Advanced Proof Systems - Problem Set 4

Yosef Goren

January 18, 2023

1 $IP \subseteq PSPACE$

1.1

Given a graph with n vertices, consider the following series of matrices:

$$M_k[i,j] := \begin{cases} 1 & \text{exists a path from } i \text{ to } j \text{ with length } \leq k \\ 0 & \text{otherwise} \end{cases}$$

Corollary: M_1 is the adjacency matrix of the graph.
Between any two matrices of size $n \times n$, define the following operation:

$$(A \otimes B)[i,j] := \bigvee_{x \in [n]} (A[i,x] \wedge B[x,j])$$

.

Lemma:

$$\forall k \in [n] : M_k \otimes M_k = M_{2k}$$

. Proof: Assume (by induction) for M_k [1]. Now, we need to prove for M_{2k}.
Consider a path $l_{i,j}$ from i to j with length $\leq 2k$.
It can be decomposed into two paths of length $\leq k$; The first path from i to x ($l_{i,x}$) and the second path from x to j ($l_{x,j}$).
Hence $M_{2k}[i,j] \Rightarrow \exists x \in [n] : M_k[i,x] \wedge M_k[x,j] \Rightarrow (M_k \otimes M_k)[i,j]$.
Conversely, if $(M_k \otimes M_k)[i,j] = 1$, then there exists $x \in [n]$ such that $M_k[i,x] = 1$ and $M_k[x,j] = 1$, which means that there exists a path from i to x of length $\leq k$ and a path from x to j of length $\leq k$. So $(M_k \otimes M_k)[i,j] \Rightarrow M_{2k}[i,j]$.
Finally we have that $M_k \otimes M_k = M_{2k}$.

[1] formally this would not prove for k values that are not a power of 2, but we actually do not make use of those sizes anyways since $\forall k \geq n, M_k = M_n$ and we only care about M_n.

Algorithm: The algorithm which the circuit will follow is as follows:

$k \leftarrow 1$
$M \leftarrow E$
while $k < n$ **do**
    $M \leftarrow M \otimes M$
    $k \leftarrow 2k$
end while
return $M[s, t]$.

Due to the lemma, at the end of each iteration - $M = M_k$, this means that when the loop ends, $M = M_n$. Thus $M[s, t]$ is the correct output.

Circuit: Here we assume the circuit recives E, s, t as input, we assume E is represented in the adjacency matrix form.
The $\otimes$ operation can be implemented with depth $O(log(n))$ using a binary tree of the big $\vee$ gate. Additionally $log(n)$ steps are required for $k := n$ - each step is a set of layers added to the circuit.
With $log(n)$ complexity we can implement a MUX gate to select s, t from M after the layer.
Hence the total depth of the circuit is $O(log(n)^2)$. The maximum width of the circuit is $O(n^2)$.

1.2

Let $L \in PSPACE$.
Hence exists a polynomial space nondeterministic turing machine M_L that decides it.
Consider the set of configurations possible for M_L. Since it's set of states is constant and the set of values for it's tape is exponentionally bounded by a polynome - the set of possible configurations for it is bounded exponentionally by a polynome.

Each such configuration can be represented as a vertex in the graph of all possible configurations. Moreover, each possible transition from one configuration to another can be represented as an edge in the graph. Denote this configuration graph as G.

Define the reduction f as $f(x) = (G, s, t)$ where s is the vertex corresponding to the initial configuration of M_L on the input x, and t is vertex corresponding to the accepting configuration [2].

[2]WLOG assume there is one accepting configuration, otherwise we can either construct a new turing machine that clears it's tape and then accepts, or add to the graph a new node (t) that is connected to all accepting configuration vertecies.

1.3

Let $L \in PSPACE$.
Due to the reduction from the previous question, we know L can be solved by a poly turing machine for the ST-connectivity problem as a graph with size exponential w.r. to n (where $x \in L, n = |x|$).
Also, due to the first question, we have a family of circuits that accept or reject these exponential graphs with $size = poly(|G|) = poly(exp(n)) = exp(n), depth = polylog(|G|) = polylog(exp(n)) = poly(n)$.

1.4

Let $L \in PSPACE$.
From the last question we have that it has an exponential size poly depth circuit family [3], hence by the reduction seen in class - it has an IP with verifier running in time $(poly(n) + n)polylog(exp(n)) = poly(n)$. Hence - $L \in IP$

□

1.5

The complexity of the prover seen in class is $poly(S)$, where S is the size of the circuit so the prover time is $poly(exp(n))$ or just $exp(n)$.

2 Batch Verification for P

Let $L \in NP$ be decided by a family of logspace uniform circuits of size $\tilde{O}(n)$ - $\{C\}_{i\in\mathbb{N}}$.
Let $L = \{(x_1, ..., x_k) \mid \forall i \in [k], x_i \in L \; wedge |x_i| = |x_1| = n\}$.
We also know that $t, k = poly(n)$.

Define the following family of circuits $C' := \{C'_i\}_{i\in\mathbb{N}}$:
The circuit C'_i is defined as the concatenation of C_i; k times side by side, followed by a tree of '$\wedge$' operators that combine the outputs of all C_i instances to a single output.

Since for each area of the input of C'_i that input is processed by the circuit C_i, and since the final result of C'_i is to accept iff $\forall j \in [k], x_j \in L$ - then C' accepts L' (note the sizes must match also to be a valid input to C'_i).

Now consider the size and depth of C'; it's size is k times the size of C's circuits, with the addition of the tree of '$\wedge$' operators. Meaning the size is $\leq \tilde{O}(t) \cdot k + 2 \cdot log(k)$. The depth of C is bounded by it's size, hence the depth of C' is $\leq \tilde{O}(t) + log(k)$.

[3] Exponential size'd family sounds amazing

Now consider the theorem on compiling logspace uniform circuits into IP's seen in lecture:

Theorem 1 (logspace uniform circuits compiler) *If L is accepted by logspace uniform circuits of depth $d(n)$ and of size $poly(n)$, then L has an Interactive proof which the prover runs in $poly(n)$ time and the verifier runs in $(n + d)polylog(n)$ time with $d \cdot polylog(n)$ rounds.*

Thus exists an IP proof for L' based on C', s.t. the prover runs in time $poly(|x'|)$ where $x' \in L'$ or $poly(n \cdot k)$, there are up to $d \cdot polylog(n \cdot k) rounds$ and the verifier runs in time $(n \cdot k + d)polylog(n \cdot k)$.
Now all that is left is to evaluate the complexities:

- Prover:
  As required $poly(n \cdot k) \leq poly(n, k, t)$.
- Verifier:

$$(n \cdot k + d)polylog(n \cdot k) \leq (n \cdot k + \tilde{O}(t) + log(k))polylog(n \cdot k)$$
$$= (n{\cdot}k{+}t{\cdot}polylog(t){+}log(k))polylog(n{\cdot}k) = (n{\cdot}k{+}t{\cdot}polylog(poly(n)){+}log(k))polylog(n{\cdot}k)$$
$$= (n \cdot k)polylog(n \cdot k) + t \cdot polylog(n)polylog(n \cdot k) + log(k)polylog(n \cdot k)$$
$$= \tilde{O}(n \cdot k) + t \cdot polylog(n \cdot k) + log(k)polylog(n \cdot k)$$
$$= {}^{4}\tilde{O}(n \cdot k) + t \cdot polylog(n) + log(k)polylog(n)$$
$$= \tilde{O}(n \cdot k) + (t + log(k))polylog(n)$$

- Communications:

$$d \cdot polylog(n \cdot k) = \tilde{O}(t)polylog(k)polylog(n \cdot k) = \tilde{O}(t)polylog(n)^{5}$$

[4] $polylog(n \cdot k) = polylog(n \cdot poly(n)) = poly(log(n \cdot n^{c})) = poly((c+1)log(n)) = polylog(n)$
[5] is the same as $(log(k) + t)polylog(n)$.