Tópicos em Complexidade Computacional

Cirucuitos e computação não uniforme

Professor Murilo V. G. da Silva

Departamento de Informática

Universidade Federal do Paraná

07/07/2022

Modelos de computação não-uniforme

Seria possível que SAT fosse tratável no sentido descrito abaixo?

Modelos de computação não-uniforme

Seria possível que SAT fosse tratável no sentido descrito abaixo?

- Embora não exista algoritmo polinomial para o problema

Modelos de computação não-uniforme

Seria possível que SAT fosse tratável no sentido descrito abaixo?

- Embora não exista algoritmo polinomial para o problema (ou seja, supondo o cenário em que $\mathrm{P} \neq \mathrm{NP}$)

Modelos de computação não-uniforme

Seria possível que SAT fosse tratável no sentido descrito abaixo?

- Embora não exista algoritmo polinomial para o problema
(ou seja, supondo o cenário em que $\mathrm{P} \neq \mathrm{NP}$)
- **Existe** um algoritmo polinomial diferente para cada instância diferente do SAT.

Modelos de computação não-uniforme

Seria possível que SAT fosse tratável no sentido descrito abaixo?

- Embora não exista algoritmo polinomial para o problema
(ou seja, supondo o cenário em que $\mathrm{P} \neq \mathrm{NP}$)
- **Existe** um algoritmo polinomial diferente para cada instância diferente do SAT.

Acredita-se que **não** seja o caso.

- Obs: note que não estamos nem mesmo considerando a factibilidade de tal modelo

Modelos de computação não-uniforme

Seria possível que SAT fosse tratável no sentido descrito abaixo?

- Embora não exista algoritmo polinomial para o problema
(ou seja, supondo o cenário em que $\mathrm{P} \neq \mathrm{NP}$)
- **Existe** um algoritmo polinomial diferente para cada instância diferente do SAT.

Acredita-se que **não** seja o caso.

- Obs: note que não estamos nem mesmo considerando a factibilidade de tal modelo

Em particular, tal modelo (que veremos agora) é dito **não uniforme**.

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito

  Exemplos: linguagem, conjunto infinito de instâncias, etc

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito

  Exemplos: linguagem, conjunto infinito de instâncias, etc

- **Solução de problema**: Modelado por objeto matemático finito

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelo não uniforme:

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelo não uniforme:

- **Problema**: Modelado por objeto matemático infinito

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelo não uniforme:

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelo não uniforme:

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático infinito

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelo não uniforme:

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático infinito
  Exemplos: Conjunto infinito de algoritmos, conjunto infinito de circuitos, etc

Modelos de computação não-uniforme

Modelos uniformes *vs* Modelos não uniformes:

Modelo uniforme (os modelos "padrão" que vimos até agora no curso):

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático finito
  Exemplos: Máquina de Turing, programa em C, autômato, etc

Modelo não uniforme:

- **Problema**: Modelado por objeto matemático infinito
  Exemplos: linguagem, conjunto infinito de instâncias, etc
- **Solução de problema**: Modelado por objeto matemático infinito
  Exemplos: Conjunto infinito de algoritmos, conjunto infinito de circuitos, etc
  - Obs: Ainda assim, trata-se de um objeto finito (e.g, um circuito) para cada uma das possíveis entradas diferente

A classe $\mathrm{P}_{/\mathrm{POLY}}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas:

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas: Um circuito C com *n entradas* e *uma saída* é um grafo direcionados acíclico com

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas: Um circuito C com n *entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas: Um circuito C com n *entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas: Um circuito C com n *entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$
- um sumidouro (saída do circuito)

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas: Um circuito C com n *entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$
- um sumidouro (saída do circuito)

A portas tem grau de entrada 2 e grau de saída 1

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com n entradas: Um circuito C com n *entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$
- um sumidouro (saída do circuito)

A portas tem grau de entrada 2 e grau de saída 1

Exemplo:

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com *n* entradas: Um circuito C com *n entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$
- um sumidouro (saída do circuito)

A portas tem grau de entrada 2 e grau de saída 1

Exemplo:

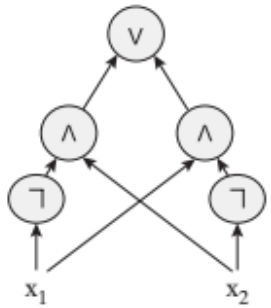

Figura: Barak & Arora, pág. 107.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com *n* entradas: Um circuito C com *n entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$
- um sumidouro (saída do circuito)

A portas tem grau de entrada 2 e grau de saída 1

Exemplo:

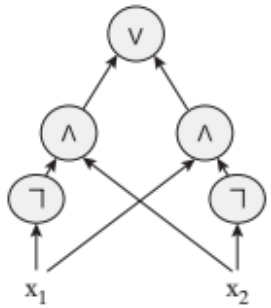

Figura: Barak & Arora, pág. 107.

- $|C|$: tamanho do circuito é o número de vértices do grafo

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Definição de circuito com *n* entradas: Um circuito C com *n entradas* e *uma saída* é um grafo direcionados acíclico com

- n fontes com rótulos $x_1, \ldots, x_n$
- demais vértices são *portas (lógicas)* $\vee$ $\wedge$, $\neg$
- um sumidouro (saída do circuito)

A portas tem grau de entrada 2 e grau de saída 1

Exemplo:

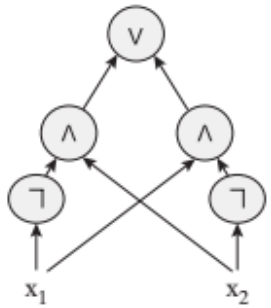

Figura: Barak & Arora, pág. 107.

- $|C|$: tamanho do circuito é o número de vértices do grafo
- $C(x)$ é a saída do circuito para entrada x

A classe $\mathrm{P}_{/\mathrm{POLY}}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos:

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

Exercício 11.2 Mostre que o problema de somar admite família de circuitos lineares

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

Exercício 11.2 Mostre que o problema de somar admite família de circuitos lineares

Exercício 11.3 Qualquer função booleana admite família de circutos de tamanho $\mathcal{O}(2^n/n)$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n\in\mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n\in\mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

Exercício 11.2 Mostre que o problema de somar admite família de circuitos lineares

Exercício 11.3 Qualquer função booleana admite família de circutos de tamanho $\mathcal{O}(2^n/n)$

Note (de 11.3):

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

Exercício 11.2 Mostre que o problema de somar admite família de circuitos lineares

Exercício 11.3 Qualquer função booleana admite família de circutos de tamanho $\mathcal{O}(2^n/n)$

Note (de 11.3): não é interessante pensar em $\mathrm{SIZE}(T(n))$ para funções exponenciais ou maiores

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

Exercício 11.2 Mostre que o problema de somar admite família de circuitos lineares

Exercício 11.3 Qualquer função booleana admite família de circutos de tamanho $\mathcal{O}(2^n/n)$

Note (de 11.3): não é interessante pensar em $\mathrm{SIZE}(T(n))$ para funções exponenciais ou maiores

A classe $\mathrm{P}_{/\mathrm{POLY}}$: Problemas decididos por famílias de circuitos de tamanho polinomial

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Família de circuitos: Seja $T : \mathbb{N} \to \mathbb{N}$ uma função. Uma *família de circuitos* de tamanho $T(n)$ é

- uma sequência (infinita) $\{C_n\}_{n \in \mathbb{N}}$ de circuitos com n entradas e 1 saída
- tal que $\forall n$, $|C_n| \leq T(n)$

Decisão por família de circuitos: Uma linguagem L está em $\mathrm{SIZE}(T(n))$ se

- $\exists$ família de circuitos $\{C_n\}_{n \in \mathbb{N}}$ de tamanho $T(n)$ tal que $x \in L \Leftrightarrow C_n(x) = 1$

Exercício 11.1 Seja $L = \{1^n \ : \ n \in \mathbb{N}\}$. Mostre que $L \in \mathrm{SIZE}(2n)$

Exercício 11.2 Mostre que o problema de somar admite família de circuitos lineares

Exercício 11.3 Qualquer função booleana admite família de circutos de tamanho $\mathcal{O}(2^n/n)$

Note (de 11.3): não é interessante pensar em $\mathrm{SIZE}(T(n))$ para funções exponenciais ou maiores

A classe $\mathrm{P}_{/\mathrm{POLY}}$: Problemas decididos por famílias de circuitos de tamanho polinomial

$$\mathrm{P}_{/\mathrm{POLY}} = \bigcup_{c \geq 1} \mathrm{SIZE}(n^c)$$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n : n$ é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n$: n é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

Sabemos que $L_{UH} \notin \mathrm{P}$, pois não existe algoritmo polinomial para L_{UH}

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n$: n é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

Sabemos que $L_{UH} \notin \mathrm{P}$, pois não existe algoritmo polinomial para L_{UH}
(na verdade não existe algoritmo algum, nem mesmo exponencial)

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n : n$ é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

Sabemos que $L_{UH} \notin \mathrm{P}$, pois não existe algoritmo polinomial para L_{UH}
(na verdade não existe algoritmo algum, nem mesmo exponencial)

Corolário 11.6 $\mathrm{P} \subsetneq \mathrm{P}_{/\mathrm{POLY}}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n : n$ é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

Sabemos que $L_{UH} \notin \mathrm{P}$, pois não existe algoritmo polinomial para L_{UH} (na verdade não existe algoritmo algum, nem mesmo exponencial)

Corolário 11.6 $\mathrm{P} \subsetneq \mathrm{P}_{/\mathrm{POLY}}$

Portanto, se mostrarmos que $\mathrm{NP} \not\subseteq P_{/\mathrm{POLY}}$, concluímos que $\mathrm{P} \neq \mathrm{NP}$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n : n$ é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

Sabemos que $L_{UH} \notin \mathrm{P}$, pois não existe algoritmo polinomial para L_{UH}
(na verdade não existe algoritmo algum, nem mesmo exponencial)

Corolário 11.6 $\mathrm{P} \subsetneq \mathrm{P}_{/\mathrm{POLY}}$

Portanto, se mostrarmos que $\mathrm{NP} \not\subseteq P_{/\mathrm{POLY}}$, concluímos que $\mathrm{P} \neq \mathrm{NP}$

- Seria mais fácil mostrar que não existe circuito polinomial para problema NP-completo?

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Teorema 11.4 $\mathrm{P} \subseteq \mathrm{P}_{/\mathrm{POLY}}$

Prova: Semelhante a demonstração do Teorema de Cook-Levin.

Teorema 11.5 Toda linguagem unária está em $\mathrm{P}_{/\mathrm{POLY}}$

Prova: Basta fazer um "and" dos bits de entrada.

Considere a versão unária do problema da parada:

- $L_{UH} = \{1^n : n$ é o número natural correspondente à string binária $\langle M, x \rangle$ tal que a MT M para com a entrada x $\}$

Sabemos que $L_{UH} \notin \mathrm{P}$, pois não existe algoritmo polinomial para L_{UH}
(na verdade não existe algoritmo algum, nem mesmo exponencial)

Corolário 11.6 $\mathrm{P} \subsetneq \mathrm{P}_{/\mathrm{POLY}}$

Portanto, se mostrarmos que $\mathrm{NP} \not\subseteq P_{/\mathrm{POLY}}$, concluímos que $\mathrm{P} \neq \mathrm{NP}$

- Seria mais fácil mostrar que não existe circuito polinomial para problema NP-completo?
- Uma das ideias é que o modelo de circuitos é mais "concreto" o podeira ser uma via para solucionar o problema P *vs* NP.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

De fato, sabemos que existem funções difíceis de serem computadas

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

De fato, sabemos que existem funções difíceis de serem computadas

Teorema 11.8 [Shannon (1949)] Existe função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

De fato, sabemos que existem funções difíceis de serem computadas

Teorema 11.8 [Shannon (1949)] Existe função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

Sabemos até mais:

Corolário 11.9 *Quase toda* função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

De fato, sabemos que existem funções difíceis de serem computadas

Teorema 11.8 [Shannon (1949)] Existe função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

Sabemos até mais:

Corolário 11.9 *Quase toda* função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

Note: Com isso, basta mostrarmos que apenas uma destas funções "difíceis" está em NP para concluirmos que $\mathrm{NP} \not\subseteq P_{/\mathrm{POLY}}$, concluímos que $\mathrm{P} \neq \mathrm{NP}$.

A classe $\mathrm{P}_{/\mathrm{POLY}}$

Como dissemos, um dos objetivos é mostrar que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Ainda estamos longe disso, mas sabemos o seguinte:

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

De fato, sabemos que existem funções difíceis de serem computadas

Teorema 11.8 [Shannon (1949)] Existe função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

Sabemos até mais:

Corolário 11.9 *Quase toda* função booleana que não pode ser computada por circuito de tamanho $2^n/10$ ou menor

Note: Com isso, basta mostrarmos que apenas uma destas funções "difíceis" está em NP para concluirmos que $\mathrm{NP} \not\subseteq P_{/\mathrm{POLY}}$, concluímos que $\mathrm{P} \neq \mathrm{NP}$.

- Porém, ainda não se encontrou tal função.

Prova do Teorema de Karp/Lipton

Antes da prova:

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$
  - O contrário não necessarimanete é verdade

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$
  - O contrário não necessarimanete é verdade (pois cada circuito pode ser diferente)

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$
  - O contrário não necessarimanete é verdade (pois cada circuito pode ser diferente)
- Podemos generalisar a definição de circuitos para que tenham m saídas

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$
  - O contrário não necessarimanete é verdade (pois cada circuito pode ser diferente)
- Podemos generalisar a definição de circuitos para que tenham m saídas
- Portanto **se existe** família de circuitos para decidir SAT

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$
  - O contrário não necessarimanete é verdade (pois cada circuito pode ser diferente)
- Podemos generalisar a definição de circuitos para que tenham m saídas
- Portanto **se existe** família de circuitos para decidir SAT
  - Existe família de circuitos para **encontrar** valoração para instância satisfazível

Prova do Teorema de Karp/Lipton

Antes da prova:

- Note que dado um algoritmo polinomial M:
  - Podemos construir uma família de circuitos polinomiais $C_1, C_2, ...$
  - O contrário não necessarimanete é verdade (pois cada circuito pode ser diferente)
- Podemos generalisar a definição de circuitos para que tenham m saídas
- Portanto **se existe** família de circuitos para decidir SAT
  - Existe família de circuitos para **encontrar** valoração para instância satisfazível

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova:

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$.

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$,

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$:

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$,

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT.

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT. (observações do slide anterior)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT. (observações do slide anterior)
- Ou seja, dado φ e u, existe uma família C'_n para computar v (se existir v)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT. (observações do slide anterior)
- Ou seja, dado φ e u, existe uma família C'_n para computar v (se existir v)

Se temos instância verdadeira de (*), então

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT. (observações do slide anterior)
- Ou seja, dado φ e u, existe uma família C'_n para computar v (se existir v)

Se temos instância verdadeira de (*), então

- $\exists w$ que é descrição de C' $\forall u$ $\varphi(u, C'_n(\varphi, u)) = 1$ (*)(*)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT. (observações do slide anterior)
- Ou seja, dado φ e u, existe uma família C'_n para computar v (se existir v)

Se temos instância verdadeira de (*), então

- $\exists w$ que é descrição de C' $\forall u$ $\varphi(u, C'_n(\varphi, u)) = 1$ (*)(*)

A volta vale também, ou seja (*) $\Leftrightarrow$ (*)(*)

Prova do Teorema de Karp/Lipton

Teorema 11.7 [Karp/Lipton (1980)] Se $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, então $\mathrm{PH} = \Sigma_2^p$

Prova: Suponha que $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$. Vamos mostrar que Π_2^p contém o problema $\Sigma_2\mathrm{SAT}$.

- Isso implica em $\Sigma_2^p \subseteq \Pi_2^p$, $\therefore$ $\Sigma_2^p = \Pi_2^p$ (pois $\Pi_2^p = \mathrm{co}\Sigma_2^p$) $\therefore$ $\mathrm{PH} = \Sigma_2^p$ (Teo. 10.6)

Considere uma instância de $\Sigma_2\mathrm{SAT}$: $\forall u \in \{0,1\}^n \; \exists v \in \{0,1\}^n \; \varphi(u,v) = 1$ (*)

- Como $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{POLY}}$, existe família de circuitos polinomiais C_n tal que
  - $C_n(\varphi, u) = 1 \Leftrightarrow \exists v$ tal que $\varphi(u,v)$ (aqui $\varphi(u,v)$ é φ com parte das variáveis "setadas" em u)
- Ou seja, existe família de circuitos para SAT.
- Portanto, existe família de circuitos para obter valoração para SAT. (observações do slide anterior)
- Ou seja, dado φ e u, existe uma família C'_n para computar v (se existir v)

Se temos instância verdadeira de (*), então

- $\exists w$ que é descrição de C' $\forall u$ $\varphi(u, C'_n(\varphi, u)) = 1$ (*)(*)

A volta vale também, ou seja (*) $\Leftrightarrow$ (*)(*)
Portanto $\Sigma_2\mathrm{SAT}$ pode ser expresso na forma Π_2^p.