, ...) possuem o mesmo poder expressivo das Máquinas de Turing; apareceram no intuito de facilitar a tarefa de programação propriamente dita

Linguagens de programação são muito mais adequadas do ponto de vista prático para a confecção de algoritmos

A Tese de Church-Turing

- A tese de Church-Turing pode ser enunciada como

Se uma função é efetivamente computável, então ela é computável por uma Máquina de Turing

Ou seja, todo algoritmo pode ser expresso mediante uma Máquina de Turing

- Embora a noção de computação efetiva não tenha sido formalmente definida, nunca houve um formalismo mais expressivo que o da Máquina de Turing; como os formalismos e linguagens de programação são equivalentes, a tese implica que

"Se uma função é efetivamente computável, então ela é computável por meio de um programa escrito em C"

A Tese de Church-Turing e Problemas de Decisão

- A tese de Church-Turing, quando particularizada para problemas de decisão, pode ser enunciada assim

Se um problema de decisão tem solução, então existe uma Máquina de Turing que o soluciona

  - Assim, para mostrar que um problema de decisão não tem solução, basta mostrar que não existe uma Máquina de Turing que o soluciona

A Tese de Church-Turing e Problemas de Decisão

- A tese de Church-Turing, quando particularizada para problemas de decisão, pode ser enunciada assim

Se um problema de decisão tem solução, então existe uma Máquina de Turing que o soluciona

  - Dada a equivalência dos diversos formalismos, um problema de decisão não tem solução se não for possível expressar uma solução em qualquer um deles
    - Ex.: se não existir um programa em C que seja solução para um problema de decisão, então tal problema de decisão é insolúvel

Autorreferência

- Um algoritmo pode receber outro algoritmo como entrada
  - Um programa (em LP) pode receber um programa como entrada
  - Uma MT pode receber uma MT como entrada

- Essa característica propicia a possibilidade de construir uma máquina universal, ou seja, uma MT (ou programa em uma LP) que seja capaz de simular uma MT (programa) qualquer como entrada

Essa máquina é conhecida como
Máquina de Turing Universal

Autorreferência

- A possibilidade de **autorreferência** é uma característica fundamental que levou à descoberta de funções não computáveis

- Exemplos
  - Não existe MT que determine se uma MT arbitrária irá parar ou não para uma determinada entrada (**problema da parada**)
  - Não existe programa em C que determine se um programa em C irá parar ou não para uma certa entrada

Vários outros problemas similares, que envolvem o processamento de MTs por MTs são insolúveis

Dica: mais sobre o problema da autorreferência: https://youtu.be/HeQX2HjkcNo.

MÁQUINAS DE TURING E PROBLEMAS DE DECISÃO

Linguagens Formais e Autômatos

MTs e PDs

- A solução de um problema de decisão P é um algoritmo que dá a resposta correta para cada instância $p \in P$

- A solução de P pode ser expressa por meio de uma Máquina de Turing que, para cada instância de $p \in P$, se a resposta para p for ***Sim***, para em um estado final, e se a resposta for ***Não***, para em um estado não final
  - A Máquina de Turing deve reconhecer a **linguagem recursiva** que consta de todas as instâncias p para as quais a resposta é ***Sim***

Como representar uma instância de p?

Representação

- Para solucionar um problema de decisão, deve-se projetar uma representação para suas instâncias utilizando um certo alfabeto Σ
  - Para uma única instância não precisa de representação, ou qualquer palavra serve para representá-la (ex.: λ)
  - Para n instâncias (finito), também não precisa de representação, ou quaisquer n palavras servem para representar as n instâncias
  - A representação se torna importante quando se tem uma infinidade de instâncias

- Exemplo para determinar se um número natural é primo: duas representações para "determinar se *j* é primo"

  Representação a) $\Sigma = \{1\}$: 1^j

  Representação b) $\Sigma = \{0,1\}$: número j na base 2

Notação

- Cada instância p de um problema de decisão P pode ser identificada com uma sequência $v_1, v_2, \ldots, v_k$ de valores específicos para os k parâmetros de P (e vice-versa)

- Assim, a representação de p é uma associação com uma palavra de Σ^* que deve satisfazer os seguintes critérios
  a) Para cada instância de P deve existir pelo menos uma palavra de Σ^* que a represente
  b) Cada palavra de Σ^* deve representar no máximo uma instância de P
  c) Para cada palavra $w \in \Sigma^*$, deve ser possível determinar se ela representa ou não alguma instância de P (deve ser decidível)

Notação

- A notação $R<v_1, v_2, \ldots, v_n>$ será utilizada para designar qualquer palavra w que represente a instância $p \in P$ correspondente

- Uma Máquina de Turing que soluciona um Problema de Decisão que recebe como entrada $v_1, v_2, \ldots, v_n$ será representada como

Exemplo

- **Problema:** Determinar se uma GLC G gera uma palavra w
- Representação das instâncias usando $\Sigma = \{0, 1\}$ para a GLC $G = (V, \Gamma, R, P)$, $V = \{X_1, X_2, ..., X_n\}$ e $\Gamma = \{a_1, a_2, ..., a_k\}$
  - **Variável:** $R\langle X_i \rangle = 1^i \ (P = X_1)$
  - **Terminal:** $R\langle a_j \rangle = 1^{n+j}$
  - **Regra:** $R\langle X \rightarrow A_1A_2...A_P \rangle = R\langle X \rangle 0 R\langle A_1 \rangle 0 R\langle A_2 \rangle 0 ... R\langle A_P \rangle$
  - **Regras:** $R\langle \{r_1, r_2, ..., r_q\} \rangle = R\langle r_1 \rangle 00 R\langle r_2 \rangle 00 ... R\langle r_q \rangle$
  - **Gramática:** $R\langle G \rangle = 1^n 0 1^k 0 R\langle \{r_1, r_2, ..., r_q\} \rangle$
  - **Instância:** $R\langle G,w \rangle = R\langle G \rangle 000 R\langle w \rangle$

Como seria a representação para a GLC $H = (\{A, B, C\}, \{a, b\}, R, A)$?

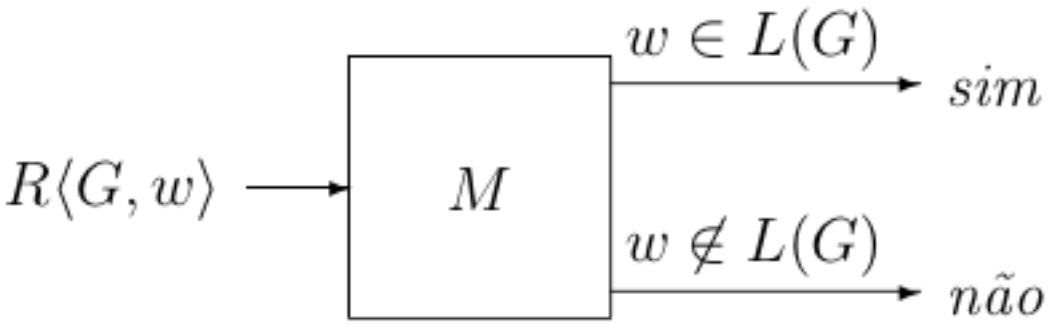
$A \rightarrow aAa \mid B$

$B \rightarrow aB \mid CC$

$C \rightarrow b \mid \lambda$

Exemplo

- Representação esquemática da Máquina de Turing que resolve esse problema de decisão



O problema de decisão tem solução se e somente se a linguagem $L = \{ R<G,w> \mid w \in L(G) \}$ for recursiva

MÁQUINA DE TURING UNIVERSAL

Máquina de Turing Universal

- Uma característica do poder computacional das Máquinas de Turing é a possibilidade de construir Máquinas de Turing capazes de simular qualquer Máquina de Turing
  - Em particular, as máquinas de Turing reconhecedoras de linguagens, como aquelas que solucionam PDs

- Para isso, é preciso definir uma representação para uma Máquina de Turing de entrada

Notação

- Uma possível representação usando o alfabeto $\{0, 1\}$ para uma Máquina de Turing $M = (E, \Sigma, \Gamma, \langle, \sqcup, \delta, i, F)$
  - **Estados** ($E = \{e_1, e_2, ..., e_n\}$):

    $R<e_1 = i> = 1, R<e_2> = 11, \ldots, R<e_n> = 1^n$
  - **Símbolos** ($\Gamma = \{a_1, a_2, ..., a_k\}$):

    $R<a_1 = \langle> = 1, R<a_2 = \sqcup> = 11, \ldots, R<a_k> = 1^k$
  - **Direção:**

    $R<D> = 1$ e $R<E> = 11$
  - **Transição:**

    $R<\delta(e_i, a_j) = [e_i', a_j', d]> = R<e_i>0R<a_j>0R<e_i'>0R<a_j'>0R<d>$
  - **Estados finais** ($F = \{f_1, f_2, ..., f_p\}$):

    $R<F> = R<f_1>0R<f_2>0 \ldots R<f_p>$
  - **Máquina de Turing** (M com transições $\{t_1, t_2, ..., t_s\}$):

    $R<M> = R<F>00R<t_1>00R<t_2>00...R<t_s>$

Exemplo

- Considere a MT $M = (\{0, 1\}, \{a, b\}, \{\langle, \sqcup, a, b\}, \langle, \sqcup, \delta, 0, \{0,1\})$, onde δ
  - $\delta(0, a) = [1, a, D]$
  - $\delta(1, b) = [0, b, E]$

- Representação
  - **Estados:** $R<0> = 1, R<1> = 11$
  - **Símbolos:** $R<\langle> = 1, R<\sqcup> = 11, R<a> = 111, R<b> = 1111$
  - **Transições:**

$R<\delta(0, a)=[1,a,D]> = R<0>0R<a>0R<1>0R<a>0R<D> = 10111011011101$

$R<\delta(1, b)=[0,b,E]> = R<1>0R<b>0R<0>0R<b>0R<E> = 1101111010111011$

  - **Estados finais:** $R<F> = R<0>0R<1> = 1011$
  - **Máquina:** $R<M> = R<F>00R<\delta(0, a)=[1,a,D]>00R<\delta(1, b)=[0,b,E]>$
    $= 10110010111011011010011011110101111011$

Notação

- Uma Máquina de Turing Universal recebe como entrada uma representação de uma Máquina de Turing M e uma representação de uma palavra de entrada w: $R<M, w>$

- Para complementar a representação, deve-se projetar uma representação para $w = a_{i1},a_{i2}...a_{iu}$, palavra de M
  - $R<w> = R<a_{i1}>0R<a_{i2}>0...R<a_{iu}>$

- Por fim, pode-se representar uma instância como
  - $R<M, w> = R<M>000R<w>$

Algoritmo

- Uma MT Universal U pode ser especificada com 3 fitas
  - A primeira fita recebe a entrada de U ($R\langle M, w\rangle$)
  - A segunda irá fazer o papel da fita de M
  - A terceira irá conter apenas a representação do estado atual de M

- A Máquina de Turing Universal U aceita a linguagem

$$L(U) = \{ R<M, w> \mid w \in L(M) \}$$

Algoritmo

1. Se a entrada não é da forma $R\langle M, w\rangle$, pare em estado não final;
2. copie $R\langle w\rangle$ na fita 2 e posicione cabeçote no início;
3. escreva $R\langle i\rangle$ na fita 3 e posicione cabeçote no início;
4. **ciclo**

    4.1 seja $R\langle a\rangle$ a representação sob o cabeçote da fita 2;
    4.2 seja $R\langle e\rangle$ a representação sob o cabeçote da fita 3;
    4.3 procure $R\langle e\rangle 0 R\langle a\rangle 0 R\langle e'\rangle 0 R\langle a'\rangle 0 R\langle d\rangle$ na fita 1;
    4.4 **se** encontrou **então**

        4.4.1 substitua $R\langle e\rangle$ por $R\langle e'\rangle$ na fita 3
            e volte cabeçote da fita 3 ao seu início;
        4.4.2 substitua $R\langle a\rangle$ por $R\langle a'\rangle$ na fita 2;
        4.4.3 mova cabeçote da fita 2 na direção d

    4.5 **senão**

        4.5.1 **se** e é estado final **então**

            4.5.1.1 pare em estado final

        senão

            4.5.1.2 pare em estado não final

        fimse

    fimse

fimciclo.

Reconhecimento por Parada

- Se o reconhecimento for por parada, tem-se duas *simplificações*
  1) Estados finais estão ausentes em $R<M, w>$
  2) O passo 4.5.1 de U será simplesmente: **pare em estado final**

- Essa nova Máquina de Turing pode ser chamada de U_p, onde tem-se

$$U_p \text{ aceita } w \text{ se e somente se } M \text{ para se a entrada é } w$$

  – Ou seja, a linguagem reconhecida por U_p é

$$L(U_p) = \{ R<M, w> \mid M \text{ para se a entrada é } w \}$$

O PROBLEMA DA PARADA

Problema da Parada

- O **problema da parada** para Máquinas de Turing pode ser enunciado como

  Dada uma MT arbitrária M e uma palavra arbitrária w,
  determinar se a computação de M com a entrada w para.

- A existência de uma MT Universal U_p, que simula M para a entrada w, prova que a linguagem $\boldsymbol{L(U_p)}$ **é LRE**

- Ao mostrar que o problema da parada é indecidível, mostra-se que não existe uma MT equivalente à MT U_p que sempre para; ou seja, $\boldsymbol{L(U_p)}$ **não é recursiva**

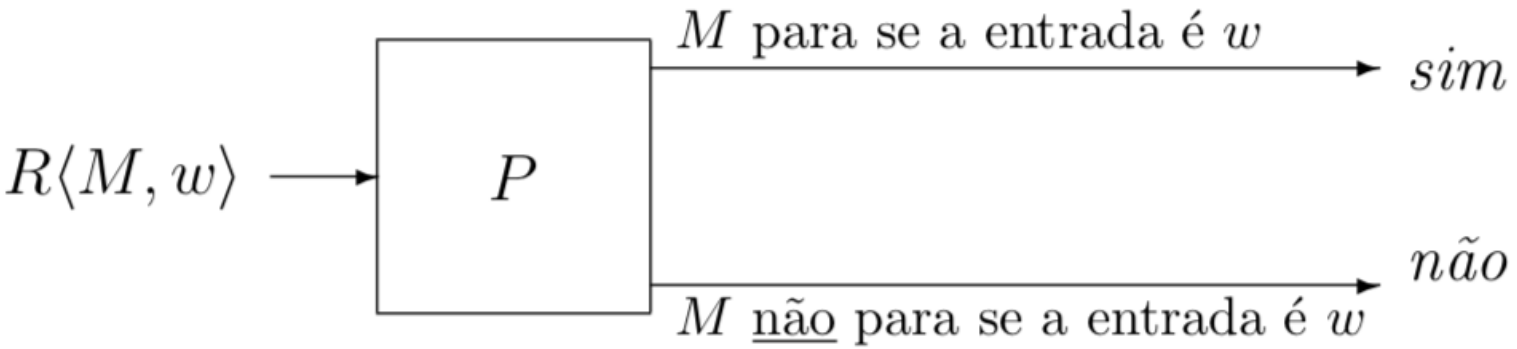
$\overline{L(U_p)}$ é LRE?

Problema da Parada para MTs

- **Teorema:** o problema da parada é indecidível

A prova será feita por contradição

  - Suponha que o problema seja decidível; nesse caso, seja P uma MT que solucionasse o problema (conforme diagrama a seguir)

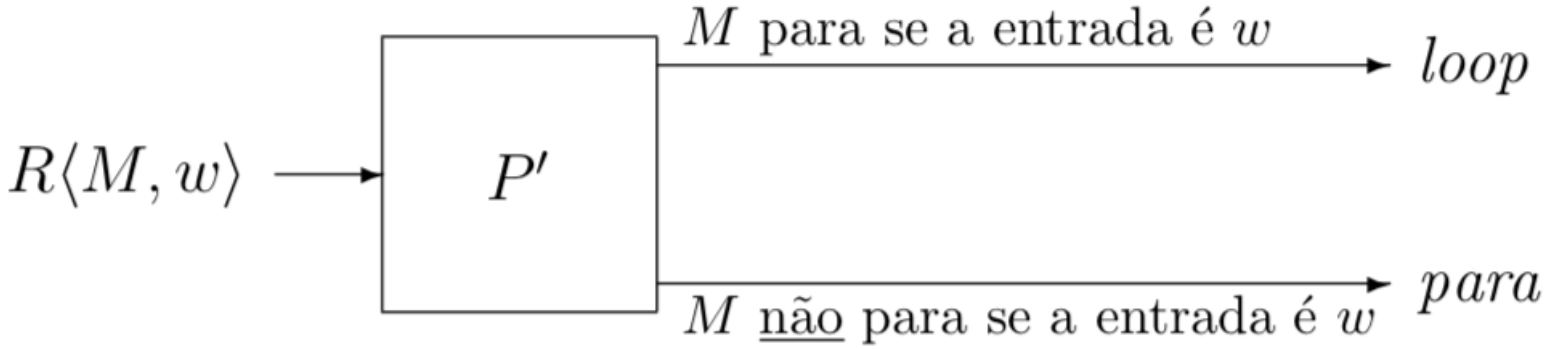


Problema da Parada para MTs

- **Teorema:** o problema da parada é indecidível

Como fazer a MT entrar em loop?

  - A partir dessa Máquina de Turing P seria possível construir uma Máquina de Turing P' cujo comportamento é mostrado a seguir



P' entra em loop se e somente se M para com entrada w

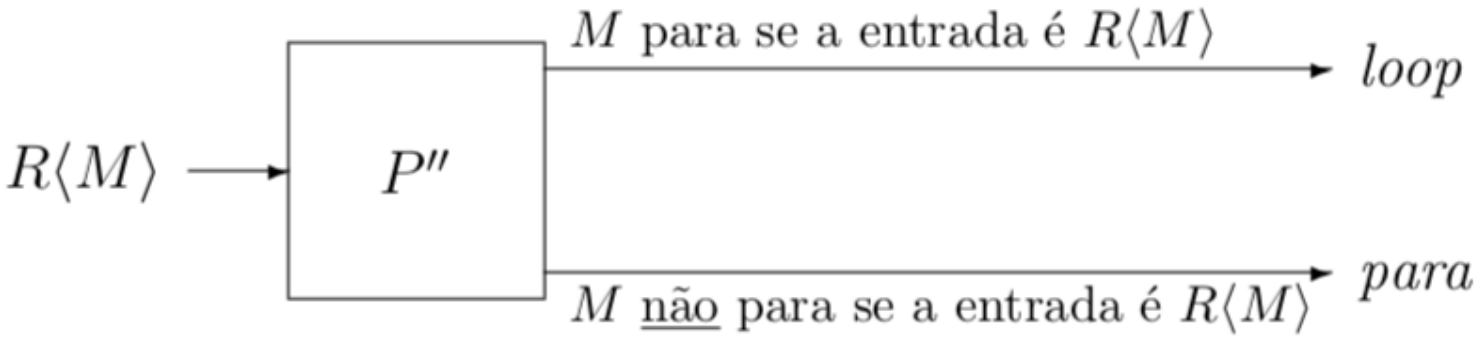
Problema da Parada para MTs

- **Teorema:** o problema da parada é indecidível

  – A partir dessa Máquina de Turing P' seria possível obter uma outra MT P'' (com um parâmetro) a partir de P', tal que

  1) P'' obtém $R\langle M, R\langle M\rangle\rangle$

  2) P'' age como P' sobre essa palavra



Problema da Parada para MTs

- **Teorema:** o problema da parada é indecidível

  - Considere o que acontece se $R\langle P''\rangle$ for submetida como entrada para MT P''
    - Se P'' entra em loop para a entrada $R\langle P''\rangle$, é porque P'' para se a entrada é $R\langle P''\rangle$
    - Se P'' para quando a entrada é $R\langle P''\rangle$, é porque P'' não para se a entrada é $R\langle P''\rangle$

P'' **para** com a entrada $R\langle P''\rangle$
se e somente se
P'' **não para** com a entrada $R\langle P''\rangle$

Contradição! Portanto, o problema da parada é indecidível

Problema da Parada para LPs

- **Teorema:** o problema da parada para linguagens procedurais comuns é indecidível

  – Suponha que a função lógica (booleana) P solucione o PD em questão, de forma que a chamada $P(x, w)$ retorna verdadeiro se e somente se o procedimento de texto x para com a entrada também de texto w; então pode-se escrever o procedimento

```
procedimento P''(x):
    enquanto P(x, x) faça
    fimenquanto
fim P''
```

Novamente, a prova é feita por contradição

Problema da Parada para LPs

- **Teorema:** o problema da parada para linguagens procedurais comuns é indecidível
  - Seja T esse texto do procedimento P'', então
    - Se $P''(T)$ para é porque $P(T, T)$ retorna falso, o que significa que $P''(T)$ não para (**se $P''(T)$ para, então $P''(T)$ não para**)
    - Se $P''(T)$ não para é porque $P(T, T)$ retorna verdadeiro, o que significa que $P''(T)$ para (**se $P''(T)$ não para, então $P''(T)$ para**)

Contradição! Conclui-se que a função P não pode existir

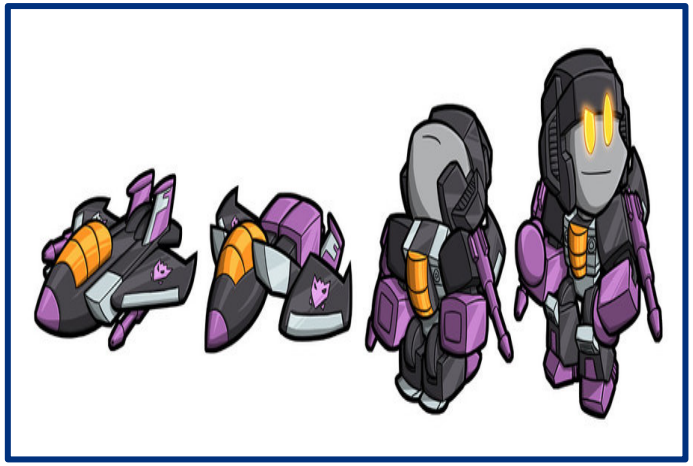
Propriedades

- A partir da existência da Máquina de Turing Universal e da indecidibilidade do problema da parada para MTs, chega-se aos teoremas a seguir para a linguagem

$$L_p = \{ R\langle M, w\rangle \mid M \text{ para se a entrada é } w \}$$

  - **Teorema:** A linguagem L_p não é recursiva
  - **Teorema:** A linguagem L_p é recursivamente enumerável
  - **Teorema:** A linguagem $\overline{L_p}$ não é recursivamente enumerável

REDUÇÃO DE UM PROBLEMA A OUTRO

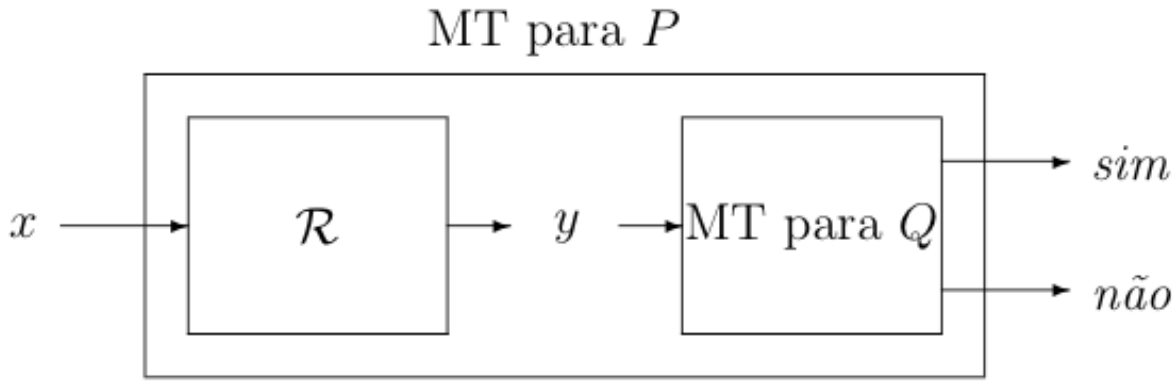


Redução de um Problema a Outro

- Um PD P é redutível a um PD Q, se existe um algoritmo $\mathcal{R}$ que, recebendo x como entrada, produz um resultado y tal que a resposta de P para a entrada x é idêntica ou complementar à resposta de Q para a entrada y, qualquer que seja a entrada x

Diz-se que o algoritmo $\mathcal{R}$ pode ser usado para reduzir o problema P ao problema Q

- O PD P pode ser solucionado mediante o algoritmo $\mathcal{R}$ e um algoritmo para o PD Q



Redução de um Problema a Outro

- A redução de um problema pode ser usada tanto para provar que um problema é **decidível**, quanto para provar que um problema é **indecidível**

  - Para problemas **decidíveis**
    - Se P é redutível a um problema decidível, então P é decidível

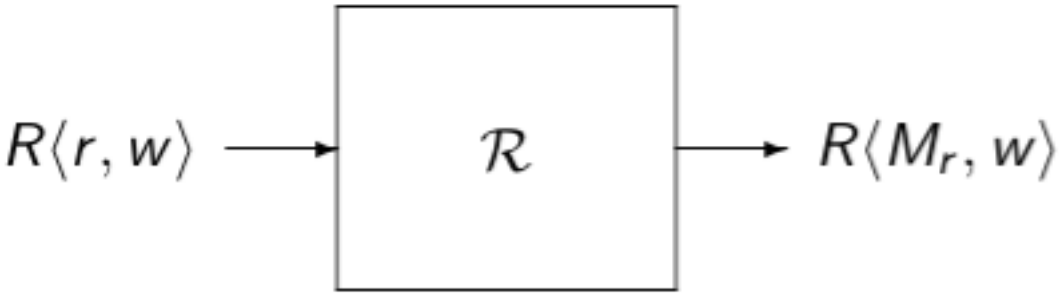
  - Para problemas **indecidíveis**
    - Se um problema indecidível é redutível a um problema *P*, então *P* é indecidível

O que se pode dizer se inverter a redução em cada um desses casos?

Exemplo Decidível

- **Problema:** determinar se $w \in L(r)$, onde r é uma expressão regular arbitrária

  - Pode ser reduzido ao problema de determinar se $w \in L(M)$, em que M é um AFD arbitrário
    - Redução $\mathcal{R}$: MT que constrói um AFD M_r a partir do ER r

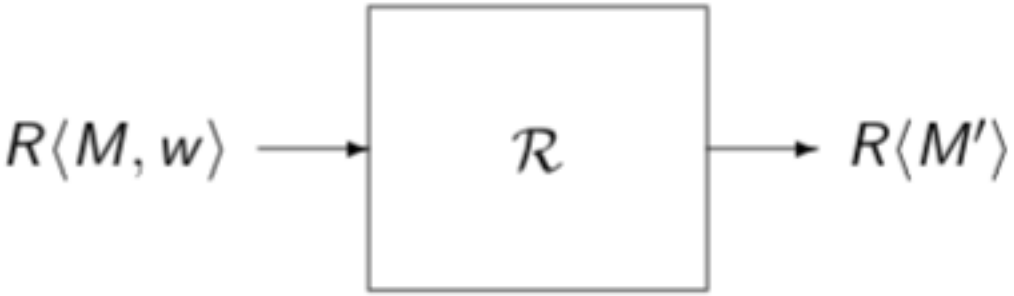


$w \in L(M_r)$ se e somente se $w \in L(r)$

Exemplo Indecidível (Problema da Fita em Branco)

- **Problema:** determinar se $\lambda \in L(M)$ para uma MT M

  - O problema da parada pode ser reduzido a esse
    - Redução: MT $\mathcal{R}$ que constrói uma MT M' a partir da MT M e entrada w



M para se a entrada é w
se e somente se
M' para se a entrada é λ

Exemplo Indecidível (Problema da Fita em Branco)

- **Problema:** determinar se $\lambda \in L(M)$ para uma MT M

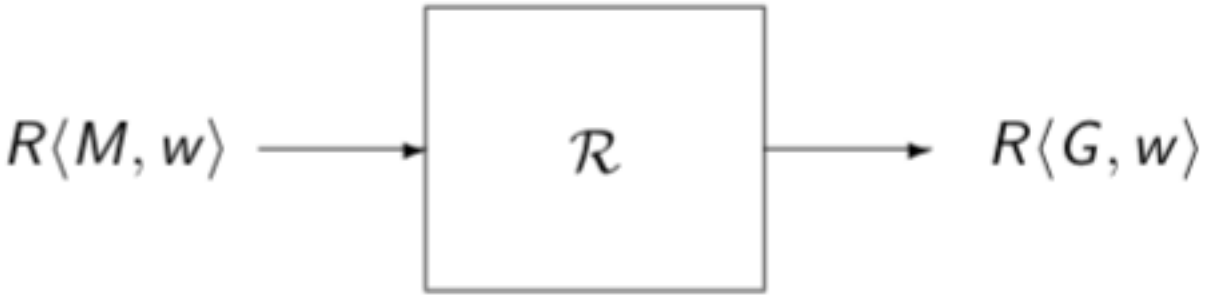
  - A MT $\mathcal{R}$ produz $R\langle M' \rangle$, a partir de $R\langle M, w \rangle$, de forma que
    1) M' escreve w
    2) M' volta o cabeçote para o início da fita
    3) M' se comporta como M

M para se a entrada é w
se e somente se
M' para se a entrada é λ

Outro Exemplo Indecidível

- **Problema:** determinar se $w \in L(G)$ para uma gramática irrestrita *G* e $w \in \Sigma^*$

  - O problema da parada pode ser reduzido a esse
    - Redução: MT $\mathcal{R}$ que constrói uma MT G a partir da MT M



M para se a entrada é w
se e somente se
G gera w

Critério de Reconhecimento para o Problema da Parada

- A partir de agora, será assumido que o critério de reconhecimento, para todas as Máquinas de Turing, é o da **parada**; ou seja, que todos os estados da máquina são finais

A MT M para se sua entrada é w se e somente se $w \in L(M)$

- Assim, $L(M)$ será o mesmo que $L_p(M)$

Classe de Problemas Indecidíveis

- Uma classe de problemas de decisão engloba todos os problemas de decisão cujo único parâmetro é uma Máquina de Turing arbitrária

- Todos os problemas dessa classe podem ser provados como **indecidíveis**; já que todos têm o seguinte tipo de enunciado

Determinar se a linguagem aceita por uma Máquina de Turing arbitrária M satisfaz a propriedade P

($\{ R<M> \mid L(M)$ satisfaz $P \}$ é recursiva?)

  - Ex.: o problema da fita em branco pode ser enunciado como

Determinar se a linguagem aceita por uma Máquina de Turing arbitrária M é tal que $\lambda \in L(M)$

($\{ R<M> \mid \lambda \in L(M) \}$ é recursiva?)

Classe de Problemas Indecidíveis

- Para que todos os PD's da classe referida sejam indecidíveis, basta expurgar dois tipos que são, trivialmente, decidíveis
  - Aqueles em que a propriedade P é sempre verdadeira
  - Aqueles em que a propriedade P é sempre falsa

- Ou seja,
  - Aqueles em que $\{ R\langle M\rangle \mid L(M)$ satisfaz $P \}$ é o conjunto de todas as representações de Máquinas de Turing
  - Aqueles em que $\{ R\langle M\rangle \mid L(M)$ satisfaz $P \} = \emptyset$

Propriedades Triviais

- **Definição:** uma propriedade P de LREs é trivial se é satisfeita por toda LRE ou por nenhuma

- Por exemplo
  - Para o problema da fita em branco, a propriedade $\lambda \in L(M)$ não é trivial: algumas LREs contêm $\lambda \in L(M)$, enquanto outras não

- Outros exemplos
  - Determinar se $L(M)$ contém alguma palavra
  - Determinar se $L(M)$ contém todas as palavras em Σ^*
  - Determinar se $L(M)$ é finita
  - Determinar se $L(M)$ é regular
  - Determinar se $L(M)$ contém palavra que começa com *0*

Redução para Mais um Problema Indecidível

- **Problema:** determinar se $L(M) \neq \emptyset$ para uma MT arbitrária M

  - O problema da parada será reduzido a esse
    - Redução: A MT $\mathcal{R}$ produz $R\langle M' \rangle$, a partir de $R\langle M, w \rangle$, de forma que
      1) M' apaga a entrada
      2) M' escreve w
      3) M' volta o cabeçote para o início da fita
      4) M' se comporta como M
  - Com isso, tem-se que

M para se a entrada é w se e somente se M' para com alguma entrada

Como seria a prova para $L(M) = \Sigma^*$

TEOREMA DE RICE

Teorema de Rice

- **Teorema:** se P é uma propriedade não trivial de LREs, então $\{ R\langle M\rangle \mid L(M)$ satisfaz $P \}$ **não é recursiva**

  - Seja P uma propriedade não trivial, a prova será feita para os seguintes dois casos

    Caso 1) $\emptyset$ não satisfaz P

    Caso 2) $\emptyset$ satisfaz P

Caso 1: ∅ não satisfaz P

- Seja uma MT M_x, tal que $L(M_x)$ satisfaz P
  (M_x existe, pois P é não trivial e $L(M_x) \neq \emptyset$)

  – O problema da parada pode ser reduzido ao de determinar se $L(M')$ satisfaz P produzindo-se $R\langle M' \rangle$ a partir de $R\langle M, w \rangle$

    1) M' escreve w na fita após sua entrada; suponha que a fita fique como $\langle x[w\sqcup...$

    2) M' se comporta como M sobre $[w\sqcup...$

    3) Quando M' para, na situação em que M para, M' se comporta como M_x sobre $\langle x\sqcup...$

Caso 1: Ø não satisfaz P

- Nesse caso, observa-se que

  $L(M')$ satisfaz P se e somente se M para com a entrada w

  - Se M para com a entrada w, $L(M') = L(M_x)$; portanto $L(M')$ satisfaz P
  - Se M não para com a entrada w, $L(M') = \emptyset$; dada a suposição inicial desse caso, então $L(M')$ não satisfaz P

- Assim, conclui-se que o problema de determinar se $L(M)$ satisfaz P, para MTs arbitrárias M, é indecidível

  $\{ R\langle M\rangle \mid L(M)$ satisfaz $P \}$ **não é recursiva**

Caso 2: ∅ satisfaz P

- Nesse caso, ∅ não satisfaz $\neg P$; como P não é trivial, $\neg P$ também não é trivial

  - Pela argumentação do caso 1, $\{ R\langle M\rangle \mid L(M)$ satisfaz $\neg P \}$ não é recursiva; como essa linguagem é o complemento de $\{ R<M> \mid L(M)$ satisfaz $P \}$, e as linguagens recursivas são fechadas sob complementação, segue-se que

$$\{ R<M> \mid L(M) \text{ satisfaz } P \} \textbf{ não é recursiva}$$

ISSO É TUDO, PESSOAL!