Aula 15 - Circuit SAT

Luís Felipe

UFF

16 de Novembro de 2023

Luís Felipe
16/11/23

Considerações sobre $\mathcal{P}$ e $\mathcal{NP}$

Teorema. Seja X um problema $\mathcal{NP}$-completo. Então X pertence à classe $\mathcal{P}$ se, e somente se, $\mathcal{P} = \mathcal{NP}$.

Prova: $(\leftarrow)$ Se $\mathcal{P} = \mathcal{NP}$, então, como $X \in \mathcal{NP}$, temos que $X \in \mathcal{P}$.
$(\rightarrow)$ Se $X \in \mathcal{P}$ e considere um problema $Y \in \mathcal{NP}$. Como $Y \propto X$, $Y \in P$. $\square$

Exercício:

Corolário. Se existe um problema em $\mathcal{NP}$ que não pode ser resolvido em tempo polinomial, então nenhum problema $\mathcal{NP}$-completo pode ser resolvido em tempo polinomial

Luís Felipe
16/11/23

Teorema de Cook

Na aula anterior vimos que:

- $3SAT \propto IS \propto VC \propto SC$
- Teoricamente, ainda não mostramos que estes problemas são $\mathcal{NP}$-completos (precisamos de um problema $\mathcal{NP}$-completo como ponto de partida)
- $Y \propto ??? \propto 3SAT \propto IS \propto VC \propto SC, \forall Y \in \mathcal{NP}$

Cook, 1972

Luís Felipe
16/11/23

O primeiro problema $\mathcal{NP}$-completo

- Um circuito é um grafo direcionado especial
  - acíclico
  - fontes: rotuladas por 0 ou 1 (entrada fixa) ou o nome de uma variável de entrada
  - sumidouro: saída do circuito
  - os outros vértices: rotulados com um operador: $\wedge, \vee, \neg$

CIRCUIT SAT

Dados: Um circuito
Questão: Existe uma atribuição de valores às variáveis de entrada que forcem a saída do circuito ter valor 1?

Luís Felipe
16/11/23

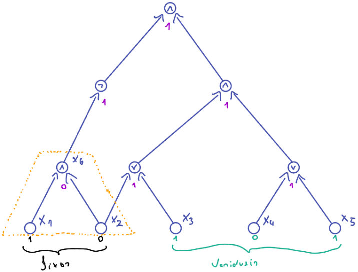
CIRCUIT SAT $\propto$ 3SAT

Teorema. CIRCUIT SAT $\propto$ 3 SAT

Prova: Considerando um circuito genérico, vamos construir uma instância particular de 3SAT (X, C) da seguinte forma:

- Variáveis x_v : vértices v do circuito
- Cláusulas:
  1. Para cada aresta direcionada, $(.)$ representa a aplicação do conectivo $\neg, \wedge, \neg$:
     - $u \to v$ $(\neg)$ corresponde à conjunção das cláusulas:
       $(x_v \vee x_u), (\overline{x_v} \vee \overline{x_u})$ assegura que $x_v = \overline{x_u}$
     - $u \to v$ $(\vee)$, $w \to v$ $(\vee)$ corresponde à conjunção das cláusulas:
       $(x_v \vee \overline{x_u}), (x_v \vee \overline{x_w}), (\overline{x_v} \vee x_u \vee x_w)$ garante que $x_v = x_u \vee x_w$
     - $u \to v$ $(\wedge)$, $w \to v$ $(\wedge)$ corresponde à conjunção das cláusulas:
       $(\overline{x_v} \vee x_u), (\overline{x_v} \vee x_w), (x_v \vee \overline{x_u} \vee \overline{x_w})$ garante que $x_v = x_u \wedge x_w$
     - Para cada vértice fonte v com valor constante (1 ou 0): uma cláusula unitária (x_v) (se v : 1) ou $(\overline{x_v})$ (se v : 0)
     - Para cada saída v, adicione uma cláusula unitária (x_v)

Luís Felipe
16/11/23



$$(x_1) \wedge (\bar{x}_2) \wedge (\bar{x}_6 \vee x_1) \wedge (\bar{x}_6 \vee x_2) \wedge (x_6 \vee \bar{x}_1 \vee \bar{x}_2)$$

Truth Table Generator

This tool generates truth tables for propositional logic formulas. You can enter logical operators in several different formats. For example, the propositional formula p ∧ q → ¬r could be written as p /\ q -> ~r, as p and q => not r, or as p && q -> !r. The connectives ⊤ and ⊥ can be entered as T and F.

```
2)/\(x6 \/ ~x1 \/ ~x2
```

x1	x2	x6	(x1 ∧ (¬x2 ∧ ((¬x6 ∨ x1) ∧ ((¬x6 ∨ x2) ∧ (x6 ∨ (¬x1 ∨ ¬x2))))))
F	F	F	**F**
F	F	T	**F**
F	T	F	**F**
F	T	T	**F**
T	F	F	**T**
T	F	T	**F**
T	T	F	**F**
T	T	T	**F**

Luís Felip
16/

$(x_v \vee x_u) \wedge (\bar{x}_v \vee \bar{x}_u)$, Garante:
$x_v = \overline{x_u}$

Truth Table Generator

This tool generates truth tables for propositional logic formulas. You can enter logical operators in several different formats. For example, the propositional formula p ∧ q → ¬r could be written as p /\ q -> ~r, as p and q => not r, or as p && q -> !r. The connectives ⊤ and ⊥ can be entered as T and F.

\/ xu) /\ (~xv \/ ~xu

xu	xv	((xv ∨ xu) ∧ (¬xv ∨ ¬xu))
F	F	**F**
F	T	**T**
T	F	**T**
T	T	**F**

$(x_v \vee \bar{x}_u) \wedge (x_v \vee \bar{x}_w) \wedge (\bar{x}_v \vee x_u \vee x_w)$, Garante:
$x_v = x_u \vee x_w$

Truth Table Generator

This tool generates truth tables for propositional logic formulas. You can enter logical operators in several different formats. For example, the propositional formula p ∧ q → ¬r could be written as p /\ q -> ~r, as p and q => not r, or as p && q -> !r. The connectives ⊤ and ⊥ can be entered as T and F.

(xv \/ ~xu) /\ (xv \/

xu	xv	xw	((xv ∨ ¬xu) ∧ ((xv ∨ ¬xw) ∧ (¬xv ∨ (xu ∨ xw))))
F	F	F	**T**
F	F	T	**F**
F	T	F	**F**
F	T	T	**T**
T	F	F	**F**
T	F	T	**F**
T	T	F	**T**
T	T	T	**T**

$(\bar{x}_v \vee x_u) \wedge (\bar{x}_v \vee x_w) \wedge (x_v \vee \bar{x}_u \vee \overline{x_w})$, Garante:
$x_v = x_u \wedge x_w$

Truth Table Generator

This tool generates truth tables for propositional logic formulas. You can enter logical operators in several different formats. For example, the propositional formula p ∧ q → ¬r could be written as p /\ q -> ~r, as p and q => not r, or as p && q -> !r. The connectives ⊤ and ⊥ can be entered as T and F.

(~xv \/ xu) /\ (~xv \

xu	xv	xw	((¬xv ∨ xu) ∧ ((¬xv ∨ xw) ∧ (xv ∨ (¬xu ∨ ¬xw))))
F	F	F	**T**
F	F	T	**T**
F	T	F	**F**
F	T	T	**F**
T	F	F	**T**
T	F	T	**F**
T	T	F	**F**
T	T	T	**T**

Luís Felipe
16/11/23

CIRCUIT SAT $\propto$ 3SAT

Prova (continuação): Para construirmos uma entrada pro 3SAT, precisamos transformar as cláusulas que têm tamanho 1 ou 2 em cláusulas de tamanho 3.
Para isso, vamos adicionar 4 variáveis auxiliares:

- variáveis: z_1, z_2, z_3, z_4
- cláusulas:
  $(\overline{z_i} \vee z_3 \vee z_4), (\overline{z_i} \vee \overline{z_3} \vee z_4), (\overline{z_i} \vee z_3 \vee \overline{z_4})(\overline{z_i} \vee \overline{z_3} \vee \overline{z_4}), i = 1, 2$
  - Neste caso, estamos forçando que $z_1 : F, z_2 : F$
- Caso haja uma cláusula com exatamente um literal:
  $(t) \mapsto (t \vee z_1 \vee z_2)$
- Caso haja uma cláusula com exatamente dois literais:
  $(t \vee t') \mapsto (t \vee t' \vee z_1)$

Luís Felipe
16/11/23

CIRCUIT SAT $\propto$ 3SAT

Prova (continuação): Resta mostrar que:

Existe uma atribuição para as variáveis do circuito tal que a saída seja $1 \leftrightarrow (X, C)$ é satisfazível.

($\rightarrow$) Por construção, observe que sempre temos um literal verdadeiro em cada uma das cláusulas de (X, C).

($\leftarrow$) Como a saída é uma conjunção de uma ou mais cláusulas por construção e (X, C) é satisfeita, então existe uma atribuição para as variáveis do circuito que conduzem à saída 1. $\square$

Luís Felipe
16/11/23

CIRCUIT SAT é $\mathcal{NP}$-completo

Note que para mostrar que todos os problemas mencionados até o momento são $\mathcal{NP}$-completos, precisaríamos mostrar que:

$$\boxed{\text{Teorema: CIRCUIT SAT é } \mathcal{NP}\text{-completo.}}$$

Prova: Precisamos mostrar que CIRCUIT SAT $\in \mathcal{NP}$ e que para todo problema $Y \in \mathcal{NP}$, temos que $Y \propto$ CIRCUIT SAT.

CIRCUIT SAT $\in \mathcal{NP}$: Tendo um circuito de altura polinomial no número de variáveis, o percurso por nível certifica uma solução em tempo polinomial, pois só devemos avaliar resultados das sucessivas portas lógicas $\wedge, \vee, \neg$ e ao final teremos o resultado do circuito no nó sumidouro (nó saída do circuito).

Luís Felipe
16/11/23

$Y \propto$ CIRCUIT SAT

Qualquer algoritmo polinomial pode ser processado como um circuito de tamanho polinomial, cujas portas lógicas codificam as entradas do algoritmo. (Afinal, qualquer algoritmo pode ser executado em um computador, e o computador é, em última análise, um circuito booleano implementado em um chip.)

Assim, se um algoritmo resolve um problema de decisão Y, então essa resposta é dada no sumidouro do circuito.

Portanto, dada uma instância I do problema Y cuja solução seja S, construímos em tempo polinomial um circuito cujas entradas são:

Entradas fixas do circuito são os bits de I,
Entradas variáveis do circuito são os bits da solução S.

Portanto, a saída do circuito é 1 (verdadeiro) se, e somente se, os bits da entrada da solução S de fato codificam uma solução da instância I de Y. $\square$