Universidade Federal de Alfenas

Linguagens Formais e Autômatos

Aula 16 – Decidibilidade

humberto@bcc.unifal-mg.edu.br

Últimas Aulas

- Uma Máquina de Turing (MT) possui:
  - uma fita infinita para representar a sua memória ilimitada;
  - Um cabeçote, para ler, e escrever na memória/fita;
  - Um controle;
  - Capacidade de movimentar-se para dois lados:
    - Direita
    - Esquerda

Últimas aulas

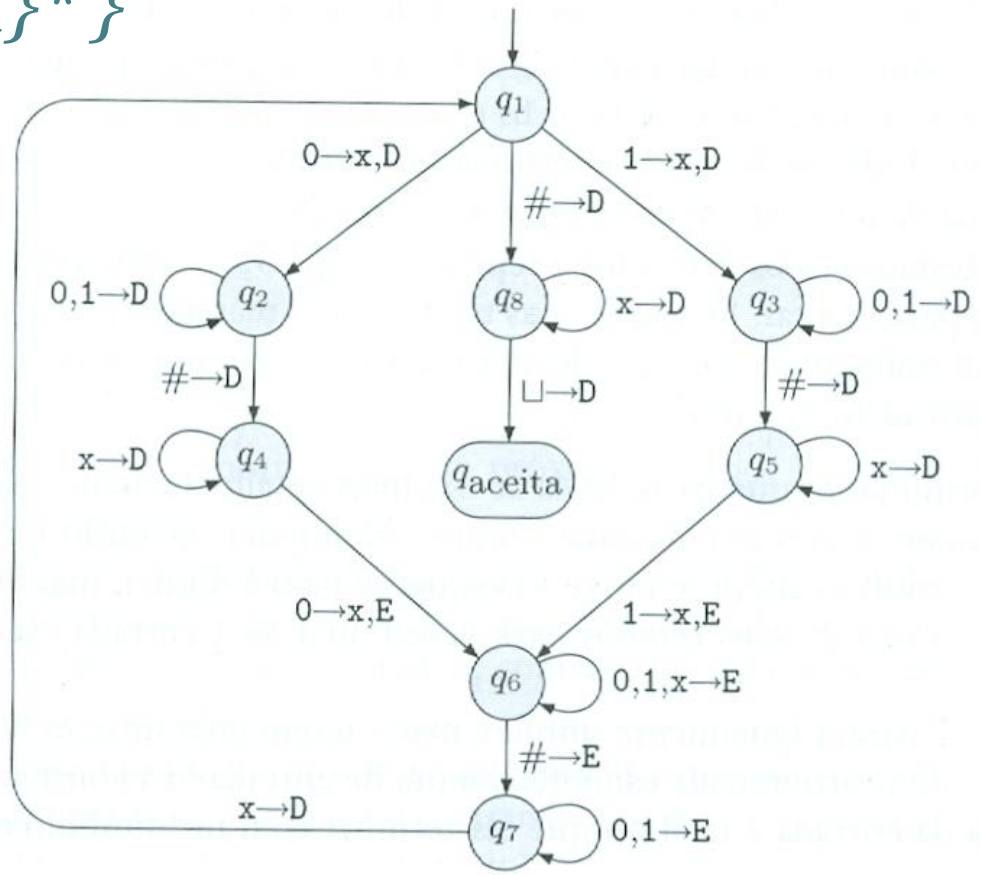
- Formalismo das MT

Uma ***máquina de Turing*** é uma 7-upla $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{aceita}}, q_{\text{rejeita}})$, onde Q, Σ, Γ são todos conjuntos finitos e

1. Q é o conjunto de estados.
2. Σ é o alfabeto de entrada que não contém o símbolo especial ***branco*** $\sqcup$,
3. Γ é o alfabeto da fita, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{E}, \text{D}\}$ é a função de transição,
5. q_0 é o estado inicial,
6. $q_{\text{aceita}} \in Q$ é o estado de aceitação,
7. $q_{\text{rejeita}} \in Q$ é o estado de rejeição.

Últimas aulas

- Exemplo de uma MT que reconhece uma linguagem que não é Livre do Contexto:
  - $L=\{ w\#w \mid w \in \{0,1\}^* \}$



Últimas aulas

- Variações da Máquina de Turing
  - Máquina de Turing com cabeçote parado;

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D, P\}$$

  - Máquina de Turing com Múltiplas Fitas

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{E, D\}^k$$

  - Máquina de Turing Não-Determinística

$$\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{E, D\})$$

Algoritmo

Algoritmo

- Um algoritmo é uma coleção de instruções simples para realizar alguma tarefa;
  - Freqüentemente chamamos de receitas ou procedimentos;

Algoritmo

- Um algoritmo é uma coleção de instruções simples para realizar alguma tarefa;
  - Freqüentemente chamamos de receitas ou procedimentos;

- A literatura antiga descreve alguns procedimentos para soluções de problemas matemáticos, como por exemplo, encontrar números primos, e máximos divisores comuns;
  - Os mesmo algoritmos que você viu no primeiro grau!

Os Problemas de Hilbert

Os Problemas de Hilbert

- No ano de 1900, o matemático David Hilbert palestrou no Congresso Internacional de Matemática em Paris;

Os Problemas de Hilbert

- No ano de 1900, o matemático David Hilbert palestrou no Congresso Internacional de Matemática em Paris;

- Na apresentação, ele expôs 23 problemas, na época sem solução, e coloco-os como um desafio para o novo século que começara;

Os Problemas de Hilbert

- No ano de 1900, o matemático David Hilbert palestrou no Congresso Internacional de Matemática em Paris;
- Na apresentação, ele expôs 23 problemas, na época sem solução, e coloco-os como um desafio para o novo século que começara;
- O décimo problema na famosa lista de David Hilbert diz respeito a algoritmos!

Os Problemas de Hilbert

- O décimo problema de Hilbert era conceber um procedimento que testasse se um polinômio tinha raiz inteira;

- Exemplo:

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

- Uma raiz de um polinômio é uma atribuição de valores a suas variáveis de modo que o valor do mesmo seja zero:

$$6x^3yz^2 + 3xy^2 - x^3 - 10 = 0$$

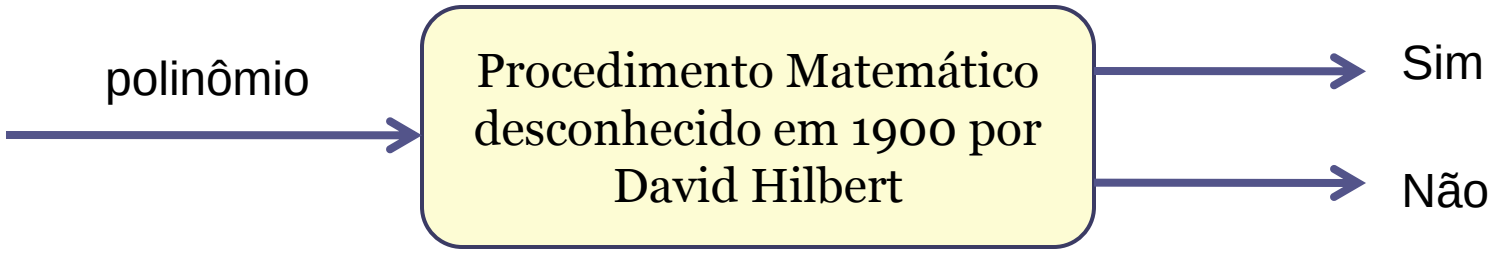
$$x = ?$$

$$y = ?$$

$$z = ?$$

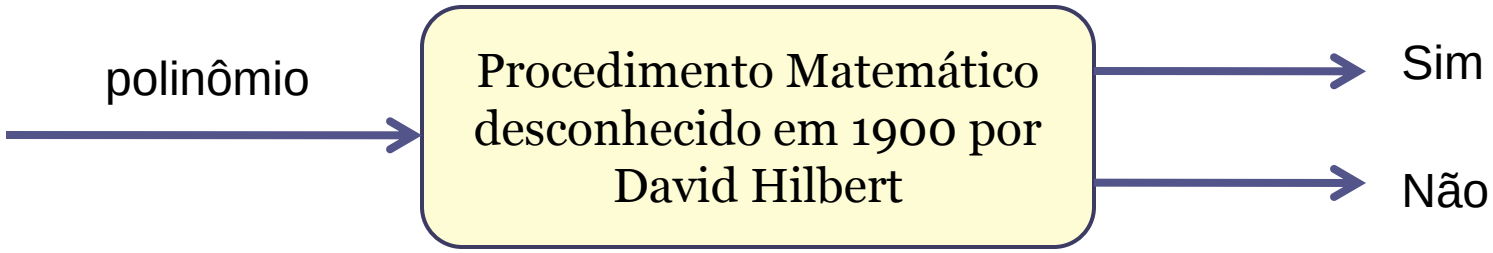
Os Problemas de Hilbert

- Alguns polinômios têm raiz inteira, e outros não têm



Os Problemas de Hilbert

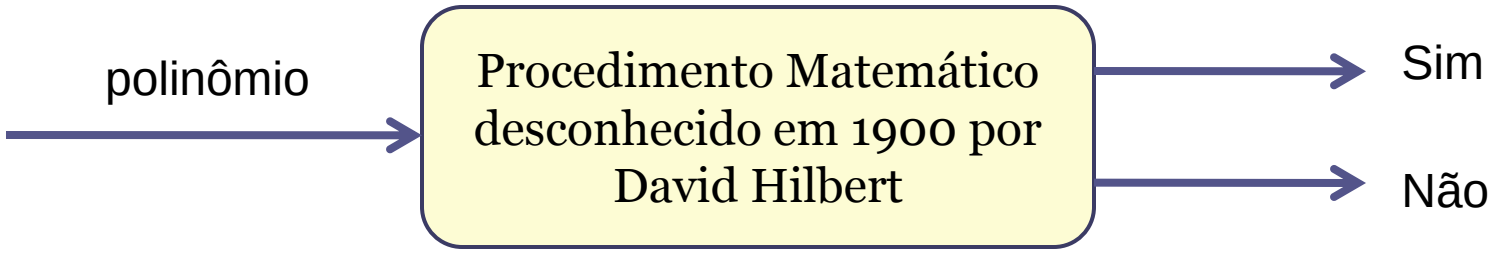
- Alguns polinômios têm raiz inteira, e outros não têm



- Hilbert não utilizou o termo algoritmo, mas sim um procedimento que fosse capaz de determinar, por um número finito de operações, se o polinômio possui raízes inteiras;

Os Problemas de Hilbert

- Alguns polinômios têm raiz inteira, e outros não têm



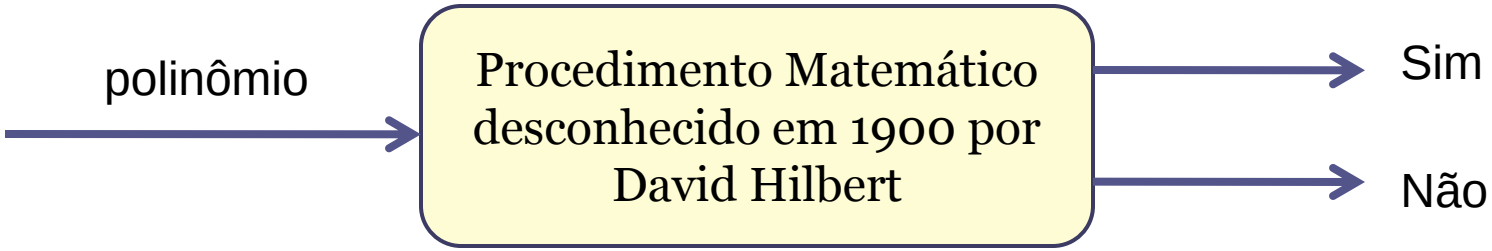
- Hilbert não utilizou o termo algoritmo, mas sim um procedimento que fosse capaz de determinar, por um número finito de operações, se o polinômio possui raízes inteiras;

- Ele aparentemente assumiu que tal procedimento tinha de existir;
  - Alguém apenas precisava encontrá-lo!

Os Problemas de Hilbert

- Nos dias de hoje sabemos que tal algoritmo não existe!

- Os matemáticos da época não conseguiram chegar a esta conclusão porque utilizavam um conceito intuitivo de algoritmo;



Os Problemas de Hilbert

- A definição formal veio em 1936 por
  - Alonzo Church e
  - Alan Turing

Os Problemas de Hilbert

- A definição formal veio em 1936 por
  - Alonzo Church e
  - Alan Turing
- Church definiu um sistema denominado λ-Cálculo;

Os Problemas de Hilbert

- A definição formal veio em 1936 por
  - Alonzo Church e
  - Alan Turing
- Church definiu um sistema denominado λ-Cálculo;
- Turing definiu suas máquinas;

Os Problemas de Hilbert

- A definição formal veio em 1936 por
  - Alonzo Church e
  - Alan Turing
- Church definiu um sistema denominado λ-Cálculo;
- Turing definiu suas máquinas;
- As definições, embora diferentes, possuem o mesmo poder computacional;

Os Problemas de Hilbert

- A definição formal veio em 1936 por
  - Alonzo Church e
  - Alan Turing
- Church definiu um sistema denominado λ-Cálculo;
- Turing definiu suas máquinas;
- As definições, embora diferentes, possuem o mesmo poder computacional;
- A conexão entre a noção informal de algoritmo e a definição precisa veio a ser chamada e a *Tese de Chuch-Turing*;

Os Problemas de Hilbert

- Baseando-se na Tese de Church-Turing, em 1970, Yuri Matijasevic mostrou que não existe algoritmo para testar se um polinômio tem raízes inteiras;

Os Problemas de Hilbert

- Baseando-se na Tese de Church-Turing, em 1970, Yuri Matijasevic mostrou que não existe algoritmo para testar se um polinômio tem raízes inteiras;

- Suponha que M1 é a máquina que indica se o polinômio p de primeiro grau possui as raízes inteiras:

  - Calcule o valor de p com x substituída sucessivamente por: 0, 1,-1,2,-2,3,-3,4,-4,...
  - Se o calculo do polinômio for igual a zero, aceite...

Os Problemas de Hilbert

- Se p possui raiz inteira, M1 em algum momento vai encontrá-la e aceitar;

Os Problemas de Hilbert

- Se p possui raiz inteira, M1 em algum momento vai encontrá-la e aceitar;

- Se p não tem raiz inteira, M1 vai rodar para sempre;

Os Problemas de Hilbert

- Se p possui raiz inteira, M1 em algum momento vai encontrá-la e aceitar;
- Se p não tem raiz inteira, M1 vai rodar para sempre;
- M1 é uma máquina reconhecedora!

Os Problemas de Hilbert

- Se p possui raiz inteira, M1 em algum momento vai encontrá-la e aceitar;
- Se p não tem raiz inteira, M1 vai rodar para sempre;
- M1 é uma máquina reconhecedora!
- M1 não é uma máquina decisora!

Decidibilidade

Decidibilidade

- Estudar decidibilidade é importante porque se você se depara com um problema indecidível, você precisa simplificá-lo, para resolvê-lo.

Decidibilidade

- Estudar decidibilidade é importante porque se você se depara com um problema indecidível, você precisa simplificá-lo, para resolvê-lo.

- Em algum momento do estudo de Ciência da Computação, os computadores parecem ser tão poderosos que você acredita que todos os problemas em algum momento cederão a eles;

Decidibilidade

- Estudar decidibilidade é importante porque se você se depara com um problema indecidível, você precisa simplificá-lo, para resolvê-lo.

- Em algum momento do estudo de Ciência da Computação, os computadores parecem ser tão poderosos que você acredita que todos os problemas em algum momento cederão a eles;

- Que tipos de problemas são insolúveis por computador?
- São problemas esotéricos, residindo somente nas mentes dos teóricos?
  - Não!

Decidibilidade

- Considere a seguinte Linguagem, que possui o conceito de virtualização:

$$A_{MT} = \{< M, w > | \; M \text{ é uma MT e } M \text{ aceita } w\}$$

Decidibilidade

- Considere a seguinte Linguagem, que possui o conceito de virtualização:

$$A_{MT} = \{< M, w > | \; M \text{ é uma MT e } M \text{ aceita } w\}$$

- A linguagem descreve todos os pares (algoritmo,entrada) que são decidíveis!

É possível descrever uma máquina/algoritmo para reconhecer tal linguagem?

Decidibilidade

$$A_{MT} = \{< M, w > | \; M \text{ é uma MT e } M \text{ aceita } w\}$$

- Considere a máquina de Turing U:
  - U = "Sobre a entrada $<M,w>$:
    - Simule M sobre a entrada w;

Decidibilidade

$$A_{MT} = \{< M, w > | \; M \text{ é uma MT e } M \text{ aceita } w\}$$

- Considere a máquina de Turing U:
  - U = "Sobre a entrada $<M,w>$:
    - Simule M sobre a entrada w;
    - Se M entra em seu estado de aceitação para w, então U também aceita a entrada $<M,w>$;

Decidibilidade

$$A_{MT} = \{< M, w > | \; M \text{ é uma MT e } M \text{ aceita } w\}$$

- Considere a máquina de Turing U:
  - U = "Sobre a entrada $<M,w>$:
    - Simule M sobre a entrada w;
    - Se M entra em seu estado de aceitação para w, então U também aceita a entrada $<M,w>$;
    - Se M entra em seu estado de rejeição para w, então U também rejeita a entrada $<M,w>$;"

Decidibilidade

- Note que a máquina U entra em *loop* sobre a entrada $\langle M,w \rangle$ se M entra em loop sobre w;

Decidibilidade

- Note que a máquina *U* entra em *loop* sobre a entrada <*M*,*w*> se *M* entra em loop sobre *w;*

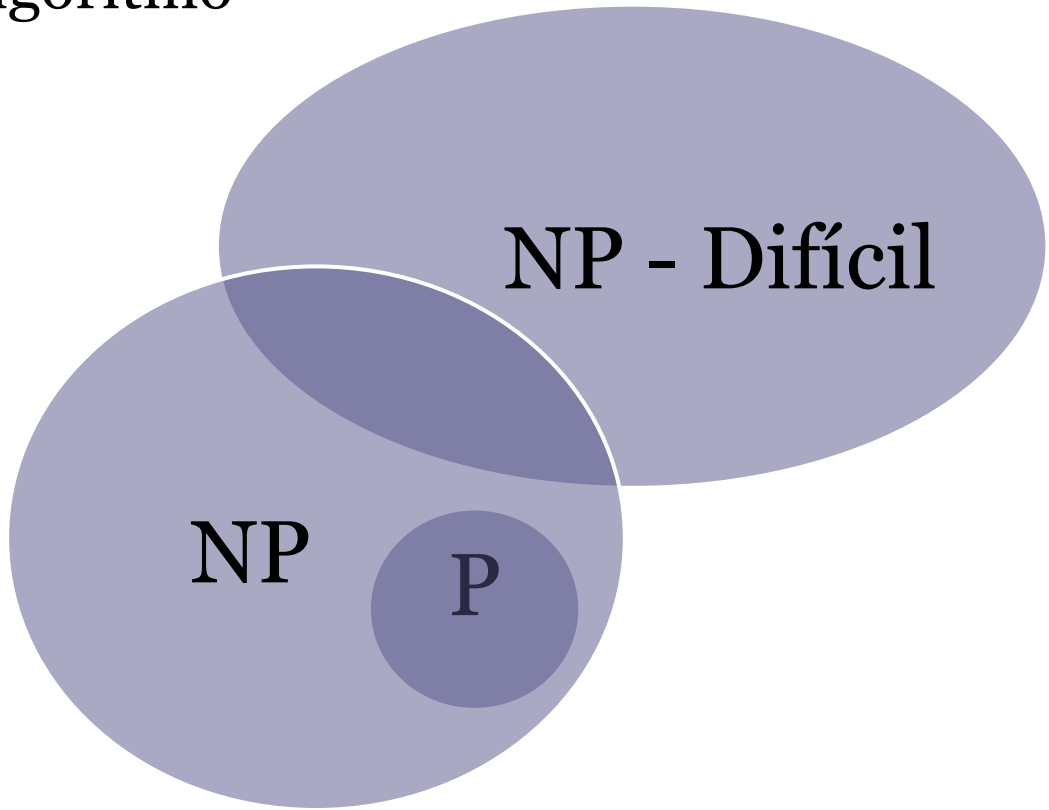
- Por este motivo dizemos que esta máquina não decide a linguagem A_{MT};

Decidibilidade

- Note que a máquina U entra em *loop* sobre a entrada $\langle M,w \rangle$ se M entra em loop sobre w;
- Por este motivo dizemos que esta máquina não decide a linguagem A_{MT};
- Embora, U reconheça A_{MT};

Retomada do assunto

- Voltaremos a discutir sobre problemas indecidíveis em outra disciplina!
  - Projeto e Análise de Algoritmo



Estudo…

- Estudar:
  - O método da diagonalização
  - O problema da parada é indecidível; (prova)
  - Um problema indecidível simples;

Bibliografia

- SIPSER, Michael. Introdução à Teoria da Computação. 2a ed.:São Paulo, Thomson, 2007.

- VIEIRA, Newton José. Introdução aos Fundamentos da Computação: Linguagens e Máquinas. 1a ed.: Rio de Janeiro: Thomson, 2006.