Linguagens Formais e Autômatos

Aula 26 - O problema da parada da Máquina de Turing

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

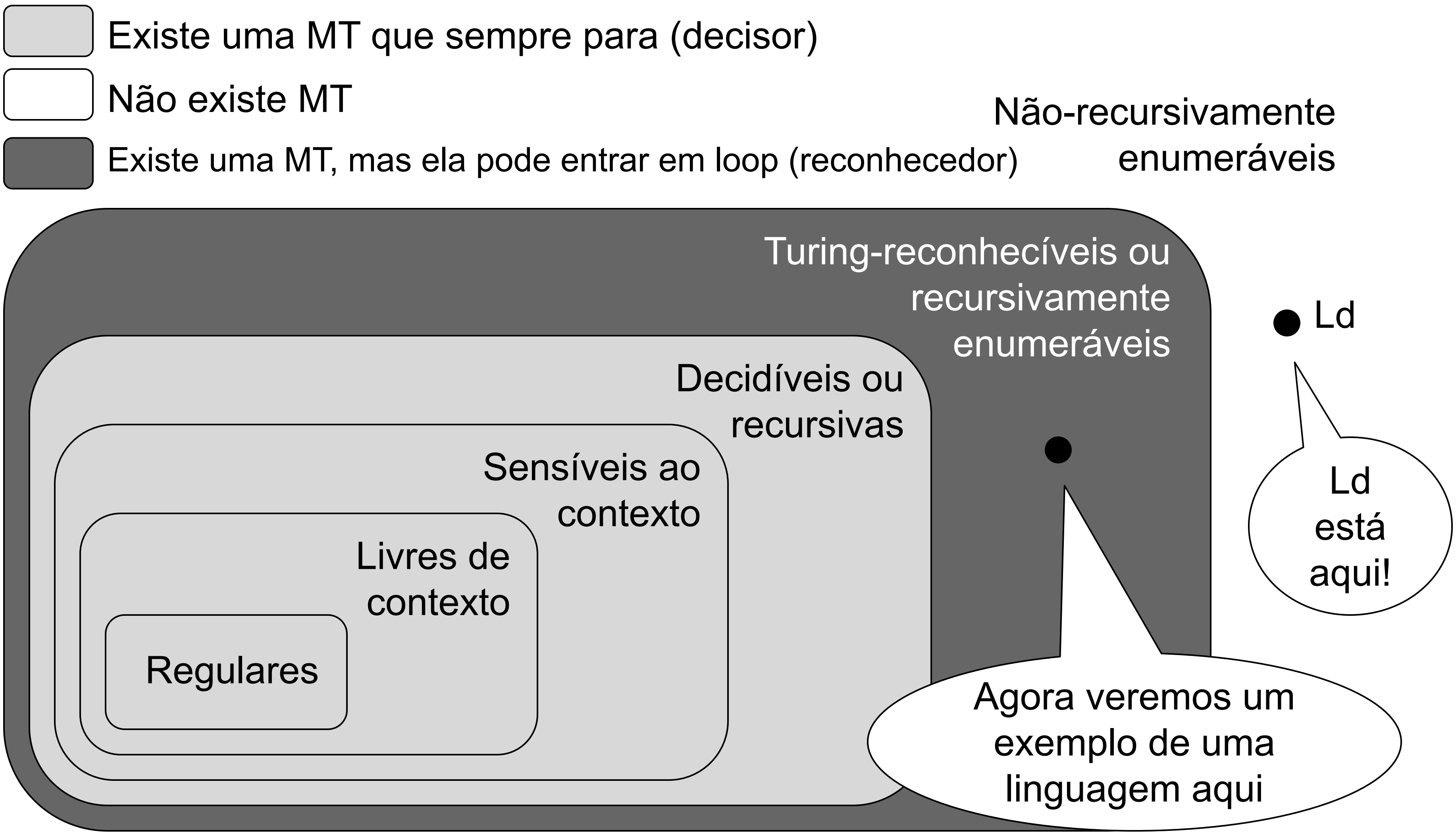
Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
  - Capítulo 9 - Seção 9.2
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
  - Capítulo 4 - Seção 4.2

Um problema indecidível que é RE

- Agora iremos refinar a estrutura das linguagens RE (ou Turing-reconhecíveis)
- Dividiremos em duas classes:
  - Algoritmos: problemas para os quais existe uma MT que reconhece a linguagem, mas também reconhece (decide) as cadeias que não pertencem à linguagem.
    - São as MTs que sempre param, independente do fato de alcançar ou não um estado de aceitação
  - Linguagens RE que não são aceitas por nenhuma máquina de Turing com garantia de parada
    - Inconvenientes: se a entrada estiver na linguagem, saberemos disso mas, se a entrada não estiver na linguagem, a MT poderá continuar funcionando para sempre, e nunca teremos certeza de que a entrada não será aceita mais tarde

Hierarquia de linguagens



Linguagens recursivas

- Uma linguagem L é recursiva se L = L(M) para alguma máquina de Turing M tal que:
  - Se w está em L, então M aceita (e portanto para)
  - Se w não está em L, então M para eventualmente, embora nunca entre em um estado de aceitação
- Uma MT desse tipo corresponde à nossa noção informal de um “algoritmo”
  - Nesse caso, L é um “problema” decidível, ou seja, existe um algoritmo que o resolve
- Na prática, a divisão entre recursiva/não-recursiva é mais importante do que a divisão RE/não-RE

Complementos de linguagens recursivas e RE

- Existem alguns teoremas bastante simples de se provar, úteis nas demonstrações a seguir:
- Teorema: Se L é uma linguagem recursiva, $\tilde{L}$ também o é.
- Prova:
  - Se L é recursiva, existe uma MT que sempre para, aceitando ou não a entrada
  - Basta modificar MT, de forma a inverter aceitação/não-aceitação, e iremos obter uma MT’ que também sempre para (a modificação não altera esse fato)
  - MT’ irá aceitar quando MT rejeita, e rejeitar quando MT aceita, reconhecendo exatamente o complemento de L
  - Ou seja, $\tilde{L}$ é recursiva, pois existe um decisor

Complementos de linguagens recursivas e RE

- Teorema: Se L e seu complemento são ambas RE, então L é recursiva (assim como seu complemento, pelo teorema anterior)
- Prova:
  - Se L é RE, existe uma MT1 que sempre para aceitando quando a entrada é um w que pertence a L (embora possa não parar nunca caso não pertença)
  - Se ~L é RE, existe uma MT2 que sempre para aceitando quando a entrada é um w que não pertence a L (embora possa não parar nunca caso pertença)
  - Basta construir uma MT' que simula MT1 e MT2, aceitando quando MT1 aceitar (e parar), e rejeite quando MT2 aceitar (e parar)
  - Dessa forma, MT' sempre para, aceitando ou rejeitando, portanto L é recursiva.

Complementos de linguagens recursivas e RE

- Ou seja, existem apenas quatro possibilidades:
  - L e ~L são ambas recursivas
  - Nem L nem ~L é RE
  - L é RE mas não-recursiva, e ~L não é RE
  - ~L é RE mas não recursiva, e L não é RE
- Todas as outras possibilidades são excluídas pelos teoremas anteriores
- Exemplo: L_d não é RE, e portanto ~L_d não pode ser recursiva
  - Ou seja, pode até existir uma MT para ~L_d, mas não há garantia de que ela vá parar sempre

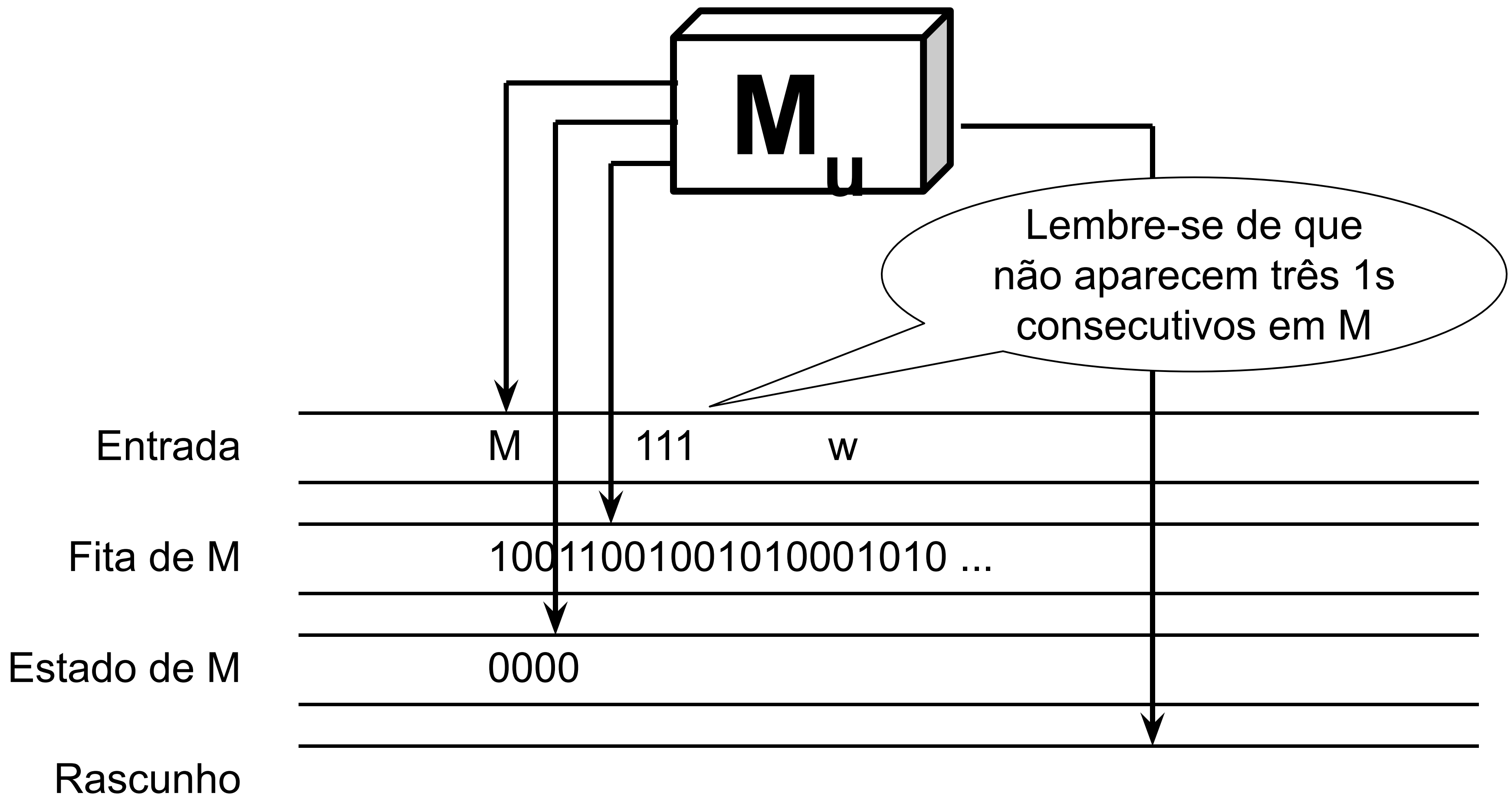
Complementos de linguagens recursivas e RE

- Esses resultados são intuitivos
  - Suponha que exista um problema indecidível (como o PCP)
  - Suponha que eu conseguisse resolver a versão “negada” do PCP
    - Ou seja, encontrar as instâncias do PCP para as quais não existe solução
    - Bastaria inverter a resposta, e pronto, resolvi um problema insolúvel!
    - Mas isso é impossível, já que o PCP é indecidível!

A linguagem universal

- Vimos anteriormente que uma MT pode simular um computador, executando um programa armazenado
  - Utilizando várias fitas, que armazenam o programa, os dados, rascunho, etc...
- E se esse programa armazenado for uma máquina de Turing?
  - Mais especificamente, a codificação binária que vimos em aula anterior?
  - É possível construir uma máquina de Turing que executa máquinas de Turing codificadas?

Máquina de Turing universal



Máquina de Turing universal

- M_u opera da seguinte forma:
  - Examina a entrada para ter certeza que é um código válido
  - Inicializa a segunda fita para conter a entrada w
  - Insere 0 (estado inicial de M) na terceira fita e move a cabeça da segunda fita de M_u para a primeira célula simulada
  - M_u procura na primeira fita uma transição $0^i10^j10^k10^l10^m$, olhando:
    - Na fita 3 em busca de i
    - Na fita 2 em busca de j

Máquina de Turing universal

- M_u opera da seguinte forma:
    - Encontrando a transição $0^i10^j10^k10^l10^m$:
        i. Muda o conteúdo da fita 3 para 0^k
        ii. Substitui 0^j na fita 2 por 0^l (usando o rascunho para fazer o deslocamento e administrar o espaço)
        iii. Move a cabeça na fita 2 para a posição do próximo 1 à esquerda ou direita, dependendo de m (m=1 → esquerda, m=2 → direita)
    - Se M não tem nenhuma transição, M_u para
    - Se M entrar em estado de aceitação, M_u aceita

A linguagem universal

- A entrada da MT universal é um par (M,w), onde M é uma MT codificada em binário e w é uma string binária
- Em alguns casos, M aceita w, em outros, M não aceita w
- O conjunto de todos os pares (M,w), tal que M aceita w é conhecido como linguagem universal, ou L_u

A linguagem universal

- Qual é o “problema” descrito pela linguagem universal?
    - Dada uma máquina de Turing M e uma cadeia qualquer w, determinar se M aceita w
- Pensando em termos mais práticos:
    - Dado um programa de computador, e as possíveis entradas e saídas, esse “problema” consiste em verificar, automaticamente (algoritmicamente), se o programa funciona conforme o esperado
    - Se for possível resolver esse problema, eliminaríamos a necessidade de testes de software!!!
        - Bastaria construir um algoritmo (ou MT, ou programa) que fosse um verificador automático

A linguagem universal

- Se a linguagem universal fosse decidível:
    - Não teríamos catástrofes como a do foguete Ariane-5
    - A NASA não perderia 125 milhões de dólares com o Mars Climate Orbiter
    - O Windows não teria bugs
    - Nossa vida de programadores seria muito mais fácil
- Mas..... (como você já deve ter adivinhado)
    - L_u é indecidível!
    - É RE! Mas não recursiva!!
    - Ou seja, não existe decisor
    - Não existe algoritmo
    - Precisamos testar nossos programas
    - Teremos muitos e muitos bugs pela frente

Indecidibilidade da linguagem universal

- Teorema: L_u é RE mas não é recursiva
- A prova de que L_u é RE já foi feita
  - M_u existe (mostramos anteriormente)
  - Portanto L_u é Turing-reconhecível
  - Portanto L_u é RE
- Continuando
  - Vamos supor que L_u fosse recursiva (buscaremos uma contradição)
    - Então, $\sim L_u$ (o complemento de L_u) também deve ser recursiva
    - Ou seja, existe uma MT M_{nu} que aceita $\sim L_u$

Indecidibilidade da linguagem universal

- Vamos supor que exista M_{nu}, tal que $L(M_{nu}) = {\sim}L_u$
  - Ou seja, dada uma entrada (M,w), M_{nu} aceita se M rejeita w, e M_{nu} rejeita se M aceita w
- E se eu aplicar, como entrada para M_{nu}, a entrada w11w? Ou melhor, um par (w,w)?
  - Ou melhor: um par (M,M)
  - Ou seja, w é um determinado w_i, da nossa enumeração de M_i's anterior
    - Ou seja, w codifica uma MT qualquer

Indecidibilidade da linguagem universal

- Vemos então que M_{nu} é uma máquina bastante poderosa!!!
  - O que significa M_{nu} aceitar w111w?
    - Significa que a máquina M rejeita a si mesma como entrada
  - O que significa M_{nu} rejeitar w111w?
    - Significa que a máquina M aceita a si mesma como entrada
- Peraí: essa é a linguagem L_d!!
  - Quer dizer que M_{nu} decide a linguagem L_d?
  - Mas L_d é indecidível!!
  - Exatamente: aí está a contradição!!
  - Mnu não pode existir!
  - Ou seja, L_u é não-RE!
  - Ou seja, L_u não é recursiva!!

O problema da parada

- Semelhante à L_u, mas mais genérico
    - Seja H(M) o conjunto de entradas w tais que uma MT M para em w (aceitando ou não)
    - O problema da parada da MT é:
        - Dado um par (M,w), decidir se M para em w ou não
- É o mesmo problema
    - Também é RE, mas não recursivo
    - Ou seja, indecidível

Fim

Aula 26 - O problema da parada da Máquina de Turing