More Verifier Efficient Interactive Proofs For Bounded Space

Joshua Cook
Program To Run

- Deterministic Machine M in TISP[T, S]
  - Time T, Space S
- Think $S \ll n \ll T$
  - $S = n^\alpha$, $T = 2S^\beta$, for $\alpha, \beta > 1$

$O(\log(n) + \log(S))$
Arthur Doesn’t Have Time!

Arthur wants to run M. Doesn’t have exponential time in S!

Merlin can help, but untrusted. Has exponential time, but just $2^{O(S)}$. 
Interactive Proofs (IPs)

Untrusted Merlin
(Prover P)
Randomized Arthur
(Verifier V)

Many Questions and Answers.
Results
Interactive Time

$L \in \text{ITIME}[T_V, T_P]$

Verifier time $T_V$, Prover time $T_P$

Completeness: If $x \in L$,
then $P$ convinces $V$ with probability $\frac{2}{3}$

Soundness: If $x \not\in L$,
then NO $P'$ convinces $V$ with probability $\frac{1}{3}$
Main Result for TISP\([T, S]\)

US: \[ \text{ITIME}[\tilde{\Omega}(\log(T)S + n), 2^{O(S)}] \]

(Previous Best, time not explicit prior)

Sha: \[ \text{ITIME}[\tilde{\Omega}(\log(T)(S + n)), 2^{O(\log(T)(S + n))}] \]

GKR: \[ \text{ITIME}[\tilde{\Omega}(\log(T)S^2 + n), 2^{O(S)}] \]

RRR: \[ \text{ITIME}[T^{o(1)}S^2 + n, T^{1+o(1)}S^{O(1)}] \]
Us Vs Shamir

IP for SPACE[n^α] \quad T = 2^S
Verifier Time n^β

α vs β

Ours is better when S < n

Our prover is ALWAYS faster

2^O(S) vs 2^O(S^2)
IPs for Randomized Space

- Let $L \in \text{BPSPACE}[S]$
- Standard: Saks Zhou, $L \in \text{SPACE}[S^{3/2}]$:
  - Shamir, $L$ has time $S^3$ verifier
- Us, Use Nisan’s PRG with Our IP:
  - Reduction: space $S$, input length $S^2$
  - Our IP, $L$ has time $S^2$ verifier
  - Match’s deterministic IP
Nondeterministic Result

IP for NTISP[T, S]

US: \( \text{ITIME}[\tilde{O}(\log(T)^2S + n), 2^{O(S)}] \)

Sha: \( \text{ITIME}[\tilde{O}(\log(T)^2(S + n)), 2^{O(\log(T)(S + n))}] \)

GKR: \( \text{ITIME}[\tilde{O}(\log(T)S^2 + n), 2^{O(S)}] \)

RRR: NA
Us Vs GKR

IP for NTISP\([_2^n \alpha, n]\)
Verifier Time \(n^\beta\)

\(\alpha \ vs \ \beta\)

Ours is better
When \(T \ll 2^S\)
Deterministic Algorithm

Both Prover \(2^{O(S)}\)
Proof
Proof Outline

Us
- Space to Matrix
  - Simpler reduction
- Matrix Sum Check
  - Simpler
- Arithmetize Multitape
  - Allows $S < n$

Shamir
- Space to QBF
  - Needs conditioning
- QBF Sum Check
  - Requires Specific Format Reduction
- Arithmetize Single
Why Not Single Tape TM?

Single tape TM require $S > n$

Concern, need $\tilde{O}(n + S)$ time arithmetization

Show for multitape TM, paper uses RAM

RAM more efficient, only constant factor
Reduction To Matrix
Computation Graph

View space $S$ program as $2^S$ state graph, $G$

Edges are state transitions

Graph is a function of Input, Program

Accepts IFF there is a length $T$ path from start to end.

Edges are fast to compute
Adjacency Matrix

Represent G as an adjacency, \( M \)

Algorithm accepts in time \( T \) iff

\[
M^T_{\text{start, end}} = 1
\]

By repeated squaring,

\[
M^T = M^{2^t}
\]

For \( t = \log(T) \)

Run matrix sum check \( \log(T) \) times
Matrix Sum Check
Sum Check (LFKN)

Given: individual degree \( d \) polynomial, \( p: \mathbb{F}^S \rightarrow \mathbb{F} \), and \( \alpha \in \mathbb{F} \)

Reduce claim: \( \alpha = \sum_{a \in \{0,1\}^S} p(a) \)

To new claim: \( \Box = p(b) \)

some \( \Box \in \mathbb{F}, b \in \mathbb{F}^S \)
Sum Check Protocol

- Ask for $p_1(x) = \sum_{a \in \{0,1\}^{s-1}} p(x, a)$
- Check if $\alpha = p_1(0) + p_1(1)$
- Set $b_1$ randomly
- Ask for $p_2(x) = \sum_{a \in \{0,1\}^{s-2}} p(b_1, x, a)$
- Check if $p_1(b_1) = p_2(0) + p_2(1)$
- ...
Sum Check Idea (Schwartz-Zippel)

If $\alpha \neq \sum_{a \in \{0,1\}^s} p(a)$, then $p_1$ is incorrect.

$p_1$ is degree $d$, equal to true $p_1 \leq d$ places

$\Pr[\text{agree at } b_1] \leq d / |F|$
Sum Check Performance

There exists an IP with verifier V, prover P:

Completeness: If $\alpha = \sum_{a \in \{0,1\}^s} p(a)$, with P, V gives $\beta \in \mathbb{F}$ and $b \in \mathbb{F}^S$ s.t. $\beta = p(b)$

Soundness: If $\alpha \neq \sum_{a \in \{0,1\}^s} p(a)$, for any P', V gives $\beta = p(b)$ with probability $< Sd / |\mathbb{F}|$

Time: Verifier $Sd \tilde{O}(\log(|\mathbb{F}|))$ Prover $2^{O(S)} \tilde{O}(\log(|\mathbb{F}|))$
Matrix Multilinear Extension

For $2^S \times 2^S$ matrix $M$ containing elements of $\mathbb{F}$

Let $M : \mathbb{F}^S \times \mathbb{F}^S \rightarrow \mathbb{F}$ be s.t.

$M$ is multilinear (individual degree 1)

For any $a, c \in \{0,1\}^S$, $M(a, c) = M_{a,c}$
Matrix Sum Check (Thaler)

By definition
\[ M_{a,c}^2 = \sum_{b \in \{0,1\}^s} M_{a,b} M_{b,c} \]

Also have
\[ M^2(a,c) = \sum_{b \in \{0,1\}^s} M(a,b)M(b,c) \]

For claim \( \alpha = M^2(a,c) \), let \( p(b) = M(a,b)M(b,c) \)
Sum check reduces to \( \square = M(a,b)M(b,c) \)
Product Reduction

Reduce claim: $\square = p(a)p(b)$
To new claim: $\alpha' = p(c)$

• Let $\psi: \mathbb{F} \rightarrow \mathbb{F}^S$ be line s.t. $\psi(0) = a$, $\psi(1) = b$
  $\psi(x) = (1-x) a + x b$

• Ask for degree $S$ polynomial $q(x) \overset{\text{def}}{=} p(\psi(x))$

• Check if $\square = q(0)q(1)$

• For random $z$, set $\alpha' = q(z)$, $c = \psi(z)$
  $\alpha' = q(z) = p(\psi(z)) = p(c)$
Repeated Square Rooting

For start a, end b:
Verifier given claim $M^T_{a,b} = 1$, or $M^T(a,b) = 1$
Reduce to claim $M^{2^{t-1}}(a',b') = \alpha'$, $M^{2^{t-2}}(a'',b'') = \alpha''$...

After $\log(T)$ times, have claim $M(a^*,b^*) = \alpha^*$
Uses $S \log(T)$ operations over $\mathbb{F}$
Arithmetization
Calculate $M$, multilinear extension

From program definition, $M_{a,b}$ simple.

How to calculate $M$?

Sum over every edge in program, simple formula can calculate easily.
Two Tape TM

Program has two tapes, input and working, \( \Lambda \) program transitions

Input \( x \),
Initial state \( a = (p, i, h, w) \)
Final state \( b = (p', i', h', w') \)

\( p, p' \)  TM program states,
\( i, i' \) input heads
\( h, h' \) working space heads
\( w, w' \) working space contents
Transition Function

\[ \sum_{\lambda \in \Lambda} u(\lambda, p) v(\lambda, p') \text{Inp}(\lambda, x, i, i') \text{Wrk}(\lambda, h, h', w, w') \]

- \( u \) : \( \lambda \) is from state \( p \)
- \( v \) : \( \lambda \) is to state \( p' \)
- \( \text{Inp} \) : \( x \) at \( i \) has symbol in \( \lambda \), \( i' \) is \( i+1 \) or \( i-1 \) from \( \lambda \)
- \( \text{Wrk} \) : \( w \) at \( h \) from \( \lambda \), \( h' \) is \( h+1 \) or \( h-1 \) from \( \lambda \), \( w' \) at \( h \) from \( \lambda \), \( w' = w \) elsewhere

Use different symbols! Calculate extensions separately!
Closer Look: \( \text{Wrk}(\lambda, h, h', w, w') \)

\[
\sum_{i \in [S]} \text{eq}(i, h) \text{eq}(i+D(\lambda), h') \text{bef}(i, w, w') \text{aft}(i, w, w') \\
\text{eq}(\text{us}(\lambda), w_i) \text{eq}(\text{vs}(\lambda), w'_i)
\]

\( \text{eq} \) checks equality, \( D \) 1 for R, -1 for L

\( \text{bef} \) equality before i, \( \text{aft} \) equality after i

\( \text{us} \) starting symbol, \( \text{vs} \) ending symbol

Use different symbols! Calculate extensions separately!
Calculate \textbf{Wrk} Efficiently

- \textit{eq}(w_i, w'_i) = w_i w'_i + (1 - w'_i)(1 - w_i)
- \textit{bef}(i+1, w, w') = \textit{bef}(i, w, w') \textit{eq}(w_i, w'_i)
- \textit{bef}(i, w, w') can be calculated for each \textit{i} in \textit{O(S)} operations. \textit{aft} similarly
- Similarly, \textit{eq}(i, h) for each \textit{i} with \textit{O(S)} ops.
- Only \textit{O(S)} operations in \textit{Wrk}
Finishing up Arithmetization

- \textbf{Inp} similarly calculated in \(O(n)\) operations
- Total \(M\) only takes \(O(n + S)\) operations.
Prover Time

Entire $M$ can be constructed in time $\sim 2^{2S}$

Each $M^k$ for $k = 2^i$ in time $\sim \log(T)2^{\omega S}$

Any $M^k(a, b)$ calculated in time $\sim 2^{2S}$
Open Problems

- Remove log(T) factor from verifier time
- Do nondeterministic algorithms have same verifier time as deterministic?
- Same verifier time, poly(T) time prover?
- Gives quadratic gap interactive hierarchy
  - Fine grain interactive hierarchy?
Citations