Sequential Composability for Rational Proofs

Matteo Campanelli   Rosario Gennaro

The City College of New York
CUNY

The Model

Verifiable Computation against a *rational* rather than *malicious* adversary

Adversary is only interested in maximizing a well-defined *utility function*

Our Results

Starting from the concept of *Rational Proofs* (AM'12)

- Consider a new model where many computations are outsourced, and define a notion of *sequential composability* to assure that providing the correct result on *all* computations is the rational strategy.
- Show that the some of the known rational proofs do *not* satisfy our notion of sequential composability.
- Present a new rational proof protocol which (for certain functions) is squentially composable.

Our Results

Starting from the concept of *Rational Proofs* (AM'12)

- Consider a new model where many computations are outsourced, and define a notion of *sequential composability* to assure that providing the correct result on *all* computations is the rational strategy.
- Show that the some of the known rational proofs do *not* satisfy our notion of sequential composability.
- Present a new rational proof protocol which (for certain functions) is squentially composable.

Our Results

Starting from the concept of *Rational Proofs* (AM'12)

- Consider a new model where many computations are outsourced, and define a notion of *sequential composability* to assure that providing the correct result on *all* computations is the rational strategy.
- Show that the some of the known rational proofs do *not* satisfy our notion of sequential composability.
- Present a new rational proof protocol which (for certain functions) is squentially composable.

Rational Proofs

An interactive proof between P and V

- On input a function f and a value x
  1. P provides V with a value y
  2. V "pays" P with a randomized reward $R(transcript)$
- The reward is maximized (in expectation) when P provides the correct value $y = f(x)$

Rational Proofs

An interactive proof between P and V

- On input a function f and a value x
  1. P provides V with a value y
  2. V "pays" P with a randomized reward $R(transcript)$
- The reward is maximized (in expectation) when P provides the correct value $y = f(x)$

Rational Proofs

An interactive proof between P and V

- On input a function f and a value x
  1. P provides V with a value y
  2. V "pays" P with a randomized reward $R(transcript)$
- The reward is maximized (in expectation) when P provides the correct value $y = f(x)$

Rational Proofs

An interactive proof between P and V

- On input a function f and a value x
  1. P provides V with a value y
  2. V "pays" P with a randomized reward $R(transcript)$
- The reward is maximized (in expectation) when P provides the correct value $y = f(x)$

Simplicity

The most attractive feature of RP is their simplicity. AM'12 shows

- A one-round proof for **PP**
- A poly-round protocol for the counting hierarchy

The above assumes an all-powerful prover and a poly-time verifier.

Simplicity

The most attractive feature of RP is their simplicity. AM'12 shows

- A one-round proof for **PP**
- A poly-round protocol for the counting hierarchy

The above assumes an all-powerful prover and a poly-time verifier.

Simplicity

The most attractive feature of RP is their simplicity. AM'12 shows

- A one-round proof for **PP**
- A poly-round protocol for the counting hierarchy

The above assumes an all-powerful prover and a poly-time verifier.

Simplicity

The most attractive feature of RP is their simplicity. AM'12 shows

- A one-round proof for **PP**
- A poly-round protocol for the counting hierarchy

The above assumes an all-powerful prover and a poly-time verifier.

Efficient Rational Proofs

If C is the complexity of computing f, for Verifiable Computation we want a $\tilde{O}(C)$ Prover and a $o(C)$ Verifier.

For a $O(\log n)$ Verifier, AM'13 presents a constant-round protocol for uniform constant-depth threshold circuits

- Assumes log-search-uniformity for the circuit
- Possible to extend to a log-depth circuit if allow polylog-Verifiers [GHRV'14]

Efficient Rational Proofs

If C is the complexity of computing f, for Verifiable Computation we want a $\tilde{O}(C)$ Prover and a $o(C)$ Verifier.

For a $O(\log n)$ Verifier, AM'13 presents a constant-round protocol for uniform constant-depth threshold circuits

- Assumes log-search-uniformity for the circuit
- Possible to extend to a log-depth circuit if allow polylog-Verifiers [GHRV'14]

Efficient Rational Proofs

If C is the complexity of computing f, for Verifiable Computation we want a $\tilde{O}(C)$ Prover and a $o(C)$ Verifier.

For a $O(\log n)$ Verifier, AM'13 presents a constant-round protocol for uniform constant-depth threshold circuits

- Assumes log-search-uniformity for the circuit
- Possible to extend to a log-depth circuit if allow polylog-Verifiers [GHRV'14]

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p}, 1) + (1-p)BSR(\tilde{p}, 0) \tag{1}$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1 - \tilde{p})^2 + 1 = 2\tilde{p}(2 - \tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1 - \tilde{p}) - \tilde{p}^2 - (1 - \tilde{p})^2 + 1 = 2(1 - \tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p}, 1) + (1 - p)BSR(\tilde{p}, 0) \tag{1}$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p}, 1) + (1-p)BSR(\tilde{p}, 0) \tag{1}$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p}, 1) + (1-p)BSR(\tilde{p}, 0) \quad (1)$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p},1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p},0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p},1) + (1-p)BSR(\tilde{p},0) \tag{1}$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p}, 1) + (1-p)BSR(\tilde{p}, 0) \qquad (1)$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

The AM'13 threshold protocol

Consider a single threshold gate with n inputs, which evaluates to 1 if at least k input bits are 1

- P announces the number $\tilde{m}$ of input bits equal to 1;
  - Let $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1;
- V sets the output to 1 if $\tilde{m} \geq k$, to 0 otherwise;
- V selects a random index $i \in [1..n]$ and looks at input bit $b = x_i$;
- V pays P Brier's Rule $BSR(\tilde{p}, b)$ defined as

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Proof: Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward for P is

$$pBSR(\tilde{p}, 1) + (1-p)BSR(\tilde{p}, 0) \tag{1}$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.
  - If $\Delta = 1/poly$, budget remains polynomial

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.
  - If $\Delta = 1/poly$, budget remains polynomial

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.
  - If $\Delta = 1/poly$, budget remains polynomial

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.
  - If $\Delta = 1/poly$, budget remains polynomial

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.
  - If $\Delta = 1/poly$, budget remains polynomial

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Reward vs. Profit

If C is the cost of computing the function then the honest prover earns a profit $R - C$. Is this profit always maximized?

- Consider a lazy prover $\tilde{P}$ which invests very little effort $\tilde{C}$, and yet it receives a reward $\tilde{R}$.
- We want $R - C \geq \tilde{R} - \tilde{C}$
  - sufficient that $R - \tilde{R} \geq C$
- Consider the *reward gap* [AM'13,GHRV'14] $\Delta = min_{\tilde{P}}[R - \tilde{R}]$
- Scale the reward by a factor C/Δ.
  - If $\Delta = 1/poly$, budget remains polynomial

In the previous protocol $0 \leq R \leq 2$, $C = n$ and $\Delta = n^{-2}$, which means we need to scale the reward by a factor of n^3.

Example of Lazy Prover

If a bad prover answers at random (a O(1)-cost strategy), how much does it earn?

$$
\begin{aligned}
\tilde{R} &= E_{m,b}[BSR(\frac{m}{n}, b)] \\
&= \frac{1}{n+1}\sum_{m=0}^{n} E_b[BSR(\frac{m}{n}, b] \\
&= \frac{1}{n+1}\sum_{m=0}^{n} (2(2p \cdot \frac{m}{n} - \frac{m^2}{n^2} - p + 1)) \\
&= 2 - \frac{2n+1}{3n} > 1
\end{aligned}
$$

Note that the honest prover earns always less than 2.

Example of Lazy Prover

If a bad prover answers at random (a O(1)-cost strategy), how much does it earn?

$$
\begin{aligned}
\tilde{R} &= E_{m,b}[BSR(\frac{m}{n}, b)] \\
&= \frac{1}{n+1} \sum_{m=0}^{n} E_b[BSR(\frac{m}{n}, b] \\
&= \frac{1}{n+1} \sum_{m=0}^{n} (2(2p \cdot \frac{m}{n} - \frac{m^2}{n^2} - p + 1)) \\
&= 2 - \frac{2n+1}{3n} > 1
\end{aligned}
$$

Note that the honest prover earns always less than 2.

Example of Lazy Prover

If a bad prover answers at random (a O(1)-cost strategy), how much does it earn?

$$
\begin{aligned}
\tilde{R} &= E_{m,b}[BSR(\frac{m}{n}, b)] \\
&= \frac{1}{n+1} \sum_{m=0}^{n} E_b[BSR(\frac{m}{n}, b] \\
&= \frac{1}{n+1} \sum_{m=0}^{n} (2(2p \cdot \frac{m}{n} - \frac{m^2}{n^2} - p + 1)) \\
&= 2 - \frac{2n+1}{3n} > 1
\end{aligned}
$$

Note that the honest prover earns always less than 2.

Many Outsourced Problems

What if there is a large number of computations to be outsourced and provers compete against each other to solve them (e.g. volunteer computations).

The honest prover pays $O(n)$ and earns ≤ 2. The random prover pays $O(1)$ and earns > 1.

In the time that it takes the honest prover to solve one problem, the random prover can solve many and collect more money.

In this scenario, a fast incorrect answer is the rational strategy since it allows the prover to solve more problems and collect more rewards.

Many Outsourced Problems

What if there is a large number of computations to be outsourced and provers compete against each other to solve them (e.g. volunteer computations).

The honest prover pays $O(n)$ and earns ≤ 2. The random prover pays $O(1)$ and earns > 1.

In the time that it takes the honest prover to solve one problem, the random prover can solve many and collect more money.

In this scenario, a fast incorrect answer is the rational strategy since it allows the prover to solve more problems and collect more rewards.

Many Outsourced Problems

What if there is a large number of computations to be outsourced and provers compete against each other to solve them (e.g. volunteer computations).

The honest prover pays $O(n)$ and earns ≤ 2. The random prover pays $O(1)$ and earns > 1.

In the time that it takes the honest prover to solve one problem, the random prover can solve many and collect more money.

In this scenario, a fast incorrect answer is the rational strategy since it allows the prover to solve more problems and collect more rewards.

Many Outsourced Problems

What if there is a large number of computations to be outsourced and provers compete against each other to solve them (e.g. volunteer computations).

The honest prover pays $O(n)$ and earns ≤ 2. The random prover pays $O(1)$ and earns > 1.

In the time that it takes the honest prover to solve one problem, the random prover can solve many and collect more money.

In this scenario, a fast incorrect answer is the rational strategy since it allows the prover to solve more problems and collect more rewards.

Sequential Composability – First Attempt

We want that the reward of the honest prover P must always be larger than the total reward of any prover $\tilde{P}$ that invests less computation cost than P.

A rational proof (P, V) for a function f is sequentially composable if for every prover $\tilde{P}$, and every sequence of inputs $x, x_1, \ldots, x_k$ such that $C(x) \geq \sum_{i=1}^{k} \tilde{C}(x_i)$ we have that $R(x) \geq \sum_i \tilde{R}(x_i)$

Actually that's not possible if we ask for *every input*: a prover may be answering correctly without doing any work.

Sequential Composability – First Attempt

We want that the reward of the honest prover P must always be larger than the total reward of any prover $\tilde{P}$ that invests less computation cost than P.

A rational proof (P, V) for a function f is sequentially composable if for every prover $\tilde{P}$, and every sequence of inputs $x, x_1, \ldots, x_k$ such that $C(x) \geq \sum_{i=1}^{k} \tilde{C}(x_i)$ we have that $R(x) \geq \sum_i \tilde{R}(x_i)$

Actually that's not possible if we ask for *every input*: a prover may be answering correctly without doing any work.

Sequential Composability – First Attempt

We want that the reward of the honest prover P must always be larger than the total reward of any prover $\tilde{P}$ that invests less computation cost than P.

A rational proof (P, V) for a function f is sequentially composable if for every prover $\tilde{P}$, and every sequence of inputs $x, x_1, \ldots, x_k$ such that $C(x) \geq \sum_{i=1}^{k} \tilde{C}(x_i)$ we have that $R(x) \geq \sum_i \tilde{R}(x_i)$

Actually that's not possible if we ask for *every input*: a prover may be answering correctly without doing any work.

Sequential Composability – Second Attempt

We want that the reward of the honest prover P must always be larger than the total reward of any prover $\tilde{P}$ that invests less computation cost than P.

A rational proof (P, V) for a function f is sequentially composable for an input distribution $\mathcal{D}$ if for every prover $\tilde{P}$, and every sequence of inputs $x, x_1, \ldots, x_k \in \mathcal{D}$ such that $C(x) \geq \sum_{i=1}^{k} \tilde{C}(x_i)$ we have that $R(x) \geq \sum_i \tilde{R}(x_i)$

Sequential Composability – Second Attempt

We want that the reward of the honest prover P must always be larger than the total reward of any prover $\tilde{P}$ that invests less computation cost than P.

A rational proof (P, V) for a function f is sequentially composable for an input distribution $\mathcal{D}$ if for every prover $\tilde{P}$, and every sequence of inputs $x, x_1, \ldots, x_k \in \mathcal{D}$ such that $C(x) \geq \sum_{i=1}^{k} \tilde{C}(x_i)$ we have that $R(x) \geq \sum_i \tilde{R}(x_i)$

Two sufficient conditions

If $R(x) = R$ and $C(x) \leq C$ for the honest prover P, it is sufficient that

$$\frac{\tilde{R}}{R} \leq \frac{\tilde{C}}{C}$$

- $\sum_i \tilde{R}(x_i) \leq \frac{R}{C} \sum_{i=1} \tilde{C}(x_i) \leq R$
- If the reward is either R or 0 then let $\tilde{p}$ be the probability that $\tilde{P}$ receives the full reward R. Then it is sufficient that

$$\tilde{p} \leq \frac{\tilde{C}}{C}$$

  - Immediate from above since $\tilde{R} = \tilde{p} \cdot R$.

Two sufficient conditions

If $R(x) = R$ and $C(x) \leq C$ for the honest prover P, it is sufficient that

$$\frac{\tilde{R}}{R} \leq \frac{\tilde{C}}{C}$$

- $\sum_i \tilde{R}(x_i) \leq \frac{R}{C} \sum_{i=1} \tilde{C}(x_i) \leq R$
- If the reward is either R or 0 then let $\tilde{p}$ be the probability that $\tilde{P}$ receives the full reward R. Then it is sufficient that

$$\tilde{p} \leq \frac{\tilde{C}}{C}$$

  - Immediate from above since $\tilde{R} = \tilde{p} \cdot R$.

Two sufficient conditions

If $R(x) = R$ and $C(x) \leq C$ for the honest prover P, it is sufficient that

$$\frac{\tilde{R}}{R} \leq \frac{\tilde{C}}{C}$$

- $\sum_i \tilde{R}(x_i) \leq \frac{R}{C} \sum_{i=1} \tilde{C}(x_i) \leq R$
- If the reward is either R or 0 then let $\tilde{p}$ be the probability that $\tilde{P}$ receives the full reward R. Then it is sufficient that

$$\tilde{p} \leq \frac{\tilde{C}}{C}$$

  - Immediate from above since $\tilde{R} = \tilde{p} \cdot R$.

A solution in the PCP model

This protocol appears in [AM'13] as a "stand-alone" RP.

- Let $\mathcal{C}$ be a circuit computing f of size S. On input x, The Prover writes down the values of all the wires of $\mathcal{C}$ when evaluated at x.
- The verifier chooses one gate at random and verifies that it has been computed correctly. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct gates written down by $\tilde{P}$. Therefore $\tilde{p} = \tilde{m}/S$.
- In a cost model in which the prover pays 1 to compute *and* write down a gate then $\tilde{p} \leq \tilde{C}/C$ as desired.

Notice that in other cost models sequential composition may not follow. For example, a prover that pays separately \$1 to compute the value of a single gate and \$1 to write that value down. For dishonest provers that write the whole string down

$$\frac{\tilde{C}}{C} = \frac{1}{2} + \frac{\tilde{s}}{S}$$

A solution in the PCP model

This protocol appears in [AM'13] as a "stand-alone" RP.

- Let $\mathcal{C}$ be a circuit computing f of size S. On input x, The Prover writes down the values of all the wires of $\mathcal{C}$ when evaluated at x.
- The verifier chooses one gate at random and verifies that it has been computed correctly. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct gates written down by $\tilde{P}$. Therefore $\tilde{p} = \tilde{m}/S$.
- In a cost model in which the prover pays 1 to compute *and* write down a gate then $\tilde{p} \leq \tilde{C}/C$ as desired.

Notice that in other cost models sequential composition may not follow. For example, a prover that pays separately \$1 to compute the value of a single gate and \$1 to write that value down. For dishonest provers that write the whole string down

$$\frac{\tilde{C}}{C} = \frac{1}{2} + \frac{\tilde{s}}{S}$$

A solution in the PCP model

This protocol appears in [AM'13] as a "stand-alone" RP.

- Let $\mathcal{C}$ be a circuit computing f of size S. On input x, The Prover writes down the values of all the wires of $\mathcal{C}$ when evaluated at x.
- The verifier chooses one gate at random and verifies that it has been computed correctly. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct gates written down by $\tilde{P}$. Therefore $\tilde{p} = \tilde{m}/S$.
- In a cost model in which the prover pays 1 to compute *and* write down a gate then $\tilde{p} \leq \tilde{C}/C$ as desired.

Notice that in other cost models sequential composition may not follow. For example, a prover that pays separately $1 to compute the value of a single gate and $1 to write that value down. For dishonest provers that write the whole string down

$$\frac{\tilde{C}}{C} = \frac{1}{2} + \frac{\tilde{s}}{S}$$

A solution in the PCP model

This protocol appears in [AM'13] as a "stand-alone" RP.

- Let $\mathcal{C}$ be a circuit computing f of size S. On input x, The Prover writes down the values of all the wires of $\mathcal{C}$ when evaluated at x.
- The verifier chooses one gate at random and verifies that it has been computed correctly. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct gates written down by $\tilde{P}$. Therefore $\tilde{p} = \tilde{m}/S$.
- In a cost model in which the prover pays 1 to compute *and* write down a gate then $\tilde{p} \leq \tilde{C}/C$ as desired.

Notice that in other cost models sequential composition may not follow. For example, a prover that pays separately $\$1$ to compute the value of a single gate and $\$1$ to write that value down. For dishonest provers that write the whole string down

$$\frac{\tilde{C}}{C} = \frac{1}{2} + \frac{\tilde{s}}{S}$$

A solution in the PCP model

This protocol appears in [AM'13] as a "stand-alone" RP.

- Let $\mathcal{C}$ be a circuit computing f of size S. On input x, The Prover writes down the values of all the wires of $\mathcal{C}$ when evaluated at x.
- The verifier chooses one gate at random and verifies that it has been computed correctly. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct gates written down by $\tilde{P}$. Therefore $\tilde{p} = \tilde{m}/S$.
- In a cost model in which the prover pays 1 to compute *and* write down a gate then $\tilde{p} \leq \tilde{C}/C$ as desired.

Notice that in other cost models sequential composition may not follow. For example, a prover that pays separately $\$1$ to compute the value of a single gate and $\$1$ to write that value down. For dishonest provers that write the whole string down

$$\frac{\tilde{C}}{C} = \frac{1}{2} + \frac{\tilde{s}}{S}$$

A solution in the PCP model

This protocol appears in [AM'13] as a "stand-alone" RP.

- Let $\mathcal{C}$ be a circuit computing f of size S. On input x, The Prover writes down the values of all the wires of $\mathcal{C}$ when evaluated at x.
- The verifier chooses one gate at random and verifies that it has been computed correctly. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct gates written down by $\tilde{P}$. Therefore $\tilde{p} = \tilde{m}/S$.
- In a cost model in which the prover pays 1 to compute *and* write down a gate then $\tilde{p} \leq \tilde{C}/C$ as desired.

Notice that in other cost models sequential composition may not follow. For example, a prover that pays separately $\$1$ to compute the value of a single gate and $\$1$ to write that value down. For dishonest provers that write the whole string down

$$\frac{\tilde{C}}{C} = \frac{1}{2} + \frac{\tilde{s}}{S}$$

A solution for large number of problems

Following [BCEJKL08] (which considered this problem though with different definitions)

- Let c be the cost incurred to compute f by the honest prover. Assume that for a randomly chosen input $x \in D$ a prover $\tilde{P}$ that invests less than c cost, can guess $f(x)$ only with negligible probability.
- Batch computations in set of k, i.e. pay the prover after he solves k computations.
- The verifier chooses one computation at random and checks by re-executing it. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct results returned by $\tilde{P}$. Therefore

$$\tilde{p} = \frac{\tilde{m}}{k} = \frac{\tilde{m}c}{kc} \leq \frac{\tilde{C}}{C}$$

This requires k to be large enough to keep the Verifier efficient (so the cost of computing f once is amortized over k executions)

A solution for large number of problems

Following [BCEJKL08] (which considered this problem though with different definitions)

- Let c be the cost incurred to compute f by the honest prover. Assume that for a randomly chosen input $x \in D$ a prover $\tilde{P}$ that invests less than c cost, can guess $f(x)$ only with negligible probability.
- Batch computations in set of k, i.e. pay the prover after he solves k computations.
- The verifier chooses one computation at random and checks by re-executing it. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct results returned by $\tilde{P}$. Therefore

$$\tilde{p} = \frac{\tilde{m}}{k} = \frac{\tilde{m}c}{kc} \leq \frac{\tilde{C}}{C}$$

This requires k to be large enough to keep the Verifier efficient (so the cost of computing f once is amortized over k executions)

A solution for large number of problems

Following [BCEJKL08] (which considered this problem though with different definitions)

- Let c be the cost incurred to compute f by the honest prover. Assume that for a randomly chosen input $x \in D$ a prover $\tilde{P}$ that invests less than c cost, can guess $f(x)$ only with negligible probability.
- Batch computations in set of k, i.e. pay the prover after he solves k computations.
- The verifier chooses one computation at random and checks by re-executing it. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct results returned by $\tilde{P}$. Therefore

$$\tilde{p} = \frac{\tilde{m}}{k} = \frac{\tilde{m}c}{kc} \leq \frac{\tilde{C}}{C}$$

This requires k to be large enough to keep the Verifier efficient (so the cost of computing f once is amortized over k executions)

A solution for large number of problems

Following [BCEJKL08] (which considered this problem though with different definitions)

- Let c be the cost incurred to compute f by the honest prover. Assume that for a randomly chosen input $x \in D$ a prover $\tilde{P}$ that invests less than c cost, can guess $f(x)$ only with negligible probability.
- Batch computations in set of k, i.e. pay the prover after he solves k computations.
- The verifier chooses one computation at random and checks by re-executing it. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct results returned by $\tilde{P}$. Therefore

$$\tilde{p} = \frac{\tilde{m}}{k} = \frac{\tilde{m}c}{kc} \leq \frac{\tilde{C}}{C}$$

This requires k to be large enough to keep the Verifier efficient (so the cost of computing f once is amortized over k executions)

A solution for large number of problems

Following [BCEJKL08] (which considered this problem though with different definitions)

- Let c be the cost incurred to compute f by the honest prover. Assume that for a randomly chosen input $x \in D$ a prover $\tilde{P}$ that invests less than c cost, can guess $f(x)$ only with negligible probability.
- Batch computations in set of k, i.e. pay the prover after he solves k computations.
- The verifier chooses one computation at random and checks by re-executing it. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct results returned by $\tilde{P}$. Therefore

$$\tilde{p} = \frac{\tilde{m}}{k} = \frac{\tilde{m}c}{kc} \leq \frac{\tilde{C}}{C}$$

This requires k to be large enough to keep the Verifier efficient (so the cost of computing f once is amortized over k executions)

A solution for large number of problems

Following [BCEJKL08] (which considered this problem though with different definitions)

- Let c be the cost incurred to compute f by the honest prover. Assume that for a randomly chosen input $x \in D$ a prover $\tilde{P}$ that invests less than c cost, can guess $f(x)$ only with negligible probability.
- Batch computations in set of k, i.e. pay the prover after he solves k computations.
- The verifier chooses one computation at random and checks by re-executing it. If the result is correct, she pays R, otherwise she pays 0.
- Let $\tilde{m}$ be the number of correct results returned by $\tilde{P}$. Therefore

$$\tilde{p} = \frac{\tilde{m}}{k} = \frac{\tilde{m}c}{kc} \leq \frac{\tilde{C}}{C}$$

This requires k to be large enough to keep the Verifier efficient (so the cost of computing f once is amortized over k executions)

A solution for certain bounded depth circuits

Let $\mathcal{C}$ be a (uniform) arithmetic circuit of depth d of bounded fan-in (say 2).

- Starting from the output gate G, the Prover sends the values out, in_0, in_1
- The verifier checks that $G(in_0, in_1) = out$. If the check fails, it pays 0 and stop.
- If in_0 and in_1 are input wires, the Verifier checks that they are the correct input values and stops. If the check fails it pays 0, otherwise it pays R.
- The verifier chooses a random bit b and the protocol is recursively called on the subcircuit that has output in_b.

A solution for certain bounded depth circuits

Let $\mathcal{C}$ be a (uniform) arithmetic circuit of depth d of bounded fan-in (say 2).

- Starting from the output gate G, the Prover sends the values out, in_0, in_1
- The verifier checks that $G(in_0, in_1) = out$. If the check fails, it pays 0 and stop.
- If in_0 and in_1 are input wires, the Verifier checks that they are the correct input values and stops. If the check fails it pays 0, otherwise it pays R.
- The verifier chooses a random bit b and the protocol is recursively called on the subcircuit that has output in_b.

A solution for certain bounded depth circuits

Let $\mathcal{C}$ be a (uniform) arithmetic circuit of depth d of bounded fan-in (say 2).

- Starting from the output gate G, the Prover sends the values out, in_0, in_1
- The verifier checks that $G(in_0, in_1) = out$. If the check fails, it pays 0 and stop.
- If in_0 and in_1 are input wires, the Verifier checks that they are the correct input values and stops. If the check fails it pays 0, otherwise it pays R.
- The verifier chooses a random bit b and the protocol is recursively called on the subcircuit that has output in_b.

A solution for certain bounded depth circuits

Let $\mathcal{C}$ be a (uniform) arithmetic circuit of depth d of bounded fan-in (say 2).

- Starting from the output gate G, the Prover sends the values out, in_0, in_1
- The verifier checks that $G(in_0, in_1) = out$. If the check fails, it pays 0 and stop.
- If in_0 and in_1 are input wires, the Verifier checks that they are the correct input values and stops. If the check fails it pays 0, otherwise it pays R.
- The verifier chooses a random bit b and the protocol is recursively called on the subcircuit that has output in_b.

Stand-Alone Analysis

The protocol is a RP in the "stand-alone" sense for log-depth circuits.

- the probability of $\tilde{P}$ to obtain R when giving an incorrect result is $1 - 2^{-d}$
- $\tilde{P}$ can always compute one sub-circuit correctly and hope that's selected.
- At each level $\tilde{P}$ survives with probability $1/2$ so it is detected only with probability 2^{-d}.

Stand-Alone Analysis

The protocol is a RP in the "stand-alone" sense for log-depth circuits.

- the probability of $\tilde{P}$ to obtain R when giving an incorrect result is $1 - 2^{-d}$
- $\tilde{P}$ can always compute one sub-circuit correctly and hope that's selected.
- At each level $\tilde{P}$ survives with probability $1/2$ so it is detected only with probability 2^{-d}.

Stand-Alone Analysis

The protocol is a RP in the "stand-alone" sense for log-depth circuits.

- the probability of $\tilde{P}$ to obtain R when giving an incorrect result is $1 - 2^{-d}$
- $\tilde{P}$ can always compute one sub-circuit correctly and hope that's selected.
- At each level $\tilde{P}$ survives with probability $1/2$ so it is detected only with probability 2^{-d}.

Sequential Composability Analysis

Assume again that $\tilde{P}$ can output the right value of a wire only by computing the associated gate.

Consider a *regular* circuit: every subcircuit at a given level has the same "weight" (number of input-output paths entering it). Then for these circuits, the probability of success for $\tilde{P}$ investing $\tilde{C}$ is $\tilde{p} = 1 - 2^{-\tilde{d}}$, where $\tilde{d}$ is the height reached by "filling" in $\tilde{C}$ gates starting from the input level.

Therefore the protocol is sequentially composable for regular circuits where at each level the number of gates at least doubles.

- Example: the circuit that computes one FFT coefficient

Things can improve somewhat by iterating the protocol r times and reducing the probability of error to $(1-2^{-d})^r$. Note that the complexity of the Verifier will be $O(rd)$.

Sequential Composability Analysis

Assume again that $\tilde{P}$ can output the right value of a wire only by computing the associated gate.

Consider a *regular* circuit: every subcircuit at a given level has the same "weight" (number of input-output paths entering it). Then for these circuits, the probability of success for $\tilde{P}$ investing $\tilde{C}$ is $\tilde{p} = 1 - 2^{-\tilde{d}}$, where $\tilde{d}$ is the height reached by "filling" in $\tilde{C}$ gates starting from the input level.

Therefore the protocol is sequentially composable for regular circuits where at each level the number of gates at least doubles.

- Example: the circuit that computes one FFT coefficient

Things can improve somewhat by iterating the protocol r times and reducing the probability of error to $(1 - 2^{-d})^r$. Note that the complexity of the Verifier will be $O(rd)$.

Sequential Composability Analysis

Assume again that $\tilde{P}$ can output the right value of a wire only by computing the associated gate.

Consider a *regular* circuit: every subcircuit at a given level has the same "weight" (number of input-output paths entering it). Then for these circuits, the probability of success for $\tilde{P}$ investing $\tilde{C}$ is $\tilde{p} = 1 - 2^{-\tilde{d}}$, where $\tilde{d}$ is the height reached by "filling" in $\tilde{C}$ gates starting from the input level.

Therefore the protocol is sequentially composable for regular circuits where at each level the number of gates at least doubles.

- Example: the circuit that computes one FFT coefficient

Things can improve somewhat by iterating the protocol r times and reducing the probability of error to $(1 - 2^{-d})^r$. Note that the complexity of the Verifier will be $O(rd)$.

Sequential Composability Analysis

Assume again that $\tilde{P}$ can output the right value of a wire only by computing the associated gate.

Consider a *regular* circuit: every subcircuit at a given level has the same "weight" (number of input-output paths entering it). Then for these circuits, the probability of success for $\tilde{P}$ investing $\tilde{C}$ is $\tilde{p} = 1 - 2^{-\tilde{d}}$, where $\tilde{d}$ is the height reached by "filling" in $\tilde{C}$ gates starting from the input level.

Therefore the protocol is sequentially composable for regular circuits where at each level the number of gates at least doubles.

- Example: the circuit that computes one FFT coefficient

Things can improve somewhat by iterating the protocol r times and reducing the probability of error to $(1 - 2^{-d})^r$. Note that the complexity of the Verifier will be $O(rd)$.

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2} \log n$ then $\tilde{d} = c \log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n \log n)$.

If $\tilde{C} \geq \frac{n}{2} \log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2} \log n$ then $\tilde{d} = c \log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n \log n)$.

If $\tilde{C} \geq \frac{n}{2} \log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2} \log n$ then $\tilde{d} = c \log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n \log n)$.

If $\tilde{C} \geq \frac{n}{2} \log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

- We can use the "check by re-execution" strategy every $O(\log n)$ executions.

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2}\log n$ then $\tilde{d} = c\log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n\log n)$.

If $\tilde{C} \geq \frac{n}{2}\log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

- We can use the "check by re-execution" strategy every $O(\log n)$ executions;
- The Verifier's complexity is $O(n^c \log^2 n + n\log n) = o(n\log^2 n)$

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2} \log n$ then $\tilde{d} = c \log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n \log n)$.

If $\tilde{C} \geq \frac{n}{2} \log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

- We can use the "check by re-execution" strategy every $O(\log n)$ executions;
- The Verifier's complexity is $O(n^c \log^2 n + n \log n) = o(n \log^2 n)$.

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2}\log n$ then $\tilde{d} = c\log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n\log n)$.

If $\tilde{C} \geq \frac{n}{2}\log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

- We can use the "check by re-execution" strategy every $O(\log n)$ executions;
- The Verifier's complexity is $O(n^c \log^2 n + n\log n) = o(n\log^2 n)$.

A Mixed Strategy

Consider the circuit that given the point representation of a degree $n-1$ polynomial outputs the value of the polynomial at an additional point.

- An FFT circuit ($\log n$ levels of $n/2$ gates each) to go from point to coefficient representation
- An evaluation circuit ($\log n$ levels, with a total of $O(n)$ gates)

Note that if $\tilde{C} < \frac{n}{2}\log n$ then $\tilde{d} = c\log n$ with $c \leq 1$, and $\frac{\tilde{C}}{C} = O(1)$. Therefore with $O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$.

- The Verifier's complexity is $O(n^c \log n) = o(n\log n)$.

If $\tilde{C} \geq \frac{n}{2}\log n$ then it will take more than $O(\log n)$ executions for $\tilde{P}$ to earn more than P.

- We can use the "check by re-execution" strategy every $O(\log n)$ executions;
- The Verifier's complexity is $O(n^c \log^2 n + n\log n) = o(n\log^2 n)$.

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

1. A rational proof for any poly-time computable function (even stand-alone).

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

1. A rational proof for any poly-time computable function (even stand-alone).
2. A better sequentially composable protocol [illegible] bounded-depth circuit.
3. Other examples of "interesting" problems that have [illegible] be used with our protocol.

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

1. A rational proof for any poly-time computable function (even stand-alone).
2. A better sequentially composable protocol that works for *any* bounded-depth circuit.
3. Other examples of "interesting" problems that have circuits that can be used with our protocol.

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

1. A rational proof for any poly-time computable function (even stand-alone).
2. A better sequentially composable protocol that works for *any* bounded-depth circuit.
3. Other examples of "interesting" problems that have circuits that can be used with our protocol.

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

1. A rational proof for any poly-time computable function (even stand-alone).
2. A better sequentially composable protocol that works for *any* bounded-depth circuit.
3. Other examples of "interesting" problems that have circuits that can be used with our protocol.

Summary & Open Problems

Summary

1. We defined a notion of sequential composability for rational proofs, motivated by scenarios where many problems are outsourced by the Verifier to the Prover.
2. Goal: to make sure that *always* answering correcly is the rational strategy.
3. Proved that some RP in the protocols do not satisfy this definition, while others do. Presented a new protocol that achieves it for certain bounded-depth circuits.

Open Problems

1. A rational proof for any poly-time computable function (even stand-alone).
2. A better sequentially composable protocol that works for *any* bounded-depth circuit.
3. Other examples of "interesting" problems that have circuits that can be used with our protocol.