

11 de outubro de 2016

Fórmulas FNC e FND

Uma fórmula FNC F sobre as variáveis x_1, \dots, x_n é a conjunção de disjunções de literais (uma das variáveis ou a sua negação).

$$F = C_1 \wedge \dots \wedge C_s$$

Onde cada cláusula C_i é a disjunção de literais e s é o tamanho de F .

Se cada cláusula tem no máximo k literais então dizemos que F tem largura k e dizemos que F é uma k -FNC.

Fórmulas FNC e FND

Uma fórmula FND F por outro lado é a disjunção de conjunções de literais.

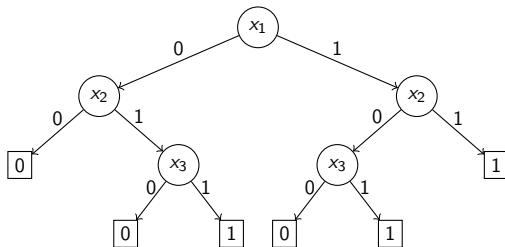
$$F = T_1 \vee \cdots \vee T_s$$

Onde cada termo T_i é a conjunção de literais e s é o tamanho de F .

De novo, se cada termo tem no máximo k literais então F tem largura k e F é uma k -FND.

Árvores de decisão

Uma árvore de decisão é algo como a imagem abaixo:



- ▶ Cada nodo leva o label de uma das variáveis.
- ▶ Começando do nodo mais alto, o algoritmo ramifica para a direita ou à esquerda dependendo do valor da variável lida.
- ▶ As folhas guardam o valor da função em cada entrada que chega nela.

Árvores de decisão

Denotamos a saída de uma árvore de decisão T sobre a entrada x por $T(x)$. Se f é tal que $f(x) = T(x)$ para todos os x então dizemos que T computa a função f .

- ▶ Por exemplo, a árvore de decisão do slide anterior computa a função que Majority_3 , que é 1 se e somente se o número de 1s na entrada é pelo menos 2.

O tamanho de T é o número de folhas e sua profundidade é o maior caminho do nó mais alto até uma das folhas.

- ▶ A árvore do slide anterior tem tamanho 6 e profundidade 3.

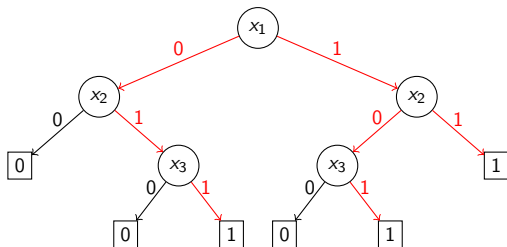
Árvores de decisão

É importante notar que árvores de decisão são mais fracas do que fórmulas FNC (FND).

- ▶ Se T é uma árvore de decisão de tamanho s e profundidade d então existe uma fórmula FND (FNC) F tal que $F(x) = T(x)$, para todos x , de tamanho $\leq s$ e largura $\leq d$.
 - ▶ FND: Cada caminho P da árvore tal que $T(P) = 1$ define uma cláusula.
 - ▶ FNC: Cada caminho P da árvore tal que $T(P) = 0$ define um termo.

Árvores de decisão

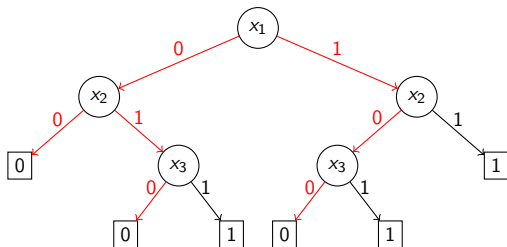
É importante notar que árvores de decisão são mais fracas do que fórmulas FNC (FND).



$$F = (\bar{x}_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_1 \wedge x_2)$$

Árvores de decisão

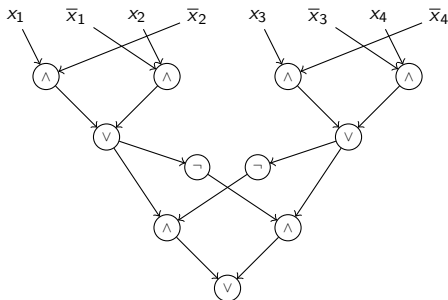
É importante notar que árvores de decisão são mais fracas do que fórmulas FNC (FND).



$$F = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$

Circuitos

Um circuito Booleano é composto de portas lógicas computando uma das funções em $\{\wedge, \vee, \neg\}$ e fios ligando estas portas lógicas como mostra a figura abaixo:



Este circuitos tem 4 variáveis de entrada (sem contar a negação de cada variável). Todos circuitos tem uma única porta lógica no nível mais alto, o valor desta porta lógica é a saída do circuito.

Circuitos

Nós dizemos que um circuito C com n variáveis de entrada computa $f : \{0, 1\}^n \rightarrow \{0, 1\}$ se $C(x) = f(x)$, para todos $x \in \{0, 1\}^n$.

- ▶ O circuito do slide anterior computa a função Parity_4 .
 - ▶ $\text{Parity}_4(x) = 1$ se e somente se $x_1 + x_2 + x_3 + x_4 \equiv 1 \pmod{2}$.

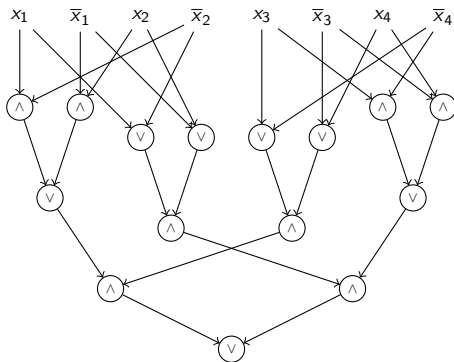
O tamanho de um circuito é o número de portas lógicas e a tua profundidade é o tamanho do maior caminho de uma variável de entrada até a porta de saída.

- ▶ O nosso circuito para Parity_4 tem tamanho 11 e profundidade 4.
- ▶ Em geral, Parity_n tem um circuito de tamanho $\mathcal{O}(n)$ e profundidade $\mathcal{O}(\log n)$.

Circuitos

Pela lei de De Morgan nós podemos empurrar as portas \neg para as variáveis de entrada.

- Se o circuito original tinha tamanho S então o circuito resultante tem tamanho $\leq 2S$.



O circuito acima também computa Parity_4 e tem tamanho 15.

Complexidade de circuitos

Dada uma função $f : \{0, 1\}^n \rightarrow \{0, 1\}$ nós queremos saber qual é o menor circuito que computa f .

- ▶ Seja \mathcal{C} o conjunto de circuitos que computam f .
- ▶ $\text{Size}(f) = \min_{C \in \mathcal{C}} \{|C|\}$.

Se $f : \{0, 1\}^* \rightarrow \{0, 1\}$ então temos que definir uma sequência de circuitos $\{C_n\}_{n \geq 1}$ onde cada C_n computa f restrita à strings de tamanho n .

- ▶ $\text{Size}(f) = \mathcal{O}(g)$ se existem constantes c e n_0 tal que $|C_n| \leq cg(n)$, para todos $n \geq n_0$.
- ▶ Como já comentamos, $\text{Size}(\text{Parity}) = \mathcal{O}(n)$.

Complexidade de circuitos: $P/poly$

Algumas classes de complexidade de circuitos:

- ▶ $P/poly$: circuitos de tamanho polinomial.
 - ▶ Contém toda a classe P .
 - ▶ Contém todas as linguagens unárias.
 - ▶ Logo contém alguns problemas indecidíveis.
- ▶ A tua versão (P-)uniforme (ou logspace-uniforme) coincide com a classe P .

Complexidade de circuitos: P/poly

- ▶ Problema em aberto: $NP \subseteq P/poly$?
 - ▶ $NP \not\subseteq P/poly$ implicaria em $P \neq NP$.
 - ▶ Teorema de Karp-Lipton: $NP \subseteq P/poly \Rightarrow PH = \Sigma_2^P$.
- ▶ Problema em aberto: Existe, para todo $k \geq 1$, uma linguagem em P que não admite circuitos de tamanho n^k ?
 - ▶ Suponha que $P \neq NP$, isto é verdade porque a classe NP não admite circuitos pequenos ou é porque circuitos para problemas em P são pequenos demais?

Complexidade de circuitos: NC e AC

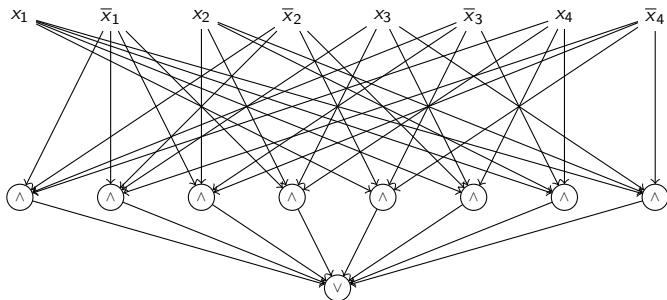
Algumas classes de complexidade de circuitos:

- ▶ NC^i : circuitos de tamanho polinomial e profundidade $\log^i n$.
- ▶ $NC = \bigcup_{i \geq 0} NC^i$.
 - ▶ Exemplo: computar a determinante de uma matriz está em NC.
- ▶ AC^i : circuitos de tamanho polinomial, profundidade $\log^i n$ e fan-in arbitrário.
- ▶ $AC = \bigcup_{i \geq 0} AC^i$.
- ▶ $NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq AC^1 \subseteq \dots \Rightarrow NC = AC$.

Complexidade de circuitos: NC e AC

Um circuito AC^0 é qualquer circuito com $fn-in$ arbitrário e que alterna portas \vee e portas \wedge .

- ▶ Por exemplo, o seguinte circuito é um circuito AC^0 para $Parity_4$.



- ▶ Todas fórmulas FNC e FND são circuitos AC^0 .

Tribes

A função $\text{Tribes}_{w,s} : \{0, 1\}^{ws} \rightarrow \{0, 1\}$ é definida da seguinte forma:

$$\text{Tribes}_{w,s}(x) = \bigvee_{i=1}^s (x_{1,i} \wedge x_{2,i} \wedge \cdots \wedge x_{w,i}).$$

Onde as variáveis são indexadas por $(i, j) \in [w] \times [s]$.

- ▶ $\text{Tribes}_{w,s}$ é trivialmente computável por um circuito FND de tamanho $s + 1$.
- ▶ Toda árvore de decisão que computa $\text{Tribes}_{w,s}$ tem que ter profundidade ws — $\text{Tribes}_{w,s}$ é evasiva.

Tribes_n

Nós estamos mais interessados na seguinte escolha de parâmetros:

- ▶ Para cada $w \geq 1$, escolhemos s o maior inteiro tal que

$$(1 - 2^{-w})^s = \Pr[\text{Tribes}_{w,s}(x) = 0] \geq 1/2.$$

- ▶ $n = ws$.

Desta forma temos que Tribes_n é uma função “imparcial”, os valores 1 e 0 aparece com basicamente a mesma probabilidade. Também temos que

- ▶ $s = \Theta\left(\frac{n}{\log n}\right)$.
- ▶ $w = \log n - \log \log n - o(1)$.

Na verdade, $s \approx 2^w \ln(2)$ e portanto $(1 - 2^{-w})^s \rightarrow 1/2$ com $w \rightarrow \infty$.

Teorema de Baker-Gill-Solovay

O teorema de Baker-Gill-Solovay diz que existem oráculos A e B tais que

- ▶ $P^A = NP^A$.
- ▶ $P^B \neq NP^B$.

Nós podemos provar que existe B tal que $P^B \neq NP^B$ (a parte não-trivial do teorema) usando o fato que a função Tribes_n é evasiva.

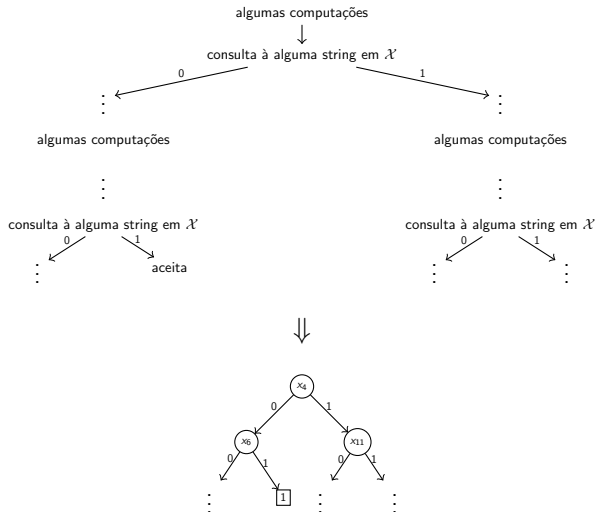
Teorema de Baker-Gill-Solovay - Prova

Seja M uma máquina de Turing de tempo polinomial que tem uma fita de oráculo e $x \in \{0, 1\}^*$. Nós consideramos o seguinte:

- ▶ \mathcal{X} um subconjunto finito de $\{0, 1\}^*$.
- ▶ $A \subseteq \{0, 1\}^* \setminus \mathcal{X}$ um oráculo.
- ▶ $T_{M^A, x}^{\mathcal{X}}$ uma árvore de decisão que recebe a string característica de um oráculo subconjunto de \mathcal{X} .
 - ▶ A string característica de $B \subseteq \mathcal{X}$ é a string x_B que é 1 no i -ésimo bit se a i -ésima string em \mathcal{X} (sobre alguma enumeração das strings binárias) está em B .
- ▶ $T_{M^A, x}^{\mathcal{X}}(x_B) = 1 \iff M^{A \cup B}(x) = 1.$

Teorema de Baker-Gill-Solovay - Prova

Se M é uma máquina de Turing de tempo polinomial e $x \in \{0, 1\}^*$.



Teorema de Baker-Gill-Solovay - Prova

Se M é uma máquina de Turing de tempo polinomial e $x \in \{0, 1\}^*$.

- ▶ No caso especial em que $\mathcal{X} = \{0, 1\}^n$, $n = |x|$.
 - ▶ $T_{M^A, x}^{\{0, 1\}^n}$ tem profundidade polilogarítmica.
 - ▶ Pois se M roda em tempo $\leq n^c$ então M faz no máximo n^c consultas ao oráculo.
 - ▶ $n^c = \text{polylog}(2^n)$.