# Digital Logic Systems

## Recitation 5: Propositional Logic contd. & Asymptotics

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

March 30, 2019

# The De Morgan dual

**Algorithm 1** DM($\phi$) - An algorithm for evaluating the De Morgan dual of a Boolean formula $\phi \in \mathcal{BF}(\{X_1, \ldots, X_n\}, \{\neg, \text{OR}, \text{AND}\})$.

1. Base Cases: (parse tree of size 1 or 2)
   1. If $\phi = 0$, then return 1.
   2. If $\phi = 1$, then return 0.
   3. If $\phi = X_i$, then return $(\neg X_i)$.
   4. If $\phi = (\neg 0)$, then return 0.
   5. If $\phi = (\neg 1)$, then return 1.
   6. If $\phi = (\neg X_i)$, then return $X_i$.
2. Reduction Rules: (parse tree of size at least 3)
   1. If $\phi = (\neg \phi_1)$, then return $(\neg \text{DM}(\phi_1))$.
   2. If $\phi = (\phi_1 \cdot \phi_2)$, then return $(\text{DM}(\phi_1) + \text{DM}(\phi_2))$.
   3. If $\phi = (\phi_1 + \phi_2)$, then return $(\text{DM}(\phi_1) \cdot \text{DM}(\phi_2))$.

**Example**

DM($\neg(X + Y)$) =?

# The De Morgan dual (cont.)

**Theorem**

*For every Boolean formula $\phi$, DM($\phi$) is logically equivalent to ($\neg\phi$).*

# Negation Normal Form

A formula is in negation normal form if negation is applied only directly to variables.

**Definition**

A Boolean formula $\phi \in \mathcal{BF}(\{X_1, \ldots, X_n\}, \{\neg, \text{OR}, \text{AND}\})$ is in *negation normal form* if the parse tree $(G, \pi)$ of $\phi$ satisfies the following condition. If a vertex in $G$ is labeled by negation (i.e., $\pi(v) = \neg$), then $v$ is a parent of a leaf labeled by a variable.

**Example**

- The formula $(\neg X) \cdot (\neg Y)$ is in negation normal form.
- The formulas $(\neg 0)$, $\neg(A \cdot B)$, $\text{NOT}(\text{NOT}(X))$ are not in negation normal form.

**Theorem**

*Let $\phi \in \mathcal{BF}(\{X_1, \ldots, X_n\}, \{\neg, \text{OR}, \text{AND}\})$. Then, $NNF(\phi)$ is logically equivalent to $\phi$ and in negation normal form.*

**Algorithm 2** NNF($\phi$) - An algorithm for computing the negation normal form of a Boolean formula $\phi \in \mathcal{BF}(\{X_1, \ldots, X_n\}, \{\neg, \text{OR}, \text{AND}\})$.

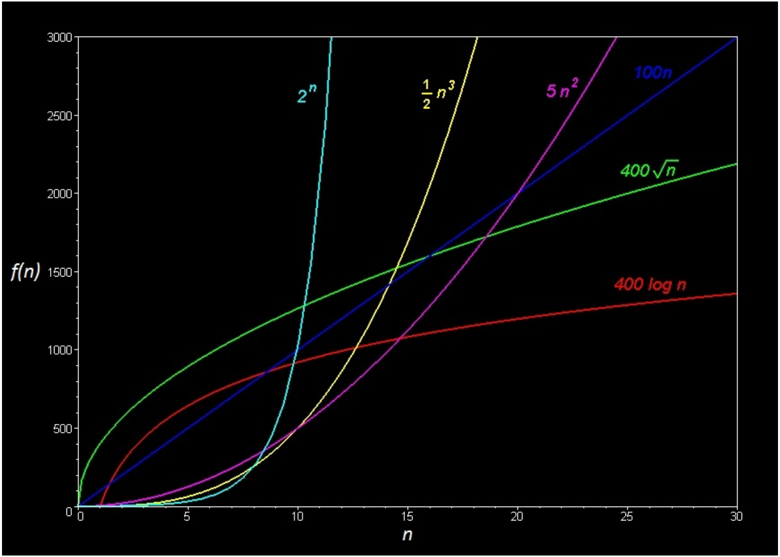
1. Base Cases: (parse tree of size 1 or 2)
   1. If $\phi \in \{0, 1, X_i, (\neg X_i)\}$, then return $\phi$.
   2. If $\phi = (\neg 0)$, then return 1.
   3. If $\phi = (\neg 1)$, then return 0.
2. Reduction Rules: (parse tree of size at least 3)
   1. If $\phi = (\neg \phi_1)$, then return DM(NNF($\phi_1$)).
   2. If $\phi = (\phi_1 \cdot \phi_2)$, then return (NNF($\phi_1$) $\cdot$ NNF($\phi_2$)).
   3. If $\phi = (\phi_1 + \phi_2)$, then return (NNF($\phi_1$) $+$ NNF($\phi_2$)).

**Example**

- $NNF(\neg\neg X) = ?$
- $NNF(\neg\neg\neg X) = ?$.

# Asymptotics

# Order of Growth - Popular functions $f(n)$



# Order of Growth: Reminder

## Definition (7.1)

Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq}$ denote two functions.

1. We say that $g(n) = O(f(n))$, if there exist constants $c_1, c_2 \in \mathbb{R}^{\geq}$ such that, for every $n \in \mathbb{N}$,

$$g(n) \leq c_1 \cdot f(n) + c_2.$$

2. We say that $g(n) = \Omega(f(n))$, if there exist constants $c_3 \in \mathbb{R}^{\geq}$, $c_4 \in \mathbb{R}^{\geq}$ such that, for every $n \in \mathbb{N}$,

$$g(n) \geq c_3 \cdot f(n) + c_4.$$

3. We say that $g(n) = \Theta(f(n))$, if $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$.

# Order of Growth: Alternative Definition

## Definition (7.2)

Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq}$ denote two functions.

1. We say that $g(n) = O(f(n))$, if there exist constants $c \in \mathbb{R}^{\geq}$ and $N \in \mathbb{N}$, such that,
$$\forall n > N : g(n) \leq c \cdot f(n) \,.$$
2. We say that $g(n) = \Omega(f(n))$, if there exist constants $d \in \mathbb{R}^{\geq}$ and $N \in \mathbb{N}$, such that,
$$\forall n > N : g(n) \geq d \cdot f(n) \,.$$
3. We say that $g(n) = \Theta(f(n))$, if $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$.

**Lemma**

*Definitions 7.1,7.2 are equivalent if $f(n) \geq 1$ and $g(n) \geq 1$, for every $n$.*

**Proof.**

On the whiteboard. □

# Order of Growth - Quick Arithmetics

It is often easy to determine the order of growth for polynomials and exponentials. Just take the dominant member:

- $n^{10} + n^9 + n^8 + n^2 + 10 = \Theta(n^{10})$
- $3^n + 2^n + n^{1000} = \Theta(3^n)$

All the constant numbers are $O(1)$.

- $0 = \Theta(1)$
- $3240009100 = \Theta(1)$

Logs of all bases have the same order of growth

- $ln(n) = \Theta(log_2(n))$
- $log_{10}(n) = \Theta(log_2(n))$

# Reminder: Is it enough to solve for powers of 2?

In the following lemma we show that, under reasonable conditions, it suffices to consider powers of two when bounding the rate of growth.

**Lemma (7.2)**

*Assume that:*

1. *The functions $f(n)$ and $g(n)$ are both monotonically nondecreasing.*
2. *The constant $\rho$ satisfies, for every $k \in \mathbb{N}$,*

$$\rho \geq \frac{g(2^{k+1})}{g(2^k)}.$$

*If $f(2^k) = O(g(2^k))$, then $f(n) = O(g(n))$.*

# Revisiting: Is it enough to solve for powers of 2? Yes!

An analogous lemma that states that $f(n) = \Omega(g(n))$ can be proved if $\frac{g(2^{k+1})}{g(2^k)} \geq \rho$, for a constant $\rho$ The lemma is as follows.

**Lemma (7.3)**

*Assume that:*

1. *The functions $f(n)$ and $g(n)$ are both monotonically nondecreasing.*
2. *The constant $\rho$ satisfies, for every $k \in \mathbb{N}$,*

$$\rho \leq \frac{g(2^{k+1})}{g(2^k)}.$$

*If $f(2^k) = \Omega(g(2^k))$, then $f(n) = \Omega(g(n))$.*

# Example - 1

Let $g(n) \triangleq \log_3 n$. We claim that $g(n) = \Theta(\log_2 n)$

**Proof.**

Recall that for every $a, b, c \in \mathbb{R}$, $a, c \neq 1$,

$$\log_a b = \frac{\log_c b}{\log_c a} . \qquad (1)$$

Hence, $\log_3 n = \frac{\log_2 n}{\log_2 3}$. Since, $5/8 < \frac{1}{\log_2 3} < 2/3$ is a constant, then $c = 2/3, d = 5/8, N = 0$ satisfy the conditions in Definition 7.2. □

Hence, when considering the order of growth of log functions with a constant base, that is $\log_c n$ and $\log_d n$ where $c, d$ are constants, we may omit the base and simply refer the order of growth of these functions as $O(\log n)$, $\Omega(\log n)$ and $\Theta(\log n)$.

# Example - 2

Let $g(n) \triangleq n^{\log_2 c}$. We claim that $g(n) = \Theta(c^{\log_2 n})$.

**Proof.**

We prove the following stronger claim.

$$n^{\log_2 c} = c^{\log_2 n}\,. \tag{2}$$

That will conclude the proof, since for every two functions $f, g : \mathbb{N} \to \mathbb{R}^{\geq}$, if $f = g$ then $f(n) = \Theta(g(n))$ and $g(n) = \Theta(f(n))$. Let us apply the $\log_2$ function on the left-hand side and the right-hand side of Eq. 2. We get

$$\log_2(n^{\log_2 c}) \stackrel{?}{=} \log_2(c^{\log_2 n}) \Leftrightarrow \log_2 c \cdot \log_2 n \quad = \log_2 n \cdot \log_2 c\,, \tag{3}$$

Where the transition follows from the fact that $\log(a^b) = b \cdot \log(a)$. Since Eq. 3 holds with equality, and since the log function is one-to-one, then their arguments are equal as well, i.e., $n^{\log_2 c} = c^{\log_2 n}$, as required. □

# Example - 3: Recurrence 1.

Consider the recurrence for every $n \in \mathbb{N}^+$

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ log_2(n) + f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases} \tag{4}$$

**Lemma**

*Find the rate of growth of the function $f(n)$ defined in Eq. 4.*

# Example - 3: Solution (1/3) - assume $n = 2^k$ and guess

Let's translate the $f(n)$ into terms of $k$

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ log_2(n) + f(\lfloor \frac{n}{2} \rfloor) & \text{if } n > 1. \end{cases} \tag{5}$$

$$f(2^k) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ k + f(2^{\text{k-1}}) & \text{if } k > 0. \end{cases} \tag{6}$$

Now, let's try to gain a guess for a closed form expression of $f(2^k)$:

$$f(2^k) = k + f(2^{k-1}) = k + (k-1) + f(2^{k-2})$$

This gives us the intuition that $f(2^k)$ can be expressed as:

$$f(2^k) = 1 + \sum_{i=1}^{k} i = 1 + \frac{k(k+1)}{2}$$

# Example - 3: Solution (2/3) - prove your guess

We will prove by an induction on $k$, that the closed form $f(2^k) = 1 + \frac{k(k+1)}{2}$ equals the recursive definition of $f(2^k)$.

- **Basis** k=0: closed form $f(2^0) = 0 \cdot \frac{0+1}{2} + 1 = 1$ is consistent with the recursive form for $f(1) = 1$
- **Hypothesis:** $f(2^k) = 1 + \frac{k(k+1)}{2}$
- **Step:** According to recursion rule: $f(2^{k+1}) = k + 1 + f(2^k)$ From induction hypothesis this is equal to

$$k+1+1+\frac{k(k+1)}{2} = 1+\frac{2(k+1)}{2}+\frac{k(k+1)}{2} = 1+\frac{(k+1)(k+2)}{2}$$

This completes our proof by induction and we are clear to state that

$$f(2^k) = 1 + \frac{k(k+1)}{2} = \Theta(1 + \frac{k(k+1)}{2}) = \Theta(k^2)$$

# Example - 3: Solution (3/3) - generalize to every $n \in \mathbb{N}^+$

We would like to show that Lemmas 7.2 and 7.3 hold:

1. Both $f$ and $g$ are monotonous non-descending. The $f(n)$ is a recurrence that with each recursive call, can only grow (see the positive terms in the Eq. 4). Whereas the $g(n)$ is the well known logarithm, which is also known to be non-descending.
2. Now we have to show that $\frac{g(2^{k+1})}{g(2^k)}$ is bounded by two constants $\rho_1$ and $\rho_2$:

$$\frac{g(2^{k+1})}{g(2^k)} = \frac{(k+1)^2}{k^2} = 1 + \frac{2}{k} + \frac{1}{k^2}$$

Observe that the rightmost expression can be bounded:

$$\rho_1 = 1 \le 1 + \frac{2}{k} + \frac{1}{k^2} \le 4 = \rho_2$$

Explanation: $0 \le 2/k \le 2$ and $0 \le 1/k^2 \le 1$

Since Lemmas 7.2 and 7.3 hold, we generalize our $\Theta$ bound:

$$f(n) = \Theta((\log n)^2)$$

# Recurrence Trees

- Recurrence tree is a way of illustrating the recursive calls of a function.
- Given a recursive function $f(n)$:
  - The root will represent the initial function call
  - The internal nodes represent the intermediate calls
  - The leaves represent reaching the base rules
- Each node of a tree is associated with a function argument and a penalty.

# Recurrence Trees - The Recipe

1. Given $f(n)$ - construct a tree, assume $n = 2^k$ if necessary.
2. Determine $L$ - the number of tree levels.
3. For every level $i \in [0, .., L-1]$ determine $penalty_i$
4. Obtain a guess $f(2^k) = \sum_{i=0}^{L-1} penalty_i$
5. Generalize the guess to obtain terms of $f(n)$

# Recurrence Trees - Example

**Question**

Find the $\Theta$ bound for the following recursive $f(n)$:

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ 2 \cdot f\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n & \text{if } n > 1. \end{cases}$$

# Recurrence Trees - Example - Assume $n = 2^k$

## Question

Find the $\Theta$ bound for the following recursive $f(n)$:

$$f(n) \triangleq \begin{cases} 1 & \text{if } n = 1 \\ 2 \cdot f(\lfloor \frac{n}{2} \rfloor) + n & \text{if } n > 1. \end{cases}$$

## We assume $n = 2^k$ and translate the function

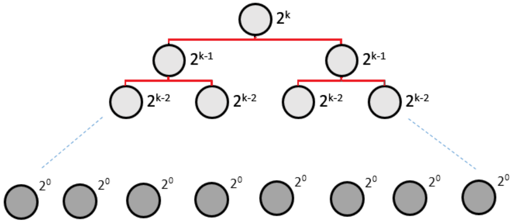
$$f(2^{\mathrm{k}}) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2 \cdot f(2^{k-1}) + 2^k & \text{if } k > 0. \end{cases}$$

# Recurrence Trees - Example - Draw the tree

**Under assumption $n = 2^k$**

$$f(2^k) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2 \cdot f(2^{k-1}) + 2^k & \text{if } k > 0. \end{cases}$$

We notice that the arity of the tree is 2.



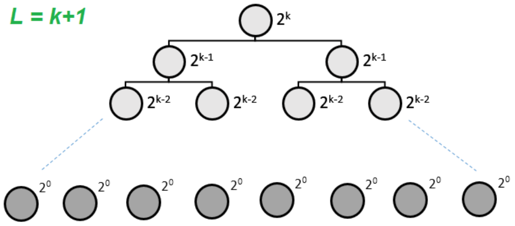
## Under assumption $n = 2^k$

$$f(2^{\mathrm{k}}) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2 \cdot f(2^{k-1}) + 2^k & \text{if } k > 0. \end{cases}$$

## Thumb rule

If each recursive call cuts the function argument $(n)$ by a factor of $b$, then the tree depth is $log_b(n)$, number of levels is $log_b(n) + 1$.

**L = k+1**

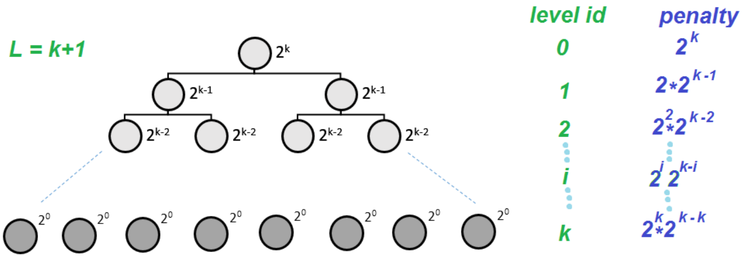


# Recurrence Trees - Example - Penalty at each level

**Under assumption $n = 2^k$**

$$f(2^k) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2 \cdot f(2^{k-1}) + 2^k & \text{if } k > 0. \end{cases}$$

We notice the penalty at each single call.

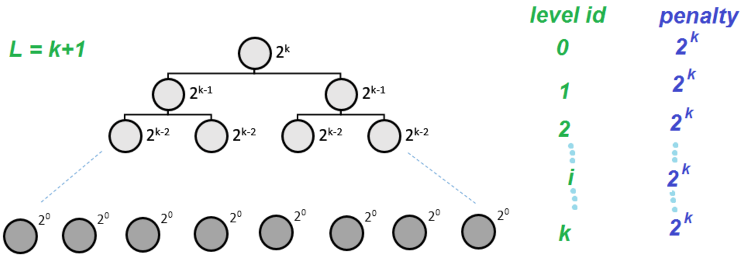


# Recurrence Trees - Example - Penalty at each level

**Under assumption $n = 2^k$**

$$f(2^k) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2 \cdot f(2^{k-1}) + 2^k & \text{if } k > 0. \end{cases}$$

We notice the penalty at each single call.



# Recurrence Trees - Example - Sum up the penalties

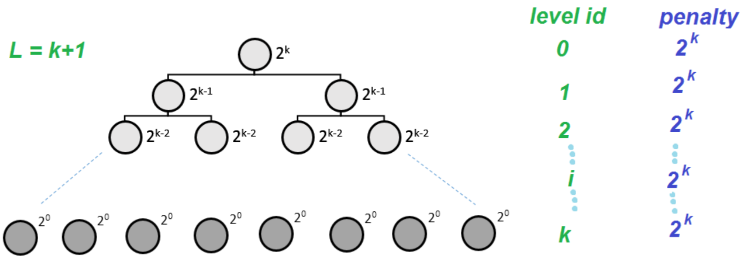
**Under assumption $n = 2^k$**

$$f(2^k) \triangleq \begin{cases} 1 & \text{if } k = 0 \\ 2 \cdot f(2^{k-1}) + 2^k & \text{if } k > 0. \end{cases}$$

We sum up the penalties over all the levels and obtain a guess:

**A guess**

$f(2^k) = \sum_{i=0}^{L-1} penalty_i = (k+1) \cdot 2^k = \Theta(k \cdot 2^k)$



We use our good old lemmas 7.2, 7.3 in order to generalize the bound to all $n \in \mathbb{N}^+$

### Lemma 7.2, 7.3 justification

1. Clearly $f(n)$,$g(n)$ are monotonous non-decreasing.
2. $\frac{g(2^{k+1})}{g(2^k)} = \frac{(k+1) \cdot 2^{k+1}}{k \cdot 2^k} = \frac{(k+1) \cdot 2 \cdot 2^k}{k \cdot 2^k} = 2 + \frac{2}{k}$ is bounded by 2 and 4.

Since we proved that $f(2^k) = \Theta(k \cdot 2^k)$, the lemmas imply that $f(n) = \Theta(n \cdot log(n))$