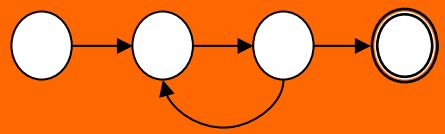
Automata e Linguagens Formais

CTC 34



9

Prof. Carlos H. C. Ribeiro

carlos@ita.br

Linguagens irrestritas e LREs

Computabilidade

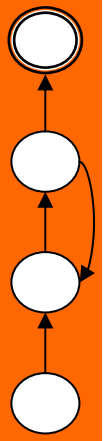
Problemas de decisão

A Tese Church-Turing

O Problema da parada

Linguagens Irrestritas

- As linguagens irrestritas formam a classe de linguagens aceitas pelas Máquinas de Turing.
- Uma linguagem irrestrita corresponde à linguagem gerada por uma Gramática Irrestrita (ou seja, são as linguagens do tipo 0).
- Analogamente aos teoremas de Kleene e Chomsky:



Linguagens Recursivas

- Problema: Para cadeias não aceitas por uma MT, pode não ocorrer a parada...
- Defino então:
  - Uma linguagem L é dita recursiva se for aceita por pelo menos uma MT que produz parada (com ou sem aceitação) para qualquer entrada.

Teorema: A classe das linguagens recursivas é um subconjunto da classe de linguagens irrestritas.

Linguagens Recursivas: Propriedades

- O complemento de uma linguagem recursiva é uma linguagem recursiva.

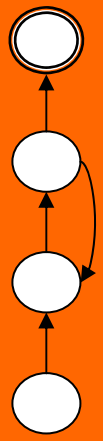
Prova: Seja L ling. recursiva e M a MT que a aceita com parada para qualquer entrada. Construa M' a partir de M como:

  - Se M entra em estado final com entrada w, M' pára sem aceitação.
  - Se M pára sem aceitação, M' entra em estado final.

As cadeias aceitas por M são exatamente aquelas não aceitas por M', e as cadeias aceitas por M' sao exatamente as não aceitas por M. Assim, L(M') é o complemento de L(M). Como M' produz parada p/ qualquer cadeia e aceita L(M'), então L(M') é recursiva.

Ilustração: Hopcroft/Ullman, pag. 180.

- A união de linguagens recursivas é uma linguagem recursiva.
- A união de linguagens irrestritas é uma linguagem irrestrita.

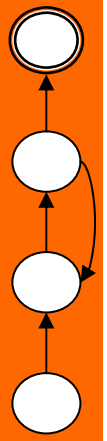


Linguagens Formais e Modelos Computacionais

- Existem dispositivos computacionais com capacidades intermediárias entre autômatos finitos e máquinas de Turing, que reconhecem linguagens do tipo 2 (livres de contexto) e tipo 1(sensíveis ao contexto).
- Os **autômatos de pilha** (já vimos) reconhecem linguagens do tipo 2 (livres de contexto).
- Os **autômatos limitados lineares** (máquina de Turing cujo cursor de leitura/gravação está limitado à parte da fita que contém a entrada original) reconhecem linguagens do tipo 1(sensíveis ao contexto).

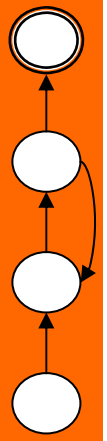
Linguagens Formais e Modelos Computacionais

Linguagem	**Gramática**	**Disp. Computacional**
irrestrita	irrestrita	Máquina de Turing
sensível ao contexto	sensível ao contexto	Autômato limitado linear
livre de contexto	livre de contexto	Autômato de Pilha
regular	regular	Autômato Finito



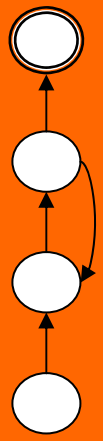
Problemas de Decisão

- Um problema de decisão P é um conjunto de questões, cada uma com uma resposta S ou N.
- Uma solução para um problema de decisão P é um algoritmo que determina a resposta apropriada para toda questão p $\in$P.
- E o que é um algoritmo? Dificil de definir, mas algumas propriedades sao claras:
  - Um algoritmo é completo: produz resposta S ou N para cada questao do problema.
  - Um algoritmo é mecânico: consiste de uma sequência finita de instruções não-ambíguas.
  - Um algoritmo é determinístico: produz sempre a mesma resposta para uma dada entrada.



Decisão × Otimização

- Muitos problemas de interesse são **problemas de otimização**, onde cada solução tem um valor associado, e deseja-se encontrar a solução com o melhor valor.
  - Ex: caminho mais curto entre dois nós em um grafo.
- **Problemas de decisão** são aqueles cuja resposta é SIM / NÃO (0 ou 1).
- Problemas de otimização podem ser facilmente convertidos em problemas de decisão.
  - Ex.: Ao invés de procurar pelo caminho mais curto entre dois vértices em um grafo, pergunta-se se existe um caminho com comprimento < k e varia-se k.



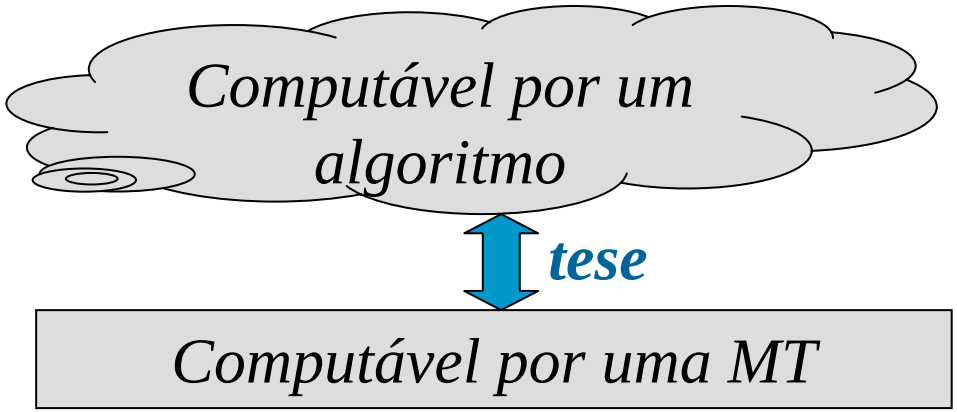
MTs e Algoritmos

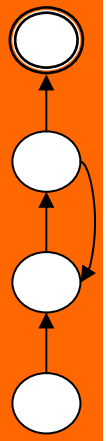
- Uma MT:
  - É mecânica: consiste de uma sequência finita de instruções não ambíguas.
  - É determinística: produz sempre a mesma resposta para uma dada entrada.
  - Se o problema de decisão a ser resolvido puder ser expresso como uma linguagem recursiva, a MT correspondente é completa: produz resposta S (equiv. aceitação com parada) ou N (equiv. a rejeição com parada) para cada questão do problema.
- Nestas condições, **uma MT seria uma representação computacional formal de um algoritmo**...

Tese de Church-Turing

- Em 1936, com os artigos publicados por Turing e Alonzo Church, é que se definiu claramente o que de fato seria um **algoritmo**: **é uma MT que pára em respostas a todas as entradas.**
- Tese de Church-Turing:

"Qualquer problema de decisão é computável por um algoritmo se, e somente se, é computável por uma Máquina de Turing."





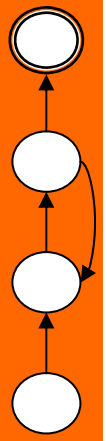
A Tese Church-Turing Estendida

Def.: Um problema de decisão P é parcialmente computável se existir um procedimento mecânico e determinista que produz a resposta S para cada questão de P de resposta S, mas que pode não produzir resposta para questões cuja resposta é N.

Tese: Um problema de decisão P é parcialmente computável se existe uma MT que aceita exatamente as questões de P cuja resposta é S.

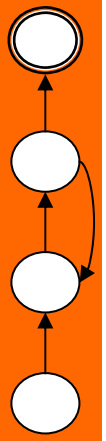
Alonzo Church (14/06/1903 – 11/08/1995)

Matemático americano, responsável por alguns dos fundamentos da CC. Nasceu em Washington, DC, recebeu BA e PhD na Universidade de Princeton, USA, em 1924 e 1927, respectivamente. Tornou-se professor em Princeton em 1929. Foi orientador de Stephen C. Kleene (1909-1994), entre outros. Também orientou Alan Turing de 1936 a1938. Seu trabalho mais conhecido é o desenvolvimento do cálculo-λ no seu famoso artigo de 1936, mostrando a existência de problemas indecidíveis. Ele e Turing mostraram que o cálculo- λ e a MT são equivalentes em capacidades, resultando na tese de Church-Turing. Como há disputa sobre "quem foi o primeiro", esta tese também é conhecida como tese de Church e tese de Turing.



Problemas e Linguagens

- Um problema de decisão pode ser visto como um problema de reconhecimento de linguagem.
- Seja U o conjunto de todas as possíveis entradas para um problema de decisão (cujas respostas podem ser sim ou não).
- Seja $L \subseteq U$ o conjunto de todas as entradas que têm sim como resposta.
- Assim, L é a **linguagem correspondente ao problema de decisão**.
- Um problema é dito **decidível** se existir uma representação na qual o conjunto de cadeias que representam suas questões forma uma linguagem recursiva.
- Naturalmente, nem sempre conseguiremos representar um problema de decisão como uma linguagem recursiva: neste caso temos um problema **não-decidível**.
- **Para problemas não-decidíveis, não existe MT que páre para qualquer cadeia lida.**



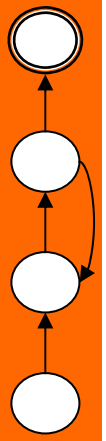
Suficiência das MTs: Um Exemplo

Argumento principal em favor da suficiência de MTs: nenhum modelo computacional desenvolvido mostrou-se mais poderoso do que a MT.

Exemplo típico: RAM

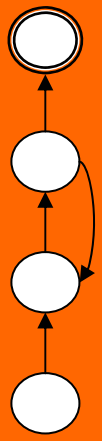
- Conjunto infinito de palavras de memória (endereços) numeradas, cada uma armazenando um número inteiro; e
- Conjunto finito de registradores aritméticos, capazes de armazenar qualquer número inteiro.
- Cada número inteiro pode codificar instruções, variáveis, etc.

É possível se projetar uma MT equivalente a uma RAM...



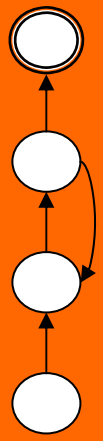
Computabilidade

- Preocupação: classificar problemas como **computáveis** ou não.
- Como a capacidade de uma MT para resolver um problema (embora não muito eficientemente) **excede** a dos computadores atuais, se encontrarmos problemas que as MTs não podem fazer, saberemos que os computadores também não o poderão.
- Um problema é **incomputável (ou insolúvel)** quando não existe um algoritmo que possa executá-lo.



Exemplo de Problema Insolúvel

- Suponha que seja dado um programa computacional e uma especificação precisa do que este programa deve fazer (ex.: ordenar uma lista de números).
  - Note: o programa e a especificação são objetos matemáticos precisos, neste exemplo.
- Pode-se automatizar o processo de verificar se o programa executa o especificado por meio de um outro programa de computador?
- Infelizmente, o problema geral de **verificação de software** é **insolúvel** por computador!

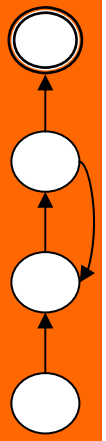


Codificando Máquinas de Turing

- Como codificar uma MT sobre um alfabeto (por exemplo, {0,1})?
  - M=(Q,{0,1},{0,1,B},δ,q_1,B,{q_2})
  - Faço: X_1=0, X_2=1, X_3=B, D_1=L, D_2=R. Assim, uma transição genérica $\delta(q_i,X_j)=(q_k,X_l,D_m)$ é representada por $0^i10^j10^k10^l10^m1$.
  - A máquina de Turing M é então representada como:

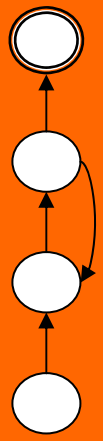
    $111c_1 11c_2 11 ... 11c_r 111$

    onde cada c_i é uma transição genérica da máquina.



Máquina × Cadeia

- Como a descrição ⟨T⟩ de uma MT é finita, podemos transformá-la em uma cadeia finita (eventualmente longa) descrevendo a MT.
- T(⟨T⟩) : significa que a MT T executa sua própria descrição ⟨T⟩ como cadeia de entrada.



Linguagem indecidível

- Seja a linguagem composta pelas cadeias $\langle T,\alpha \rangle$, correspondentes à concatenação das descrições $\langle T \rangle$ de MTs com cadeias α:

$$\mathbf{A_{MT} = \{ \langle T,\alpha \rangle \mid T \text{ é uma MT e } T \text{ aceita } \alpha \}}$$

- A linguagem A_{MT} é decidível? Este problema consiste em saber se existe uma MT H que **decide** A_{MT}.
- Provaremos, por contradição, que A_{MT} é indecidível, considerando como hipótese a existência de H.

Linguagem Indecidível

$H(\langle T,\alpha\rangle)$: { H aceita $\langle T,\alpha\rangle$ se T aceita α ; H rejeita $\langle T,\alpha\rangle$ se T não aceita α

- Considere a máquina D, definida em função de H:

D = "Para a entrada $\langle M\rangle$, sendo M uma MT:

1. Execute H com a entrada $\langle M, \langle M\rangle\rangle$.
2. Se H rejeitar, aceite.
3. Se H aceitar, rejeite."

Assim, $D(\langle M\rangle)$: { D aceita $\langle M\rangle$ se M não aceita $\langle M\rangle$; D rejeita $\langle M\rangle$ se M aceita $\langle M\rangle$

Linguagem Indecidível

Hipótese: $H(\langle T,\alpha\rangle)$: { H aceita $\langle T,\alpha\rangle$ se T aceita α ; H rejeita $\langle T,\alpha\rangle$ se T não aceita α } **(1)**

- Seja a MT D:

  $D(\langle M\rangle)$: { D aceita $\langle M\rangle$ se H rejeita $\langle M,\langle M\rangle\rangle$; D rejeita $\langle M\rangle$ se H aceita $\langle M,\langle M\rangle\rangle$ }

- Se $\langle M\rangle = \langle D\rangle$, vem:

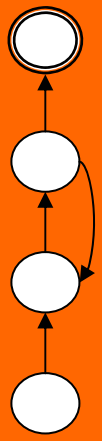
  $D(\langle D\rangle)$: { D aceita $\langle D\rangle$ se H rejeita $\langle D, \langle D\rangle\rangle$; D rejeita $\langle D\rangle$ se H aceita $\langle D, \langle D\rangle\rangle$ }

- Mas, segundo (1), o que significa $H(\langle D, \langle D\rangle\rangle)$?

  $D(\langle D\rangle)$: { D aceita $\langle D\rangle$ se D não aceita $\langle D\rangle$; D rejeita $\langle D\rangle$ se D aceita $\langle D\rangle$ }

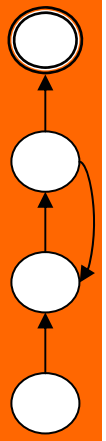
  Contradição!!!

- **Assim, D e H não podem existir e A_{MT} é indecidível**!



Problema da Parada

- Uma MT T que começa com uma cadeia α ou pára ou nunca pára.
- Problema de decisão: **"Existe um algoritmo que decide, dada qualquer MT e qualquer cadeia α, se a MT começando com α vai parar?"**
  - O interesse consiste num método geral que decida corretamente **todos** os casos.
- O problema da parada de uma MT é **insolúvel**.



Problema da Parada

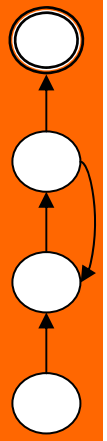
- Seja a linguagem composta pelas cadeias $\langle T,\alpha \rangle$, correspondentes à concatenação das descrições $\langle T \rangle$ de MTs com cadeias α:

$HALT_{MT}$ = { $\langle T,\alpha \rangle$ | T é uma MT e T pára com a entrada α}

- O Problema da Parada corresponde a saber se existe uma MT R que **decide** $HALT_{MT}$:

$R(\langle T,\alpha \rangle)$ {
- R aceita $\langle T,\alpha \rangle$ se T pára com α
- R rejeita $\langle T,\alpha \rangle$ se T não pára com α

Problema da Parada



- A MT R pode ser usada na construção da MT H que decide a linguagem A_{MT}.

H = "Para a entrada $\langle$**T,**$\alpha\rangle$**:**

1. **Execute R com a entrada $\langle$T,$\alpha\rangle$.**
2. **Se R rejeitar, <u>rejeite</u>.** *// T não pára com α*
3. **Se R aceitar, simule T com α até T parar** *// T pára com α*
4. **Se T aceitar, <u>aceite</u>. Se T rejeitar, <u>rejeite</u>."**

- Assim, se R decide $HALT_{MT}$, então H decide A_{MT}.
- Como vimos que H não pode existir e que A_{MT} é indecidível, então R também não existe e $HALT_{MT}$ é indecidível.