SAT, 3-SAT pertencem a NP-Completo

Prof. Daniel Kikuti

Universidade Estadual de Maringá

SAT

O problema

Consiste em determinar se uma fórmula booleana ϕ é satisfazível, isto é, se existe uma atribuição de valores verdade para as variáveis tal que ϕ seja verdadeira.

Instância

- n variáveis booleanas: $x_1, x_2, \ldots, x_n$;
- m conectivos lógicos tais como: $\wedge$ (e), $\vee$ (ou), $\neg$ (negação), $\rightarrow$ (condicional), $\leftrightarrow$ (bicondicional);
- Parênteses (sem redundância).

Linguagem

$$\text{SAT} = \{\langle\phi\rangle : \phi \text{ é uma fórmula booleana satisfazível}\}.$$

Exemplo

A fórmula ϕ a seguir é satisfazível?

$$\phi = \Big((x_1 \to x_2) \vee \neg\big((\neg x_1 \leftrightarrow x_3) \vee x_4\big)\Big) \wedge \neg x_2$$

- Como funcionaria um algoritmo força bruta para este problema?
- Qual a complexidade deste algoritmo?
- Mostraremos que este é um problema **NP-Completo**.

Exemplo

A fórmula ϕ a seguir é satisfazível?

$$\phi = \Big((x_1 \to x_2) \vee \neg\big((\neg x_1 \leftrightarrow x_3) \vee x_4\big)\Big) \wedge \neg x_2$$

Sim. A atribuição $x_1 = F$, $x_2 = F$, $x_3 = T$ e $x_4 = T$ satisfaz ϕ.

- Como funcionaria um algoritmo força bruta para este problema? Tabela verdade e substituição na fórmula.
- Qual a complexidade deste algoritmo? Custo $O(2^n \times eval(\phi))$.
- Mostraremos que este é um problema $\mathbf{NP}$-**Completo**.

SAT ∈ NP-Completo

1 - Mostrar que SAT ∈ NP

- Mostrar um algoritmo polinomial não determinístico que resolve SAT.
- Mostrar um algoritmo verificador para o problema SAT que executa em tempo polinomial.

2 - Escolher um problema NP-Completo

- Usaremos o problema CIRCUIT-SAT: "Dado um circuito composto por portas $\wedge$, $\vee$, $\neg$, existe uma atribuição de valores verdade na entrada tal que sua saída seja 1?"
- O tamanho de um circuito é o número de portas lógicas mais o número de trilhas.
- CIRCUIT-SAT $= \{\langle C \rangle : C$ é um circuito satisfazível$\}$.

SAT $\in$ NP-Completo

3 - Mostrar que CIRCUIT-SAT $\preceq_p$ SAT

Como reduzir qualquer instância de CIRCUIT-SAT para uma instância de SAT em tempo polinomial?

SAT $\in$ NP-Completo

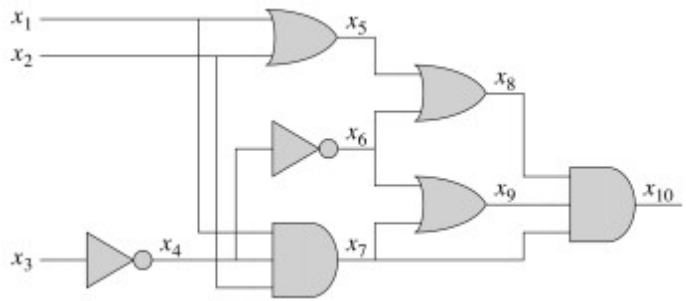
3 - Mostrar que CIRCUIT-SAT $\preceq_p$ SAT

Como reduzir qualquer instância de CIRCUIT-SAT para uma instância de SAT em tempo polinomial?

- **Primeira tentativa:** usar indução para expressar qualquer circuito combinatório booleano como uma fórmula booleana. Verifique a porta que produz a saída para o circuito e expresse indutivamente cada uma das portas de entrada como fórmulas em função de suas entradas.
- O Exercício 34.4-1 do Cormen pede para você mostrar que, para algumas instâncias de C, esta redução é exponencial.

SAT $\in$ NP-Completo

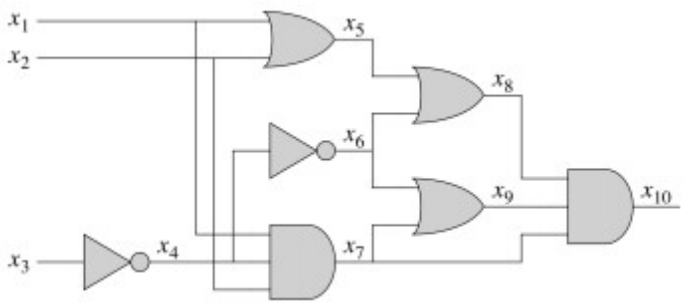
Exemplo



Uma redução polinomial

- Para cada trilha x_i em C, associar uma variável em ϕ.
- Representar como cada porta opera por meio de uma pequena fórmula envolvendo as variáveis de entrada e saída.
- Por exemplo: a operação da porta AND de saída será $x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9)$.

SAT $\in$ NP-Completo



$$\begin{aligned}\phi = x_{10} \wedge & \\ & (x_4 \leftrightarrow \neg x_3) \wedge \\ & (x_5 \leftrightarrow (x_1 \vee x_2)) \wedge \\ & (x_6 \leftrightarrow \neg x_4) \wedge \\ & (x_7 \leftrightarrow (x_1 \wedge x_2 \wedge x_4)) \wedge \\ & (x_8 \leftrightarrow (x_5 \vee x_6)) \wedge \\ & (x_9 \leftrightarrow (x_6 \vee x_7)) \wedge \\ & (x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9)).\end{aligned}$$

SAT $\in$ NP-Completo

Proposição: C é satisfazível se e somente se ϕ é satisfazível.

- $\Rightarrow$ C possui uma atribuição tal que a saída é 1 (cada trilha também possui um valor definido). Assim, quando atribuímos os valores das trilhas para as variáveis em ϕ, cada subfórmula representando as portas lógicas que inserimos em ϕ torna-se verdadeira e, a conjunção de todas produz verdadeiro.
- $\Leftarrow$ Se uma atribuição satisfaz ϕ, então o circuito também é satisfazível por um argumento análogo.

3-SAT

- Um **literal** é uma variável booleana ou sua negação.
- Uma **cláusula** é uma fórmula composta por vários literais conectados por '$\vee$'s. Exemplo: $(x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4)$.
- Uma fórmula está na **forma normal conjuntiva** (FNC) se compreende várias cláusulas conectadas por '$\wedge$'s. Exemplo: $(x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_3 \vee \neg x_5 \vee x_6) \wedge (x_3 \vee x_6)$.
- Uma fórmula é **3FNC** se todas as cláusulas possuem três literais. Exemplo: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_3 \vee x_5 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$.

O problema

3-SAT é o problema de determinar se uma fórmula ϕ em 3FNC é satisfazível.

3-SAT $\in$ NP-Completo

1 - Mostrar que 3-SAT $\in$ NP

Mesma demonstração usada para provar que SAT $\in$ **NP**.

2 - Escolher um problema NP-Completo

Usaremos SAT.

3 - SAT $\preceq_p$ 3-SAT

Precisamos mostrar que é possível reduzir instâncias de SAT para instâncias de 3-SAT em tempo polinomial.

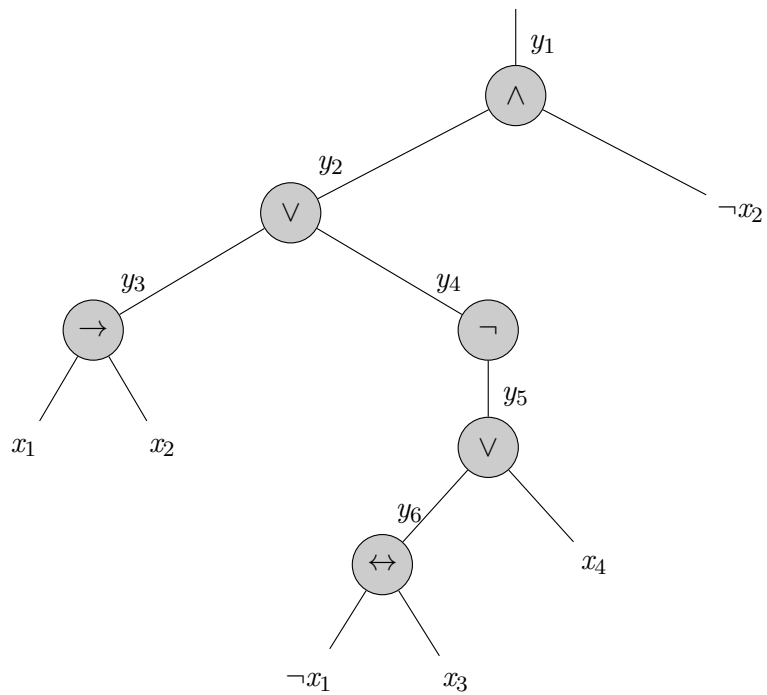
SAT $\preceq_p$ 3-SAT

Descrevendo ϕ por meio de árvore binária

Considere a seguinte fórmula:

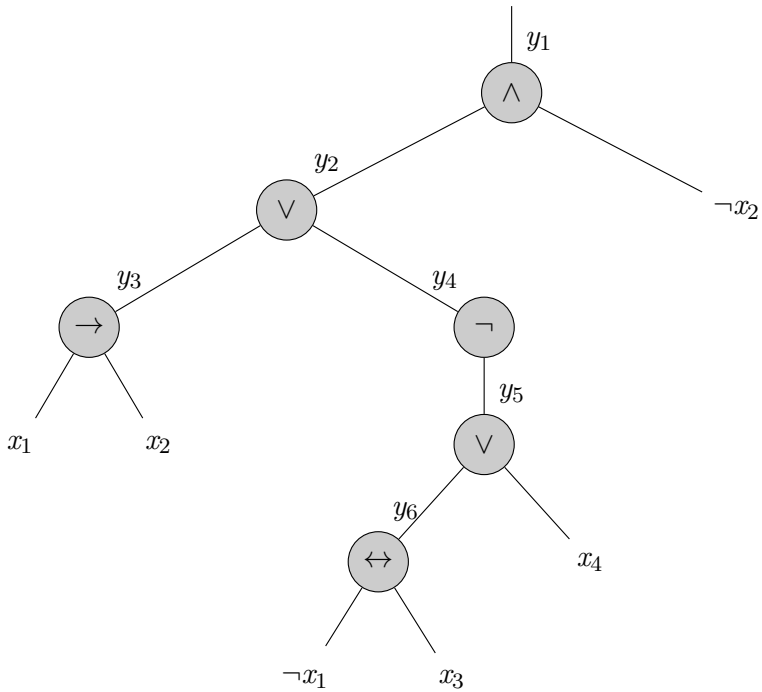
$$\phi = \Big((x_1 \to x_2) \vee \neg\big((\neg x_1 \leftrightarrow x_3) \vee x_4\big)\Big) \wedge \neg x_2.$$

A árvore a seguir representa esta fórmula.



SAT $\preceq_p$ 3-SAT

Árvore representando ϕ



Mesma ideia da redução de CIRCUIT-SAT

$$\begin{aligned}\phi' = y_1 &\wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ &\wedge (y_2 \leftrightarrow (y_3 \vee y_4)) \\ &\wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ &\wedge (y_4 \leftrightarrow \neg y_5) \\ &\wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \\ &\wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3)).\end{aligned}$$

Observe que ϕ' é uma conjunção de subfórmulas, mas ainda não está na 3FNC.

SAT $\preceq_p$ 3-SAT

Obtendo FNC para cada subfórmula booleana

1. Para cada subfórmula ϕ'_i, faça a tabela verdade.
2. Em cada linha onde a fórmula tem valor verdade 1, insira uma cláusula disjuntiva equivalente.

Exemplo para a subfórmula $\phi'_1 = (y_1 \leftrightarrow (y_2 \wedge \neg x_2))$

y_1	y_2	x_2	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$\phi''_1 = (\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee \neg y_2 \vee x_2) \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee y_2 \vee \neg x_2).$$

SAT $\preceq_p$ 3-SAT

Cada cláusula de ϕ'' terá no máximo 3 variáveis.

ϕ'' está na FNC, mas não em 3FNC

- Se C_i de ϕ'' possui três literais, então simplesmente inclua C_i de ϕ'' em ϕ'''.
- Se C_i de ϕ'' possui dois literais distintos ($C_i = (l_1 \vee l_2)$), então inclua $(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$ como cláusulas de ϕ'''. Independentemente do valor de p, uma das cláusulas terá valor 1 (valor identidade para o $\wedge$) e a outra terá valor $l_1 \vee l_2$.
- Se C_i de ϕ'' possui apena um literal l, então inclua $(l \vee p \vee q) \wedge (l \vee p \vee \neg q) \wedge (l \vee \neg p \vee q) \wedge (l \vee \neg p \vee \neg q)$ como cláusulas de ϕ'''. Independentemente dos valores de p e q, uma das quatro cláusulas é equivalente a l, e as outras três terão valor 1.

SAT $\preceq_p$ 3-SAT

Equivalência lógica

Como as transformações efetuadas nos passos mantém a equivalência lógica das fórmulas, concluímos que ϕ''' é satisfazível se e somente se ϕ é satisfazível.

Redução em tempo polinomial

Nos três passos, cada transformação foi feita em tempo polinomial.

- A construção de ϕ' a partir de ϕ introduz no máximo uma variável e uma cláusula por conectivo em ϕ.
- A construção de ϕ'' a partir de ϕ' pode introduzir no máximo oito cláusulas em ϕ' por cada cláusula em ϕ', pois cada cláusula em ϕ' possui no máximo três variáveis, e a tabela verdade para cada cláusula possui no máximo $2^3 = 8$ linhas.
- A construção de ϕ''' a partir de ϕ'' introduz no máximo quatro cláusulas em ϕ''' por cláusula em ϕ''.

Portanto, o tamanho resultante da fórmula ϕ''' é polinomial no tamanho da fórmula original.

Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. **Introduction to Algorithms**, Third Edition. The MIT Press. Chapter 34.
- Kleinberg J., and Tardos E. **Algorithm Design**. 2005. Pearson. Chapter 8.