Lecture 19
Linear-Size IOPs with Sublinear-Time Verification

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

\[
\text{RICKS(FF) \leq IOp} \left[ \begin{array}{l}
\epsilon_c = 0, \\
\Sigma = \mathbb{F}, \\
p_t = O(n \log n), \\
\nu_t = O(n), \\
k = O(\log n), \\
r = O(\log n), \\
l = O(n), \\
g = O(\log n)
\end{array} \right]
\]

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:

\[
\text{NTIME}(T) \leq IOp \left[ \begin{array}{l}
\epsilon_c = 0, \\
\Sigma = \{0, 1\}^T, \\
p_t = O(T), \\
\nu_t = \text{poly}(n, \log T), \\
k = * \\
l = O(T), \\
g = \text{poly}(\log T)
\end{array} \right]
\]

This remains a challenging open question.

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

\[
\text{NTIME}(T, \mathbb{F}) \leq IOp \left[ \begin{array}{l}
\epsilon_c = 0, \\
\Sigma = \mathbb{F}, \\
p_t = O(T \log T), \\
\nu_t = \text{poly}(n, \log T), \\
k = * \\
l = O(T), \\
g = \text{poly}(\log T)
\end{array} \right]
\]

Theorem: for every field FF of size \(\Omega(T)\) that is smooth [smoothness is for the LDI].
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 state).

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

A T-step computation looks as follows:

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.
The bounded-halting problem for automata:

\begin{itemize}
  \item \textbf{BH} is the set of instances \((E, z, T)\) where \(E : \mathbb{F}^k \to \mathbb{F}^k\), \(z \in \mathbb{F}^n\), \(T \in \mathbb{N}\) for which
  \item there exists an execution trace \(A_1, \ldots, A_k : [T] \to \mathbb{F}\) s.t.
  \item each step follows the transition function: \(\forall t \in \{0, 1, \ldots, T-1\} \quad E(A_1(t), \ldots, A_k(t)) = A_1(t+1), \ldots, A_k(t+1)\)
  \item the first \(n\) values of \(A_1\) are \(z\): \(A_1|_{[n]} = z\)
  \item the last value of \(A_1\) is 0: \(A_1(T) = 0\)
\end{itemize}

Let's massage this into a more convenient problem:

\begin{itemize}
  \item Identify \([T]\) with a multiplicative subgroup \(H = \langle \omega \rangle \leq \mathbb{F}\) s.t. \(|H| = T\).
    
    Crucially, representing \(H\) requires only \(O(\log |\mathbb{F}|)\) bits, rather than \(O(|H| \log |\mathbb{F}|)\).
  \item We are interested to check not compute, so we translate the circuit \(E : \mathbb{F}^k \to \mathbb{F}^k\)
    into quadratic equations \(p_1, \ldots, p_m \in \mathbb{F}[x_1, \ldots, x_{k+1}]\) with \(m = O(1E1)\) and \(l = O(1E1)\) auxiliary vars
\end{itemize}

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

\begin{itemize}
  \item \(\forall t \in \{0, \ldots, T-1\} : \forall j \in [m] \quad p_j(A_1(t \omega), \ldots, A_k(t \omega), A_1(t \omega^j), \ldots, A_k(t \omega^j), B_1(t \omega^j), \ldots, B_k(t \omega^j)) = 0\)
  \item \(A_1|_{H_0} = z\), \(A_1(\omega^{-1}T) = 0\)
\end{itemize}
Target-on-Subdomain Testing

Consider the setting where the verifier has oracle access to a function \( f : L \to \mathbb{F} \) and wishes to check that \( \hat{f} \mid_H = \hat{z} \) for a given target function \( z : H \to \mathbb{F} \). (E.g. \( z \) is all 0's.)

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

Hence:

\[
P((\mathbb{F}, L, H, z), f) \quad \xrightarrow{f : L \to \mathbb{F}} \quad V((\mathbb{F}, L, H, z))
\]

\( h : L \to \mathbb{F} \)

- Test that \( h \) is \( d \)-close to \( RS[\mathbb{F}, L, d-1|H|] \)
- Sample \( y \in L \) and check \( f(y) - \hat{z}(y) = h(y) V_H(y) \)

Completeness: if \( \hat{f} \mid_H = \hat{z} \) then \( h := \hat{h} \mid_L \in RS[\mathbb{F}, L, d-1|H|] \) and passes check \( \forall y \in L \).

Soundness: if \( \hat{f} \mid_H \neq \hat{z} \) then \( \forall h : L \to \mathbb{F} \) we have two cases:

- \( h \) is \( d \)-far from \( RS[\mathbb{F}, L, d-1|H|] \) \( \rightarrow \) verifier accepts w.p. \( \epsilon_{ver}(s) \)
- \( h \) is \( \delta \)-close to \( \hat{h} \) of degree \( d-1|H| \) \( \rightarrow \hat{f}(x) - \hat{z}(x) \neq \hat{h}(x) V_H(x) \) so verifier accepts w.p. \( \frac{d}{|H|} + \delta \)

Time complexity of the verifier: [Ignore LDT because if using FRI \( t_{cor} = O(\log |H|) \), which is small]

- if \( \hat{z} \neq 0^H \) then: evaluate \( V_H \) at \( \hat{y} \) and evaluate \( \hat{z} \) at \( \hat{y} \) \( \rightarrow \text{poly}(1|H|) \)
- if \( \hat{z} = 0^H \) then: evaluate \( V_H \) at \( \hat{y} \) \( \rightarrow \text{poly}(1|H|) \) in general but \( \text{poly}(\log 1|H|) \) 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

\[ P((E, z, T), \mathbf{A}) \]

- Run computation on trace \( A_1, \ldots, A_k \), augment it with \( B_1, \ldots, B_k \).
- For each \( i \in [k] \):
  
  \[ f_i := \text{\( \hat{A}_i \)} |_L \in RS[\{F, L, 1H], H] \]

- For each \( j \in [k] \):
  
  \[ g_j := \text{\( \hat{B}_j \)} |_L \in RS[\{F, L, 1H], H] \]

- For each \( j \in [m] \):
  
  \( h_j := \text{\( \hat{h}_j \)} (x) |_L \in RS[\{F, L, 1H], H] \)

  \[ h_j(x) := \frac{P_j (\hat{A}_1(x), \ldots, \hat{A}_k(x), \hat{B}_1(x), \ldots, \hat{B}_k(x))}{V_{H}(x)/(x^{T-1})} \]

- \( h_x := \text{\( \hat{h}_x \)} (x) |_L \in RS[\{F, L, 1H], H] \)

  \[ h_x(x) := \frac{\hat{A}_x(x) - \hat{x}(x)}{V_{H}(x)} \]

- \( h_0 := \text{\( \hat{h}_0 \)} (x) |_L \in RS[\{F, L, 1H], H] \)

  \[ h_0(x) := \frac{\hat{A}_0(x)}{(x^{T-1})} \]

\[ V((E, z, T)) \]

\( \{f_i : L \rightarrow \mathbb{R}, i \in [k]\} \)

\( \{g_j : L \rightarrow \mathbb{R}, j \in [k]\} \)

\( \{h_j : L \rightarrow \mathbb{F}, j \in [m]\} \)

\( h_x, h_0 : L \rightarrow \mathbb{R} \)

- Test each of the received function for the appropriate degree
  
  \[ \text{we will come back to this} \]

- Sample \( Y \in L \) and check that:

  \( \forall j \in [m] \)

  \[ h_j(x) V_{H}(x) = P_j (f_1(x), \ldots, f_k(x), g_1(x), \ldots, g_k(x)) \]

  \( h_x(x) = f_1(x) - \hat{x}(x) \)

  \( h_0(x) (x^{T-1}) = f_1(x) \)
Completeness

Suppose that $A_1, ..., A_k : [T] \rightarrow F$ is a witness for $(E, z, T) \in BH$. The prover can evaluate $E$ at each time step to augment the trace with $B_1, ..., B_k : [T] \rightarrow F$ that satisfy all $m$ quadratic equations $p_1, ..., p_m$ derived from $E$. So the prover can find $\hat{h}_1(x), ..., \hat{h}_m(x)$.

- $A_i$ 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**: $O((k + L + m)|L|) = O((k + L + m)|H|) = O(1E1 \cdot T)$ elts
- **query complexity**: $O((k + L + m) \log |L|) = O(1E1 \log T)$
- **prover time**: $O((k + L + m)|L| \log |L|) = O(1E1T \log T)$
- **verifier time**: $O((k + L + m) \log |L| \text{poly}(n)) = O(1E1 \log T) + \text{poly}(n)$
Soundness

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

There are two cases:

1. One of the functions is far from RS.
   - \(\exists i \in \{\mathcal{E}\}: f_i \text{ is } \delta\text{-far from } RS[\mathcal{F}, L, 1H-1]\)
   - \(\exists i \in \{\mathcal{E}\}: g_i \text{ is } \delta\text{-far from } RS[\mathcal{F}, L, 1H-1]\)
   - \(\exists j \in \{\mathcal{M}\}: h_j \text{ is } \delta\text{-far from } RS[\mathcal{F}, L, 1H-1]\)
   - \(h_0 \text{ is } \delta\text{-far from } RS[\mathcal{F}, L, 1H-1]\)

\(\Rightarrow\) verifier accepts w.p. \(\leq \varepsilon \text{lost}(\delta)\)

2. All functions are close to (unique) polynomials \(\{\hat{f}_i \}_{i \in \{\mathcal{E}\}}, \{\hat{g}_i \}_{i \in \{\mathcal{E}\}}, \{\hat{h}_j \}_{j \in \{\mathcal{M}\}}, \hat{h}_0, \hat{h}_0\) of the appropriate degree.
   - \(\exists j \in \{\mathcal{M}\}: \hat{h}_j(x) V_{\mathcal{H}_0}(x) \neq \hat{f}_j(x) V_{\mathcal{H}_0}(x) \Rightarrow \) consistency test passes w.p. \(\leq \frac{2H-2+2(k+b)\delta}{1-H}\)
   - \(\hat{h}_0(x) V_{\mathcal{H}_0}(x) \neq \hat{f}_0(x) \Rightarrow \) consistency check accepts w.p. \(\leq \frac{H-1}{1-H} + 2\delta\)
   - \(\hat{h}_0(x)(x - w^{T-1}) \neq \hat{f}_0(x) \Rightarrow \) consistency check accepts w.p. \(\leq \frac{H-1}{1-H} + 2\delta\)

Several options to make this < 1 :

- Set proximity parameter \(\delta = O(\frac{1}{\log 2}) = O(1/\log 2)\)
  - this requires setting repetition parameter \(t\) in FRI to \(t = O(1/\log 2)\) to ensure that \(E(\delta) = O(1)\)
- Repeat consistency test \(t = O(\log 1/\delta)\) times, as the term becomes \(\frac{2H-2+2(k+b)\delta}{1-H}t\)

- Send random coefficients to prove \(\Delta\) test \(\Sigma_i \delta f_i + \Sigma_i \delta g_i\) instead of individually
  - distortion statements imply the error becomes \(\frac{2H-2+2\delta}{1-H} + 2\delta\) (due to column distance)
From Automata to Machines

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 fetch 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>2</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>r</td>
<td>2</td>
<td>21</td>
<td>10</td>
</tr>
<tr>
<td>r</td>
<td>2</td>
<td>31</td>
<td>10</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 \( \text{op} = \text{r} \rightarrow \text{val}' = \text{val} \)
- 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.

- $R(E, z, T)$ outputs quadratic equations $p_1, \ldots, p_m \in \mathbb{F}[X_{x_1}, \ldots, X_{x_{k+1}}]$ with $m, l = O(N^1)$
- $(E, z, T) \in \text{BH}$ iff $\exists$ augmented execution trace $A_1, \ldots, A_k, B_1, \ldots, B_e : H \rightarrow \mathbb{F}$

A permutation $\tau : [T] \rightarrow [T]$ such that

- $\forall t \in \{0, 1, \ldots, T-1\} : \forall j \in [m] \quad \exists \ p_j \left( A_i(\omega^t), \ldots, A_i(\omega^{t+1}), A_{i+1}(\omega^t), \ldots, A_{i+1}(\omega^{t+1}), B_1(\omega^t), \ldots, B_e(\omega^{t+1}) \right) = 0$

- $A_1|_{H_{in}} = z$, $A_1(\omega^{-1}) = 0$

Proof: Set $p_1, \ldots, p_m$ to be the quadratic equations obtained by translating the transition function $&$ also the logic for "what you write is what you read".

Completeness: choose $\tau$ to be the permutation that reorders the trace by address then time, so that the memory checks pass.

Soundness: for any choice of permutation $\tau$, 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 \( \{\hat{f}(a)\}_{a \in H} \) equal as multisets, which in turn is true if
\[
\prod_{a \in H} (x - \hat{g}(a)) \equiv \prod_{a \in H} (x - \hat{f}(a)).
\]

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

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

**Compute partial products:**
- \( f_{\pi} : L \rightarrow \mathbb{F} \text{ s.t. } \hat{f}_{\pi}(w) = \prod_{i \in \pi}(1 - \hat{f}(w^i)) \)
- \( g_{\pi} : L \rightarrow \mathbb{F} \text{ s.t. } \hat{g}_{\pi}(w) = \prod_{i \in \pi}(1 - \hat{g}(w^i)) \)

**Compute** \( h_1, h_2, h_3, h_4, h_5 : L \rightarrow \mathbb{F} \text{ s.t.} \)
\[
\begin{align*}
\hat{h}_1(x) &= \frac{f_{\pi}(x) - (r - f(x))f_{\pi}(w^x)}{v_H(x)/(x-1)} \\
\hat{h}_2(x) &= \frac{g_{\pi}(x) - (r - g(x))g_{\pi}(w^x)}{v_H(x)/(x-1)} \\
\hat{h}_3(x) &= \frac{f_{\pi}(x) - (r - f(x))}{(x-1)} \\
\hat{h}_4(x) &= \frac{g_{\pi}(x) - (r - g(x))}{(x-1)} \\
\hat{h}_5(x) &= \frac{f_{\pi}(x) - g_{\pi}(x)}{(x - w^{\pi^{-1}})}
\end{align*}
\]

**V((L, H))**

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

- **Sample** \( \pi \in \mathbb{F} \) and check:
  \[
  \begin{align*}
  h_1(x) &= \frac{f_{\pi}(x) - (r - f(x))f_{\pi}(w^x)}{v_H(x)/(x-1)} \\
  h_2(x) &= \frac{g_{\pi}(x) - (r - g(x))g_{\pi}(w^x)}{v_H(x)/(x-1)} \\
  h_3(x) &= \frac{f_{\pi}(x) - (r - f(x))}{(x-1)} \\
  h_4(x) &= \frac{g_{\pi}(x) - (r - g(x))}{(x-1)} \\
  h_5(x) &= \frac{f_{\pi}(x) - g_{\pi}(x)}{(x - w^{\pi^{-1}})}
  \end{align*}
  \]

- Test that all received functions are LD.