Tese de Church-Turing

Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática – CEEI

Departamento de Sistemas e Computação – DSC

Professor: Andrey Brito

Período: 2023.2

Onde estamos?

- Conhecemos as máquinas de Turing e vimos exemplos
  - Como usam a fita e resolvem problemas que os APs não resolvimos (ex., w#w)
  - Como variações não mudam seu poder de resolver problemas
    - Mais possibilidades para a fita (ex., infinita para os dois lados, mais operações para o cabeçote, multi-fita)
    - Não-determinismo
- Vimos alguns casos onde uma máquina resolvia uma parte de um problema maior e máquinas poderiam ser combinadas
  - Calculadoras
  - Estudo dirigido
- Devemos ter agora alguma confiança que elas podem resolver problemas complexos
  - Pense sobre como a transformação da não-determinística em determinística funcionava!

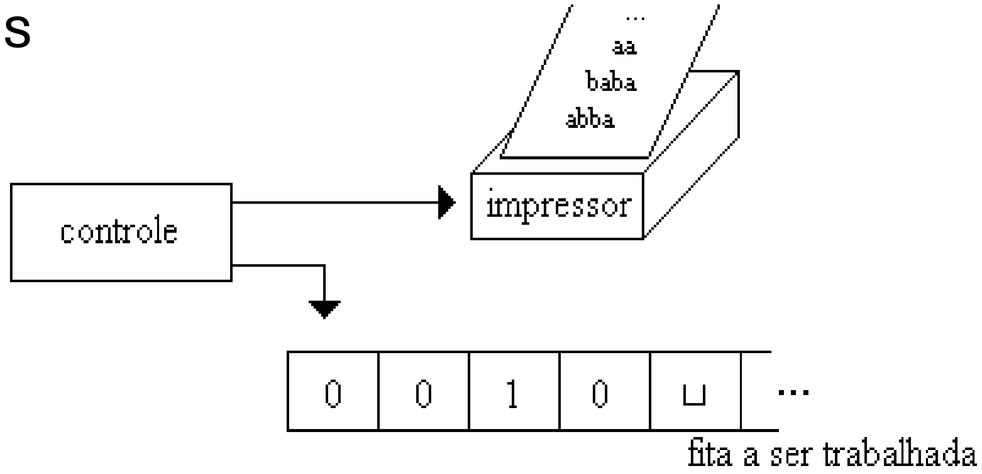
Mais alguns conceitos

Decidir versus reconhecer

- Uma MT aceita uma palavra se a partir da configuração inicial, ela chega no estado de aceitação
- De forma semelhante, uma MT rejeita uma palavra se chega no estado de rejeição
- Se para qualquer palavra da sua linguagem de entrada a máquina chega em uma configuração de aceitação ou rejeição, essa máquina **decide** essa linguagem
- Mas se uma máquina aceita as palavras de sua linguagem, mas pode "entrar em loop" e nunca chegar numa configuração de rejeição para as outras, ela somente **reconhece** a sua linguagem

Outra variação: Enumerador

- Ainda mais abstrata
- De certa forma “uma MT com uma impressora”
- A linguagem de E é o conjunto das palavras que ele imprime
  - Se não para, pode gerar uma lista infinita
  - A ordem é arbitrária e pode repetir palavras



Relação com MT

- Teorema: uma linguagem é Turing-reconhecível se e somente se um enumerador a enumera

Relação com MT

- Teorema: uma linguagem é Turing-reconhecível se e somente se um enumerador a enumera
  - Parte 1: Se existe um E para uma linguagem A, então existe uma MT M que reconhece A
  - Parte 2: Se existe uma MT M que reconhece L, existe um enumerador E gera L(M):

Relação com MT

- Teorema: uma linguagem é Turing-reconhecível se e somente se um enumerador a enumera

  - Parte 1: Se existe um E para uma linguagem A, então uma MT M a reconhece
    - Execute E. Cada vez que E imprime uma palavra compare-a com w.
    - Se w aparece como uma saída de E, aceitar.

  - Parte 2: Se existe uma MT M que reconhece L, E gera L(M):
    - Execute x passos de M para as palavras da linguagem, se M aceitou, imprima
    - Execute mais passos para cada palavra

Observação: enumerador e reconhecimento

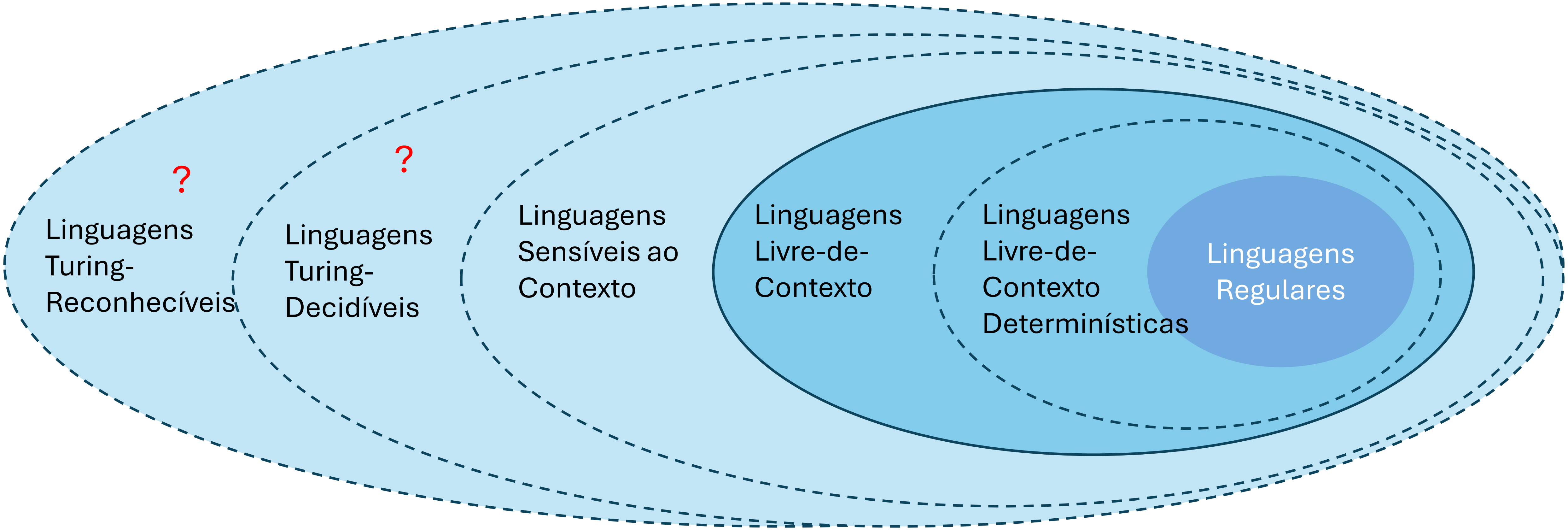
- Se você usar um enumerador para construir uma MT então essa máquina pode não parar se a palavra não pertence à linguagem

Linguagem Turing-Decidível

- Uma linguagem é Turing-Decidível se existe uma MT que
  - Reconhece, i.e., aceita, todas as palavras da linguagem
  - Rejeita todas as palavras que não estão na linguagem

- Outros nomes:
  - Linguagem Turing-decidível → Linguagem Recursiva
  - Linguagem Turing-reconhecível → Linguagem Recursivamente Enumerável

Hierarquia de linguagens



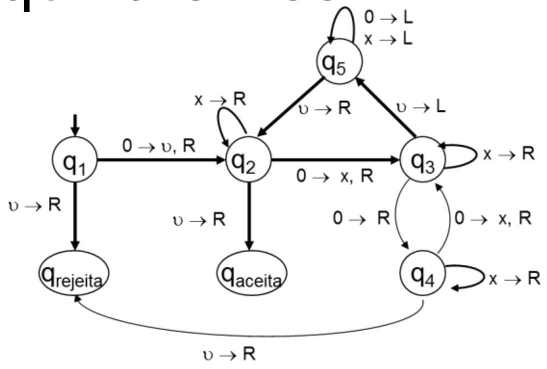
A tese de Church-Turing

A “Tese” de Church-Turing

- Na realidade uma “hipótese”, definição é intuitiva

$(x, y) \mapsto x*x + y*y$

- Modelos computacionais bem diferentes foram provados equivalentes
  - Funções recursivas
  - Cálculo Lambda
  - Máquinas de Turing
  - Assim como outros sistemas considerados “Turing completos”



- Gerou confiança que eles definiam todas as linguagens “computáveis”

https://pt.wikipedia.org/wiki/Tese_de_Church-Turing
https://pt.wikipedia.org/wiki/Função_computável

Algoritmo = MT = Computáveis

- Tese de Church-Turing: relaciona a noção intuitiva de algoritmos com a existência de algoritmos para uma MT

- Algoritmo: conjunto de passos, receitas, procedimentos, protocolos

https://pt.wikipedia.org/wiki/Tese_de_Church-Turing - Requisitos:

1. O algoritmo consiste de um conjunto finito de instruções simples e precisas, que são descritas com um número finito de símbolos.
2. O algoritmo sempre produz resultado em um número finito de passos.
3. O algoritmo pode, a princípio, ser executado por um ser humano com apenas papel e lápis.
4. A execução do algoritmo não requer inteligência do ser humano além do necessário para entender e executar as instruções.

Algoritmo

- Passos válidos (já em um nível alto de abstração)
  - Ache as raízes reais da equação do 2º grau definida por a,b,c (por quê?)
  - Teste se existe um valor inteiros de w,x,y,z entre 0 e 100 que tornam x.y.z + yz = w

- Passos inválido
  - Teste se existe um valor real de de w,x,y,z entre 0 e 100 que torna x.y.z + yz = w
  - Veja se existe algum ... tal que ...

Trabalhando com MTs e algoritmos

- Descrição de baixo nível: formal & diagrama de estados

- Descrição de nível médio: descrição da implementação (como no exemplo)
  - Movimento do cabeçote e símbolos
  - Gerência da fita e codificação da entrada (ex., 0^{2^n})

- Descrição de alto nível
  - Ignora cabeçote ou símbolos
  - Assume que a entrada pode ser codificada (ex., um número inteiro para determinar se é potência de 2)

M_1 = "Na cadeia de entrada w :

- Examine a entrada para ter certeza de que ela contém um símbolo # único. Se não, *rejeite*.
- Faça múltiplas passagens (ziguezague) pela fita:
  1. Marque a primeira posição não marcada no lado esquerdo do # guarde o símbolo original
  2. Avance até o lado direito do # e pule todos os símbolos marcados, se a palavra acabou rejeite.
  3. Compare o primeiro símbolo não marcado com o marcado no lado esquerdo, se forem diferentes, rejeite.
  4. Volte para o lado esquerdo do #
     a) Se todos os símbolos foram marcados, vá para o final da palavra e veja se todos também estão marcados, se não, rejeite. Se sim, aceite.
     b) Se ainda restam símbolos não marcados volte para o passo 1.

$$L = \{ w\#w \mid w \in \{0,1\}^* \}$$

Definição de "nível médio"

M_1 = “Na cadeia de entrada w :

- Faça múltiplas passagens (ziguezague) pela fita:
  - “Ziguezague” para os lugares correspondentes dos dois lados do # e determinar se eles “casam”
  - Em cada lado, use uma marcação para lembrar os símbolos já comparados
  - Se todos os símbolos respectivos casarem, ir para o estado de aceitação.

Definição de “alto nível”, o suficiente para entender que poderia ser feito.

$$L = \{ w\#w \mid w \in \{0,1\}^* \}$$

Decidibilidade

O décimo problema de Hilbert

- “Especifique um processo com um número finito de operações com o qual se possa determinar se um polinômio com múltiplas variáveis tem raízes inteiras” (David Hilbert, 1900)

- Exemplos:
  - $6.x^3.y.z + 3.x.y^2 - x^3 - 10$
  - Tem uma raiz em $x = 5$, $y = 3$ e $z = 0$

- Pede um algoritmo... Mas não existe um que garanta resultado!

Relembrando: Turing-reconhecível

- Uma ***linguagem*** é **Turing-reconhecível** se uma máquina de Turing a reconhece

- Rejeitar uma palavra não é o mesmo que “não aceitá-la”
  - Rejeitar: entrar no estado de rejeição
  - Entrar em loop: nunca entrar no estado de aceitação ou rejeição

Relembrando: Turing-decidível

- Uma ***máquina*** é **decidível** se ela nunca entra em loop (**decide** uma linguagem)

- Uma ***linguagem*** é **decidível** se alguma máquina de Turing a decide
  - Qual a relação entre o conjunto das linguagens decidíveis e o das linguagens reconhecíveis?

- Eu posso ter uma máquina que apenas reconhece, associada a uma linguagem decidível?

Decidíveis vs. Reconhecíveis

- L_1 = { p | p é um polinômio sobre uma variável, com raizes inteiras }
  - Exemplo: $4.x^3 - 2.x^2 + x – 7$

- L_2 = { p | p é um polinômio sobre múltiplas variáveis, com raízes inteiras}
  - Exemplo: $6.x^3.y.z + 3.x.y^2 - x^3 - 10$

- L_3 = { <G> | G é um grafo não-direcionado conexo}
  - Grafo conexo se todo nó pode ser atingido a partir de qualquer nó

É importante também entender bem o problema formulado na forma de conjunto.

L_1 é decidível

- L = { p | p é um polinômio sobre x com uma raiz inteira }
  - Exemplo: $4.x^3 - 2.x^2 + x - 7$

L_1 é decidível

- L = { p | p é um polinômio sobre x com uma raiz inteira }
  - Exemplo: $4.x^3 - 2.x^2 + x - 7$

- Qual a solução?
  - Se sua máquina testa todas as possibilidades, ela somente reconhece quando há uma solução e, mas pode nunca parar se não houver
  - Mas existe um teorema que diz que se a raiz existe ela está entre dois valores (logo, a procura tem fim)
    - K é o número de termos
    - $c_{máx}$ é coeficiente com maior valor absoluto
    - c_1 é o coeficiente do termo de mais alta ordem

$$\pm k \frac{c_{máx}}{c_1}$$

Um problema decidível (alto nível)

- L = { <G> | G é um gráfico não-direcionado conexo}
  - Grafo conexo se todo nó pode ser atingido a partir de qualquer nó

Um problema decidível (alto nível)

- L = { <G> | G é um gráfico não-direcionado conexo}
  - Grafo conexo se todo nó pode ser atingido a partir de qualquer nó

- Em alto nível: M = “ Sobre a entrada <G>, a codificação de um grafo G:
  - Selecione o primeiro nó de G e marque-o;
  - Repita até que nenhum nó adicional seja marcado: para cada nó de G, marque-o se ele está ligado a um nó marcado;
  - Se todos os nós de G estiverem marcados, aceite; senão, rejeite.”

Em um nível um pouco mais baixo

- Codificação: (1,2,3,4)((1,2);(2,3);(3,1);(1,4))
- O que a MT faria
  - Primeiro verificaria o formato: parênteses, repetições, consistência
  - Depois, marca o primeiro nó
  - Pega um não marcado, faz uma marcação diferente em um nó marcado e no não marcado
    - Se os dois estão ligados, ok
    - Senão, tente ver se o não marcado está ligado a outro marcado
  - Depois de passar por todos os nós, todos deveriam estar marcados: aceita ou rejeita