Soham Chatterjee
Email: sohamc@cmi.ac.in
Course: Parallel Algorithms and Complexity

Assignment - 5
Roll: BMC202175
Date: August 28, 2023

Problem 1

Prove that $GapNC^1 = DiffNC^1$

Solution: $GapNC^1$ is in $DiffNC^1$ because if $f - g \in GapNC^1$ then we can compute f and g normally and at the top there is a addition gate where the coefficient of g is -1. So $GapNC^1 \subseteq DiffNC^1$.

Suppose $f \in DiffNC^1$. Now we will try to move the negative coefficients upwards and when it reaches to the children of the topmost gate we will transform it to the form of $g - h$. Since it is $DiffNC^1$ we can assume fanin is 2. If there are no -1 coefficients for any child of a gate we will do nothing. Now we can transform the gates like this

$(-f) \times (-g) = f \times g$	remove the -1's from children
$(-f) \times g = -(f \times g)$	remove the -1's from children and add to the edge outgoing from the gate
$(-f) + (-g) = -(f + g)$	remove the -1's from children and add to the edge outgoing from the gate

Apart from these cases one case is remaining. The last case is $(-f) + g$ Then suppose from the addition gate it is going to another gate g. Then we make two copies of g where one has the child $-f$ and the other one has child g in place of the original $-f + g$ in the single gate g. Then we do the process above again.

Now if $-f + g$ is the case where the $+$ is the topmost gate then we replace the $+$ gate with $-$ and the children of that gate will be g and f respectively. If $(-f) + (-g)$ is the case where the additoin gate is the topmost gate then compute it by

$$(-f) + (-g) = 1 - 1 + (-f) + (-g) = 1 - (1 + f + g)$$

Also if the $(-f) \times g$ is the topmost $\times$ gate then we compute it like

$$(-f) \times g = -(f \times g) = 1 - (1 + f \times g)$$

So we replace the $(-f) \times g$ with $1 - (1 + fg \times g)$.

In this way the negative coefficients moves upwards and in the end we have a $h - k$ form at the top which satisfies the $GapNC^1$ conditions. Also with this reconstructing at most every gate is duplicated and at the top at most 1 more gate appeared. So Size blow up is constant. Therefore $DiffNC^1 \subseteq GapNC^1$. Hence $GapNC^1 = DiffNC^1$.

$\square$

Problem 2

Formula depth reduction

Solution: Suppose ϕ is a formula. Lets assume its fanin is 2. Suppose the size of ϕ is s. Now starting from the root walk down to the leaves by aleays taking the child with a larger subtree under it. Consider the first node in this path v such that the size of the formula rooted at v is smaller than $\frac{2s}{3}$. Let ϕ_v be the subformula rooted at v, By the choice of the path from the root we have

$$\frac{s}{3} \leq |\phi_v| < \frac{2s}{3}$$

Let ϕ'_v be the formula if the subformula rooted at v is replaced by a variable y. Hence

$$\phi'_v(y) = Ay + B \qquad \text{and} \phi = A\phi_v + B$$

for some polynomials A and B. But we can compute both A and B from $\phi'_v(y)$ as

$$A = \phi'_v(1) - \phi'_v(0) \qquad B = \phi'_v(0)$$

Thus

$$f = (\phi'_v(1) - \phi'_v(0))\phi_v + \phi'_v(0)$$

All the formulas $\phi'_v(1)$, $\phi'_v(0)$, ϕ_v have size at most $\frac{2s}{3}$. Thus by recursvely applying the process on each of the subformulas we have

$$Depth(s) = Depth\left(\frac{2s}{3}\right) + 3 \implies Depth(s) = O(\log s)$$

and

$$Size(s) \leq 4Size\left(\frac{2s}{3}\right) + O(1) \implies Size(s) = poly(s)$$

$\square$