Lecture 19
Linear-Size IOPs with Sublinear-Time Verification

We have proved that arithmetic “circuit-like” computations have linear-size IOPs:

for every field \( \mathbb{F} \) of size \( \Omega(n) \) that is smooth [smoothness is for the LDIT],

\[
\text{RCS(\mathbb{F})} \leq \text{IOP} \left[ \varepsilon_c = 0, \varepsilon_s = 0.5, \Sigma = \mathbb{F}, p_t = O(n\log n), v_t = O(n) \right.
\]

\[
\left. k = O(\log n), r = O(\log n), l = O(n), g = O(\log n) \right] \]

The running time of the verifier is optimal, because just reading the statement takes \( \Omega(n) \) time. Similarly to before if we seek sublinear-time verification we need to consider problems whose description is smaller than computation size.

The holy grail would be a statement like the following:

\[
\text{NTIME}(T) \leq \text{IOP} \left[ \varepsilon_c = 0, \Sigma = \{0,1\}, p_t = O(T), v_t = \text{poly}(n, \log T) \right. 
\]

\[
\left. \varepsilon_s = 0.5, \ k = *, \ l = O(T), \ q = \text{poly}(\log T) \right] 
\]

This remains a challenging open question.

Instead, we will prove a “large alphabet” relaxation of the theorem:

**Theorem:** for every field \( \mathbb{F} \) of size \( \Omega(T) \) that is smooth [smoothness is for the LDIT]

\[
\text{NTIME}(T, \mathbb{F}) \leq \text{IOP} \left[ \varepsilon_c = 0, \Sigma = \mathbb{F}, p_t = O(T\log T), v_t = \text{poly}(n, \log T) \right. 
\]

\[
\left. \varepsilon_s = 0.5, \ k = *, \ l = O(T), \ q = \text{poly}(\log T) \right] 
\]
Machine Computations

Informally, a machine is an automaton that can read/write to some type of memory. If memory = tapes then you get Turing machines. If memory = RAM then you get register machines (very close to how we think of a computer).

We are going to define languages that model machines that compute over finite fields. Let's start simple by first doing this for automata (i.e., no memory beyond internal states).

Consider:
- \( k \in \mathbb{N} \) — number of internal registers, i.e., a state is \( \mathbb{F}^k \)
- \( E : \mathbb{F}^k \to \mathbb{F}^k \) — transition function mapping current state to next state

A \( T \)-step computation looks as follows

\[
\begin{array}{cccccccc}
S_0 & \quad & S_1 & \quad & S_2 & \quad & \cdots & \quad & S_T \\
E & \quad & E & \quad & E & \quad & \cdots & \quad & E \\
\vdots & \quad & \vdots & \quad & \vdots & \quad & \vdots & \quad & \vdots \\
1 & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
2 & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\vdots & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
k & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\end{array}
\]

Specifying the computation requires \( O(1E1 + \log T) \) bits.

The specified computation involves \( O(1E1 \cdot T) \) operations, exponentially more in \( T \).

We are in fact interested in non-deterministic computations, and need an appropriate language.
Algebraic Automata

The bounded-halting problem for automata:

\[ \text{def: } BH = \{ (E, z, T) \mid E: \mathbb{F}^k \to \mathbb{F}^k, z \in \mathbb{F}^n, T \in \mathbb{N} \text{ for which} \]

\[ \exists \text{ execution trace } A_1, \ldots, A_k : [T] \to \mathbb{F} \text{ s.t.} \]

1. each step follows the transition function: \( \forall t \in \{0, \ldots, T-1\}, E(A_1(t), \ldots, A_k(t)) = A_1(t+1), \ldots, A_k(t+1) \)
2. the first \( n \) values of \( A_i \) are \( z \): \( A_1|_{[n]} = z \)
3. the last value of \( A_i \) is 0: \( A_i(T) = 0 \)

Let's massage this into a more convenient problem:

- Identify \([T]\) with a multiplicative subgroup \( H = \langle \omega \rangle \subseteq \mathbb{F} \) s.t. \( |H| = T \).
  
  Crucially, representing \( H \) requires only \( O(\log |H|) \) bits, rather than \( O(|H| \log |H|) \).
- We are interested to check not compute, so we translate the circuit \( E: \mathbb{F}^k \to \mathbb{F}^k \)
  into quadratic equations \( p_1, \ldots, p_m \in \mathbb{F}[X_1, \ldots, X_{k+e}] \) with \( m := O(l|E|) \) and \( l := O(l|E|) \) auxiliary vars

\((E, z, T) \in BH \iff \exists \text{ augmented execution trace } A_1, \ldots, A_k, B_1, \ldots, B_e : H \to \mathbb{F} \]

- \( \forall t \in \{0, \ldots, T-1\}, \{ p_1(A_1(t), \ldots, A_k(t), A_1(t+1), \ldots, A_k(t+1)) = 0 \} \forall j \in [m] \)
- \( A_1|_{H_0} = z, A_1(\omega^{-1}) = 0 \)

\( \text{size } (k + e + T = O(l|E|)) \)
Consider the setting where the verifier has oracle access to a function \( f : L \rightarrow \mathbb{F} \) and wishes to check that \( \hat{f}|_H = \mathbb{1} \) for a given "target" function \( \tilde{z} : H \rightarrow \mathbb{F} \). (E.g., \( \tilde{z} \) is all 0's.)

We have seen this before: \( \hat{f}(x) \) vanishes on \( H \) if \( \exists \hat{h}(x) \) s.t. \( \hat{f}(x) - \tilde{z}(x) = \hat{h}(x) v_H(x) \)

Hence:

\[
\begin{align*}
P(L,F,L,H,\tilde{z}),f) & \quad f : L \rightarrow \mathbb{F} \\
\text{Compute } \hat{h}(x) & = \frac{\hat{f}(x) - \tilde{z}(x)}{v_H(x)} \\
h & : L \rightarrow \mathbb{F}
\end{align*}
\]

\[ V(L,F,L,H,\tilde{z}) \]

- Test that \( h \) is \( \delta \)-close to \( RS[F,L,d-1|H|] \)
- Sample \( \forall \in L \) and check \( f(\bar{x}) - \tilde{z}(\bar{x}) \approx h(\bar{x}) v_H(\bar{x}) \)

**Completeness:** if \( \hat{f}|_H = \mathbb{1} \) then \( h := \hat{h}|_L \in RS[F,L,d-1|H|] \) and passes check \( \forall \in L \)

**Soundness:** if \( \hat{f}|_H \neq \mathbb{1} \) then \( \forall h : L \rightarrow \mathbb{F} \) we have two cases:

- \( h \) is \( \delta \)-far from \( RS[F,L,d-1|H|] \) \( \rightarrow \) verifier accepts w.p. \( \leq \varepsilon_{\text{orr}}(\delta) \)
- \( h \) is \( \delta \)-close to \( \hat{h} \) of degree \( d-1|H| \) \( \rightarrow \hat{f}(x) - \tilde{z}(x) \neq \hat{h}(x) v_H(x) \) so verifier accepts w.p. \( \frac{d}{|H|} + \delta \)

Time complexity of the verifier:

- Ignore LDT because if using FRI, \( t_{\text{orr}} = O(\log |H|) \), which is small
- if \( \tilde{z} \neq 0^H \) then: evaluate \( v_H \) at \( \bar{x} \) and evaluate \( \hat{z} \) at \( \bar{x} \) \( \rightarrow \text{poly}(|H|) \)
- if \( \tilde{z} = 0^H \) then: evaluate \( v_H \) at \( \bar{x} \) \( \rightarrow \text{poly}(|H|) \) in general but \( \text{poly}(\log |H|) \) if \( H \) is a subgroup!

E.g., if \( H \) is a multiplicative subgroup then \( v_H(x) = x^{1|H|-1} \). Crucial for us today.
IOP for Algebraic Automata

\[\text{V}(\langle E, \mathbb{Z}, T \rangle)\]

- Test each of the received function for the appropriate degree.
  [we will come back to this]

- Sample \( \mathbb{T} \) and check that:
  - for each \( k \):
    \[
    h_j(x) V_{\mathbb{T}}(x) = \prod_{e \in [k]} \left( f_i(x), \ldots, f_k(x), g_i(x), \ldots, g_k(x) \right)
    \]
  - for each \( i \):
    \[ h_i(x) V_{\mathbb{T}}(x) = f_i(x) - \hat{\Delta}(x) \]
  - for each \( o \):
    \[ h_o(x) (x - \omega^{-1}) = f_o(x) \]
Completeness

Suppose that $A_1, \ldots, A_k : [T] \rightarrow F$ is a witness for $(E, z, T) \in \mathcal{B}H$.

- The prover can evaluate $E$ at each time step to augment the trace with $B_1, \ldots, B_L : [T] \rightarrow F$ that satisfy all $m$ quadratic equations $p_1, \ldots, p_m$ derived from $E$. So the prover can find $\hat{h}_1(x), \ldots, \hat{h}_m(x)$.

- $A_i$ agrees with $z$ on first $n$ entries, and is 0 on last entry so $\hat{h}_i, h_0$ too can be found.

Moreover:
- **Proof length**: $O((k + L + m)1L1) = O((k + L + m)1H1) = O(1E1 \cdot T) \cdot \text{elts}$
- **Query complexity**: $O((k + L + m) \log 1L1) = O(1E1 \log T)$
- **Prover time**: $O((k + L + m)1H1 \log 1L1) = O(1E1T1 \log T)$
- **Verifier time**: $O((k + L + m) \log 1L1 + \text{poly}(n)) = O(1E1 \log T) + \text{poly}(n)$
Soundness

Suppose that $(E, z, T) \notin BH$.

There are two cases:

1. One of the functions is far from $RS$.
   - If $i \in [k]$ then $f_i$ is $\delta$-far from $RS[F_i, L, LH^{-1}]$.
   - If $j \in [l]$ then $g_j$ is $\delta$-far from $RS[F_j, L, LH^{-1}]$.
   - If $m \in [m]$ then $h_m$ is $\delta$-far from $RS[F_m, L, LH^{-1}]$.
   - $h_0$ is $\delta$-far from $RS[F_0, L, LH^{-1}]$.

   $\Rightarrow$ verifier accepts w.p. $\leq \epsilon_{\text{loss}}(\delta)$.

2. All functions are close to (unique) polynomials $\{\hat{f}_i: i \in [k], \hat{g}_j: j \in [l], \hat{h}_m: m \in [m], \hat{h}_0\}$ of the appropriate degree.
   - If $j \in [m]$ then $\hat{h}_j(x) V_{ih}(x) \neq p_3(f_j(x), \hat{f}_j(x), \hat{g}_j(x))$.
   - The consistency test accepts w.p. $\leq \frac{1}{14} + \frac{2(2k+l)\delta}{14}$.

Several options to make this $\leq 1$:

- Set proximity parameter $\delta = O(\frac{1}{\sqrt{\log t}}) = O(\frac{1}{16})$.
  - This requires setting repetition parameter $t$ in $FRI$ to $t = O(16\log)$ to ensure that $E[\epsilon(\delta)] = O(\delta)$.

- Repeat consistency test $t = O(\log|E|)$ times, as the term becomes $\frac{4(4k+2l)\delta}{14}$.

- Send random coefficients to prove a test $\sum_i x_i \bar{f}_i + \sum_i x_i \bar{g}_i$ instead of individually.

- Distortion statements imply the error becomes $\frac{2(4k+2l)\delta}{14}$ (due to column distance).
We now add memory:

If we extend the state with all of memory, we end up with $T^2$ variables—well beyond linear.

**Observation**: it suffices to check correctness of memory operations, “what you work is what you read.”

Consider the memory trace ordered first by address and then by time stamp:

<table>
<thead>
<tr>
<th>op</th>
<th>addr</th>
<th>time</th>
<th>val (read or written)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>2</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>r</td>
<td>1</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>2</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The trace is correct iff for every two adjacent pairs

$$(\text{op}, \text{addr}, \text{time}, \text{val}), (\text{op}', \text{addr}', \text{time}', \text{val}')$$

the following holds

- if $\text{addr} = \text{addr}'$ then $\text{time} < \text{time}'$ and $(\text{op}=r \Rightarrow \text{val}'=\text{val})$
- if $\text{addr} \neq \text{addr}'$ then $\text{addr} < \text{addr}'$

This leads to a language that represents machine computations...
Memory from a Permuted Trace

**Lemma:** There is a polynomial-time reduction \( R \) s.t.

- \( R(E, \tau, T) \) outputs quadratic equations \( p_1, \ldots, p_m \in \mathbb{F}[x_1, \ldots, x_{mk+n}] \) with \( m, l = O(|E|) \)
- \( (E, \tau, T) \in \text{BH} \) if and only if there exists an augmented execution trace \( A_1, \ldots, A_k, B_1, \ldots, B_e : H \to \mathbb{F} \) and a permutation \( \pi : [T] \to [T] \) such that
  
  \[ \forall \tau \in \{0, \ldots, T-1\} : \left\{ \begin{array}{l}
  p_j \left( A_i(\omega^k) \ldots, A_k(\omega^k), A_i(\omega^m) \ldots, A_k(\omega^m), B_i(\omega^k), \ldots, B_e(\omega^k) \right) = 0 \\
  A_1|_{H_{in}} = 2, \ A_1(\omega^{T-1}) = 0
  \end{array} \right\} \forall j \in [m]
  \]

**Proof:** Set \( p_1, \ldots, p_m \) to be the quadratic equations obtained by translating the transition function and also the logic for "what you wrote is what you read".

**Completeness:** Choose \( \pi \) to be the permutation that reorders the trace by address then time, so that the memory checks pass.

**Soundness:** for any choice of permutation \( \pi \), either some memory check fails, or the read/write operations are all correct so the transition function is fed the correct values.
Consider the setting where the verifier has oracle access to \( f, g : L \rightarrow \mathbb{F} \) and wishes to check the claim:

\[ \exists \pi : H \rightarrow H \text{ s.t. } \forall a \in H \quad \hat{g}(a) = \hat{f}(\pi(a)) \]

Idea: the condition is equivalent to asking if \( \{ \hat{g}(a) \}_{a \in H} \) and \( \{ \hat{f}(\pi(a)) \}_{a \in H} \) equal as multisets, which in turn is true if

\[ \prod_{a \in H} (x - \hat{g}(a)) = \prod_{a \in H} (x - \hat{f}(\pi(a))). \]

This directly leads to a protocol when \( H = \langle w \rangle \):

\[ P((L, H), (f, g)) \]

Compute partial products:
- \( f_{\pi} : L \rightarrow \mathbb{F} \text{ s.t. } \hat{f}_{\pi}(w^i) = \prod_{j < i} (r - \hat{f}(w^j)) \)
- \( g_{\pi} : L \rightarrow \mathbb{F} \text{ s.t. } \hat{g}_{\pi}(w) = \prod_{j < i} (r - \hat{g}(w^j)) \)

Compute \( h_1, h_2, h_3, h_4, h_5 : L \rightarrow \mathbb{F} \text{ s.t.} \)

\[ \hat{h}_1(x) = \frac{f(x) - (r - f(x))}{v_{\pi}(x)/(x-1)} \]
\[ \hat{h}_2(x) = \frac{g_{\pi}(x) - (r - g_{\pi}(x))}{v_{\pi}(x)/(x-1)} \]
\[ \hat{h}_3(x) = \frac{f_{\pi}(x)(x - w)}{v_{\pi}(x)/(x-1)} \]
\[ \hat{h}_4(x) = \frac{g_{\pi}(x)(x - w)}{v_{\pi}(x)/(x-1)} \]
\[ \hat{h}_5(x) = \frac{f_{\pi}(x) - g_{\pi}(x)}{(x - w^{T-1})} \]

\[ V((L, H)) \]

Sample \( r \leftarrow \mathbb{F} \)

- Test that all received functions are LD.
- Sample \( r \in L \) and check:

\[ h_1(x) \cdot \frac{v_{\pi}(x)}{x - 1} = f_{\pi}(x) - (r - f(x))f_{\pi}(w^i) \]
\[ h_2(x) \cdot (x - 1) = f_{\pi}(x) - (r - f(x)) \]
\[ h_3(x) \cdot \frac{v_{\pi}(x)}{x - 1} = g_{\pi}(x) - (r - g_{\pi}(x))g_{\pi}(w^i) \]
\[ h_4(x)(x - 1) = g_{\pi}(x) - (r - g_{\pi}(x)) \]
\[ h_5(x)(x - w^{T-1}) = f_{\pi}(x) - g_{\pi}(x) \]