Optimal proof systems and acceptors: Distributional proving problems, and beyond

Edward A. Hirsch

http://logic.pdmi.ras.ru/~hirsch

(joint with D. Itsykson, I. Monakhov, V. Nikolaenko, A. Smal)

Steklov Institute of Mathematics at St.Petersburg, RAS

St.Petersburg Academic University, RAS

Optimal proof systems

- A proof system Σ simulates a proof system Ω iff Σ-proofs are at most as long as Ω-proofs (up to a polynomial p):

$$\forall F \in L \quad |\text{shortest } \Sigma\text{-proof of } F| \leq p(|\text{shortest } \Omega\text{-proof of } F|, |F|).$$

- p-simulation is a constructive version: For any w-size Ω-proof, one can compute a $p(w)$-size Σ-proof in polynomial time.
- (p-)optimal proof system (p-)simulates any other proof system.
- **Does it exist?..**

Optimal proof systems

- A proof system Σ simulates a proof system Ω iff Σ-proofs are at most as long as Ω-proofs (up to a polynomial p):

$$\forall F \in L \quad |\text{shortest } \Sigma\text{-proof of } F| \leq p(|\text{shortest } \Omega\text{-proof of } F|, |F|).$$

- p-simulation is a constructive version: For any w-size Ω-proof, one can compute a $p(w)$-size Σ-proof in polynomial time.
- (p-)optimal proof system (p-)simulates any other proof system.
- **Does it exist?..**

Theorem

$\exists$ ***p-optimal proof system*** $\iff$ $\exists$ ***optimal acceptor.***

For **TAUT**: [Krajícek, Pudlák].

For paddable languages: [Messner].

For **co-NP**-complete languages: [Chen, Flüm, Müller].

Simulations

- pointwise simulation $\mathcal{A} \prec \mathcal{B}$:
  $\exists$ polynomial p $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(t_{\mathcal{B}}(x) + |x|)$$

Simulations

- pointwise simulation $\mathcal{A} \prec \mathcal{B}$:
  $\exists$ polynomial p $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(t_{\mathcal{B}}(x) + |x|)$$

- (yet weaker!) worst-case simulation $\mathcal{A} \prec_{wc} \mathcal{B}$:
  $\exists$ polynomials p, q $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(\max_{\substack{|x'| \leq q(|x|) \\ x' \in L}} t_{\mathcal{B}}(x') + |x|)$$

Simulations

- pointwise simulation $\mathcal{A} \prec \mathcal{B}$:
  $\exists$ polynomial p $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(t_{\mathcal{B}}(x) + |x|)$$
- (weaker) average-case simulation $\mathcal{A} \prec_D \mathcal{B}$ w.r.t. D:
  $\forall \epsilon > 0$ $\exists c > 0$
  $$\mathop{\mathbf{E}}_{x \leftarrow D_n}[{t_{\mathcal{A}}}^c(x)] = O(n \mathop{\mathbf{E}}_{y \leftarrow D_n}[{t_{\mathcal{B}}}^\epsilon(y)])$$
- (yet weaker!) worst-case simulation $\mathcal{A} \prec_{wc} \mathcal{B}$:
  $\exists$ polynomials p, q $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(\max_{\substack{|x'| \leq q(|x|) \\ x' \in L}} t_{\mathcal{B}}(x') + |x|)$$

Simulations

- pointwise simulation $\mathcal{A} \prec \mathcal{B}$:
  $\exists$ polynomial p $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(t_{\mathcal{B}}(x) + |x|)$$
- (weaker) average-case simulation $\mathcal{A} \prec_D \mathcal{B}$ w.r.t. D:
  $\forall \epsilon > 0$ $\exists c > 0$
  $$\mathop{\mathbf{E}}_{x \leftarrow D_n}[{t_{\mathcal{A}}}^c(x)] = O(n \mathop{\mathbf{E}}_{y \leftarrow D_n}[{t_{\mathcal{B}}}^\epsilon(y)])$$
- (weaker) simulation scheme:
  simulate everywhere except for the set of D-prob. $1/2d$.
- (yet weaker!) worst-case simulation $\mathcal{A} \prec_{wc} \mathcal{B}$:
  $\exists$ polynomials p, q $\forall x$
  $$t_{\mathcal{A}}(x) \leq p(\max_{\substack{|x'| \leq q(|x|) \\ x' \in L}} t_{\mathcal{B}}(x') + |x|)$$

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction:
    On input x, run $|x|$ algorithms in parallel:
    1. $A_1(x)$ (brute-force search); output the result;
    2. $A_2(x)$; check the solution; output if it's correct;

    ... ...

    n. $A_{|x|}(x)$; check the solution; output if it's correct.

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction:
    On input x, run $|x|$ algorithms in parallel:
    1. $A_1(x)$ (brute-force search); output the result;
    2. $A_2(x)$; check the solution; output if it's correct;

    ... ...

    n. $A_{|x|}(x)$; check the solution; output if it's correct.

    Not a pointwise optimal acceptor for **co -NP** problems;

    Not a pointwise optimal acceptor for **NP** problems;

    Main obstacle: how to verify a 1-bit answer to a decision problem?

    Worst-case optimal acceptor for **NP**-complete problems:
    extract satisfying assignment for F by queries to $F[v = 0]$, $F[v = 1]$, ...

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction.
  - worst-case (and stronger) optimal randomized acceptor for GNI:
    verification by Goldwasser-Micali-Sipser protocol.

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction.
  - worst-case (and stronger) optimal randomized acceptor for GNI.
  - pointwise-optimal acceptor for Time(f)-immune sets [Messner],
    pointwise-optimal algorithm for bi-immune sets [Chen,Flum,Müller].

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction.
  - worst-case (and stronger) optimal randomized acceptor for GNI.
  - pointwise-optimal acceptor, algorithm for a set in $\mathbf{E} \setminus \mathbf{P}$.
- Distributional problem (D, L): is $x \in L$ with accuracy d?
  Complexity measure = time(n, d).
  Errorless average-case complexity: count **E** or give up with D-prob. $1/d$.
  - average-case optimal randomized acceptor for GNI for some D.

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction.
  - worst-case (and stronger) optimal randomized acceptor for GNI.
  - pointwise-optimal acceptor, algorithm for a set in $\mathbf{E} \setminus \mathbf{P}$.
- Distributional problem (D, L): is $x \in L$ with accuracy d?
  Complexity measure = time(n, d).
  Errorless average-case complexity: count **E** or give up with D-prob. $1/d$.
  - average-case optimal randomized acceptor for GNI for some D.
- Same problem, solved by heuristic algorithms:
  allow false negatives and positives with D-prob. $1/d$.
  - pointwise optimal randomized *algorithm* for **Im** of an injective function,
  - "scheme-optimal" deterministic *algorithm* for —"—"—.

Problems and complexities

- Decision problem L: is $x \in L$?
  Solved by decision algorithms: complexity measure = time.
- Same problem, solved by acceptors: complexity measure = time on L.
  - worst-case optimal acceptor for **NP**-complete problems:
    Levin's universal search + self-to-decision reduction.
  - worst-case (and stronger) optimal randomized acceptor for GNI.
  - pointwise-optimal acceptor, algorithm for a set in $\mathbf{E} \setminus \mathbf{P}$.
- Distributional problem (D, L): is $x \in L$ with accuracy d?
  Complexity measure = time(n, d).
  Errorless average-case complexity: count **E** or give up with D-prob. $1/d$.
  - average-case optimal randomized acceptor for GNI for some D.
- Same problem, solved by heuristic algorithms:
  allow false negatives and positives with D-prob. $1/d$.
  - pointwise optimal randomized *algorithm* for **Im** of an injective function,
  - "scheme-optimal" deterministic *algorithm* for —"—"—.
- Distributional proving problem (D, L): **supp** $D \subseteq \overline{L}$.
  Solved by heuristic acceptors, may allow false positives only.
  - pointwise optimal randomized heuristic acceptor for p.-t.s. D, r.e. L.

Heuristic acceptors

Distributional proving problem (D, L) consists of a language L of "theorems" and a polynomial-time samplable distribution $D = \{D_n\}_{n \in \mathbb{N}}$ on $\overline{L}$.

Definition

Heuristic acceptor A for (D, L):

(completeness) $\forall x \in L \ \forall d \in \mathbb{N} \quad A(x, d) = 1.$

(correctness) $\Pr_{r \leftarrow D_n} \left\{ \Pr_A\{A(r, d) = 1\} > \frac{1}{8} \right\} < \frac{1}{d}.$

(correctness') $\Pr_{r \leftarrow D_n;\ A} \{A(r, d) = 1\} < \frac{1}{d}.$

Heuristic acceptors

Distributional proving problem (D, L) consists of a language L of "theorems" and a polynomial-time samplable distribution $D = \{D_n\}_{n \in \mathbb{N}}$ on $\overline{L}$.

Definition

Heuristic acceptor A for (D, L):

(completeness) $\forall x \in L \; \forall d \in \mathbb{N} \quad A(x, d) = 1.$

(correctness) $\Pr_{r \leftarrow D_n} \left\{ \Pr_A\{A(r, d) = 1\} > \frac{1}{8} \right\} < \frac{1}{d}.$

(correctness') $\Pr_{r \leftarrow D_n;\, A} \{A(r, d) = 1\} < \frac{1}{d}.$

- Time $\tau_A(x, d)$ is a random variable.
- For random variable X, define $\mu^{(p)}[X] = \min\{T : \Pr[X \geq T] \geq p\}$.
- $t_A(x) = \mu^{(1/2)}[\tau_A(x, d)]$ is the median running time of $A(x, d)$.

Heuristic acceptors

Distributional proving problem (D, L) consists of a language L of "theorems" and a polynomial-time samplable distribution $D = \{D_n\}_{n \in \mathbb{N}}$ on $\overline{L}$.

Definition

Heuristic acceptor A for (D, L):

(completeness) $\forall x \in L \ \forall d \in \mathbb{N} \quad A(x, d) = 1.$

(correctness) $\Pr_{r \leftarrow D_n} \left\{ \Pr_A\{A(r, d) = 1\} > \frac{1}{8} \right\} < \frac{1}{d}.$

(correctness') $\Pr_{r \leftarrow D_n;\, A} \{A(r, d) = 1\} < \frac{1}{d}.$

- Time $\tau_A(x, d)$ is a random variable.
- For random variable X, define $\mu^{(p)}[X] = \min\{T \colon \Pr[X \geq T] \geq p\}$.
- $t_A(x) = \mu^{(1/2)}[\tau_A(x, d)]$ is the median running time of $A(x, d)$.

Theorem

*$\exists$ polynomial-time samplable D $\exists L \in$ **co-NP** $\nexists$ polynomial-time heuristic acceptor for (D, L) $\iff$ $\exists$ infinitely-often one-way function.*

Optimal heuristic acceptor

Definition

Heuristic acceptor S simulates W if there are polynomials p and q such that $\forall x \in L$, $\forall d \in \mathbb{N}$, $t_S(x,d) \le \max_{d' \le q(d \cdot |x|)} p(t_W(x,d') \cdot |x| \cdot d)$.

Idea: Certify A_i by testing it on samples $x \leftarrow D_n$.

Optimal heuristic acceptor

Definition

Heuristic acceptor S simulates W if there are polynomials p and q such that $\forall x \in L$, $\forall d \in \mathbb{N}$, $\quad t_S(x,d) \le \max_{d' \le q(d \cdot |x|)} p(t_W(x,d') \cdot |x| \cdot d)$.

Idea: Certify A_i by testing it on samples $x \leftarrow D_n$.

Optimal heuristic acceptor $U(x,d)$:

- For each $i \le \log |x|$ in parallel:
  1. Execute $A_i(x,d')$.

Optimal heuristic acceptor

Definition

Heuristic acceptor S simulates W if there are polynomials p and q such that $\forall x \in L$, $\forall d \in \mathbb{N}$, $$t_S(x, d) \leq \max_{d' \leq q(d \cdot |x|)} p(t_W(x, d') \cdot |x| \cdot d).$$

Idea: Certify A_i by testing it on samples $x \leftarrow D_n$.

Optimal heuristic acceptor $U(x, d)$:

- For each $i \leq \log |x|$ in parallel:
  1. Execute $A_i(x, d')$.
  2. If it accepts (in T_i steps), test its correctness: let $E_i = 0$ and execute k times:
     - $r \leftarrow D_{|x|}$,
     - if $A_i(r, d') = 1$ in T_i steps, then $E_i := E_i + 1$;

Optimal heuristic acceptor

Definition

Heuristic acceptor S simulates W if there are polynomials p and q such that $\forall x \in L$, $\forall d \in \mathbb{N}$, $$t_S(x,d) \le \max_{d' \le q(d \cdot |x|)} p(t_W(x,d') \cdot |x| \cdot d).$$

Idea: Certify A_i by testing it on samples $x \leftarrow D_n$.

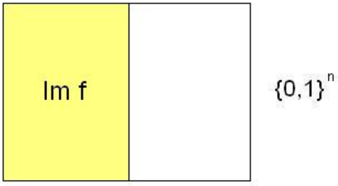
Optimal heuristic acceptor $U(x,d)$:

- For each $i \le \log |x|$ in parallel:
  1. Execute $A_i(x,d')$.
  2. If it accepts (in T_i steps), test its correctness: let $E_i = 0$ and execute k times:
     - $r \leftarrow D_{|x|}$,
     - if $A_i(r,d') = 1$ in T_i steps, then $E_i := E_i + 1$;
  3. If $E_i < \delta k$, output "1".

Here $d' = 4d|x|$, $k = 2d^3|x|^3$, $\delta = \frac{1}{2d|x|}$.

Derandomization

Deterministic *scheme*-optimal acceptor for $(U, \overline{\mathbf{Im}\, f})$, where...



- $f : \{0,1\}^* \to \{0,1\}^*$,
- $|f(x)| = |x| + 1$,
- f is injective,
- f is polynomial-time computable.

Derandomization

Deterministic *scheme*-optimal acceptor for $(U, \overline{\mathbf{Im}\, f})$,

- Use pseudorandom graph based on expanders.
- The input is a source of randomness!

Derandomization

Deterministic *scheme*-optimal acceptor for $(U, \overline{\mathbf{Im}\, f})$,

- Use pseudorandom graph based on expanders.
- The input is a source of randomness!
- Not optimal when the simulated algorithm is erroneously disqualified.

Definition

Simulation **scheme** of A by A':

Simulate everywhere except for the fraction $\frac{1}{2d}$:

$\exists$ polynomials p, q $\forall n, d \in \mathbb{N}$

$$\Pr_{x \leftarrow D_n} [t_A(x, d) \leq p(n \cdot d \cdot t_{A'}(x, q(n, d)))] \geq 1 - \frac{1}{2d},$$

$$q(n, d) \geq 2d.$$

Graph nonisomorphism

$\mathrm{GNI} = \{(G_1, G_2) \mid G_1 \not\cong G_2, |V(G_1)| = |V(G_2)|\}$,

- n is the number of vertices,
- G^π is the result of permuting $V(G)$ by $\pi \in S_n$.

Graph nonisomorphism

$\mathrm{GNI} = \{(G_1, G_2) \mid G_1 \not\cong G_2, |V(G_1)| = |V(G_2)|\}$,

- n is the number of vertices,
- G^π is the result of permuting $V(G)$ by $\pi \in S_n$.

Recall **two-round interactive protocol** for GNI [Goldreich, Micali, Wigderson, 1987]:

- Prover claims that $G_1 \not\cong G_2$;
- Verifier picks random $i \in \{1, 2\}$, $\pi \in S_n$ and sends G_i^π;
- Prover sends j;
- Verifier accepts if $i = j$.

Graph nonisomorphism

$\mathrm{GNI} = \{(G_1, G_2) \mid G_1 \not\cong G_2, |V(G_1)| = |V(G_2)|\}$,

- n is the number of vertices,
- G^π is the result of permuting $V(G)$ by $\pi \in S_n$.

Recall **two-round interactive protocol** for GNI

[Goldreich, Micali, Wigderson, 1987]:

- Prover claims that $G_1 \not\cong G_2$;
- Verifier picks random $i \in \{1, 2\}$, $\pi \in S_n$ and sends G_i^π;
- Prover sends j;
- Verifier accepts if $i = j$.

If the claim is wrong, Verifier rejects with probability $\geq 1/2$.

Correcting a GNI algorithm

SelfCorrect$_{A,N}$, corrects any (randomized) algorithm A:

- Run $N+1$ instances of A in parallel for random $\pi_{ij} \in S_n$:
  - $A({G_1}^{\pi_{11}}, {G_1}^{\pi_{12}})$
  - $A({G_1}^{\pi_{21}}, {G_1}^{\pi_{22}})$
  - $\ldots$
  - $A({G_1}^{\pi_{N1}}, {G_1}^{\pi_{N2}})$
  - $A({G_1}^{\pi_{N+1,1}}, {G_2}^{\pi_{N+1,2}})$
- Return 1 if the last instance was the fastest; otherwise diverge.

Correcting a GNI algorithm

SelfCorrect$_{A,N}$, corrects any (randomized) algorithm A:

- Run $N+1$ instances of A in parallel for random $\pi_{ij} \in S_n$:
  - $A({G_1}^{\pi_{11}}, {G_1}^{\pi_{12}})$
  - $A({G_1}^{\pi_{21}}, {G_1}^{\pi_{22}})$
  - ...
  - $A({G_1}^{\pi_{N1}}, {G_1}^{\pi_{N2}})$
  - $A({G_1}^{\pi_{N+1,1}}, {G_2}^{\pi_{N+1,2}})$
- Return 1 if the last instance was the fastest; otherwise diverge.

Lemma

- *If $G_1 \simeq G_2$, then* $\Pr[accept] \leq \frac{1}{N+1}$.
- *If $G_1 \not\simeq G_2$ and A errs with probability $\leq \frac{1}{2^n}$, then* $\Pr[accept] \geq 1 - \frac{N+1}{2^n}$.

Optimal acceptor for GNI

Algorithm $Opt(G_1, G_2)$:

- Execute in parallel:
  - $A_1(G_1, G_2)$ (brute-force search),
  - 3 times $\mathrm{SelfCorrect}_{A_2,30n}(G_1, G_2)$,
  - 3 times $\mathrm{SelfCorrect}_{A_3,30n}(G_1, G_2)$,
  - ...
  - 3 times $\mathrm{SelfCorrect}_{A_n,30n}(G_1, G_2)$.
- Accept if any of the $3n + 1$ parallel threads accepts.

Optimal acceptor for GNI

Algorithm $Opt(G_1, G_2)$:

- Execute in parallel:
  - $A_1(G_1, G_2)$ (brute-force search),
  - 3 times $\mathrm{SelfCorrect}_{A_2, 30n}(G_1, G_2)$,
  - 3 times $\mathrm{SelfCorrect}_{A_3, 30n}(G_1, G_2)$,
  - ...
  - 3 times $\mathrm{SelfCorrect}_{A_n, 30n}(G_1, G_2)$.
- Accept if any of the $3n + 1$ parallel threads accepts.

Lemma (correctness)

If $G_1 \simeq G_2$, *then* $\Pr[Opt(G_1, G_2) = 1] \leq \frac{3n}{30n+1} < \frac{1}{10}$.

Optimal acceptor for GNI

Algorithm $Opt(G_1, G_2)$:

- Execute in parallel:
  - $A_1(G_1, G_2)$ (brute-force search),
  - 3 times $\mathrm{SelfCorrect}_{A_2,30n}(G_1, G_2)$,
  - 3 times $\mathrm{SelfCorrect}_{A_3,30n}(G_1, G_2)$,
  - ...
  - 3 times $\mathrm{SelfCorrect}_{A_n,30n}(G_1, G_2)$.
- Accept if any of the $3n + 1$ parallel threads accepts.

Lemma (correctness)

If $G_1 \simeq G_2$, *then* $\Pr[Opt(G_1, G_2) = 1] \leq \frac{3n}{30n+1} < \frac{1}{10}$.

Lemma (simulation)

For any randomized acceptor A for GNI $\exists$ polynomial p such that $\forall x \in GNI,\ t_{Opt}(x) \leq p\left(\mu^{(1/4)}_{y \leftarrow U(C_x)}[\tau_A(y)]\right)$, *where* $C_{(G_1,G_2)} = \{(G_1^{\pi_1}, G_2^{\pi_2}) \mid \pi_1, \pi_2 \in S_n\}$ *is a cluster of* (G_1, G_2).

Optimal acceptor for GNI

Algorithm $Opt(G_1, G_2)$:

- Execute in parallel:
  - $A_1(G_1, G_2)$ (brute-force search),
  - 3 times $\text{SelfCorrect}_{A_2,30n}(G_1, G_2)$,
  - 3 times $\text{SelfCorrect}_{A_3,30n}(G_1, G_2)$,
  - ...
  - 3 times $\text{SelfCorrect}_{A_n,30n}(G_1, G_2)$.
- Accept if any of the $3n + 1$ parallel threads accepts.

Lemma (simulation)

For any randomized acceptor A for GNI $\exists$ *polynomial p such that*
$\forall x \in GNI,\ t_{Opt}(x) \le p\left(\mu^{(1/4)}_{y \leftarrow U(C_x)}[\tau_A(y)]\right)$, *where*
$C_{(G_1,G_2)} = \{(G_1^{\pi_1}, G_2^{\pi_2}) \mid \pi_1, \pi_2 \in S_n\}$ *is a cluster of* (G_1, G_2).

Optimal acceptor for GNI

Algorithm $Opt(G_1, G_2)$:

- Execute in parallel:
  - $A_1(G_1, G_2)$ (brute-force search),
  - 3 times $\text{SelfCorrect}_{A_2, 30n}(G_1, G_2)$,
  - 3 times $\text{SelfCorrect}_{A_3, 30n}(G_1, G_2)$,
  - ...
  - 3 times $\text{SelfCorrect}_{A_n, 30n}(G_1, G_2)$.
- Accept if any of the $3n + 1$ parallel threads accepts.

Lemma (simulation)

For any randomized acceptor A for GNI $\exists$ polynomial p such that $\forall x \in GNI,\ t_{Opt}(x) \leq p\left(\mu^{(1/4)}_{y \leftarrow U(C_x)}[\tau_A(y)]\right)$, where $C_{(G_1,G_2)} = \{(G_1^{\pi_1}, G_2^{\pi_2}) \mid \pi_1, \pi_2 \in S_n\}$ is a cluster of (G_1, G_2).

Corollary

Opt is average-case optimal provided D is uniform on every cluster.

From acceptors to proof systems

Definition

L is paddable if there is an injective non-length-decreasing polynomial-time padding function $\mathrm{pad}_L \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ that is polynomial-time invertible on its image and such that $\forall x, w\ (x \in L \iff \mathrm{pad}_L(x, w) \in L)$.

Optimal proof [Messner, 99]:

- A proof π of x in some system Π;
- padding.

Verification:

- run optimal acceptor on $\mathrm{pad}_L(x, \pi)$;
- for a correct proof π, it accepts in a polynomial time because for a correct system Π, the set $\{\mathrm{pad}_L(x, \pi) \mid x \in L,\ \Pi(x, \pi) = 1\} \subseteq L$ can be accepted in a polynomial time.

From acceptors to proof systems

Optimal proof [Messner, 99]:

- A proof π of x in some system Π;
- padding.

Verification:

- run optimal acceptor on $\mathrm{pad}_L(x, \pi)$;
- for a correct proof π, it accepts in a polynomial time because for a correct system Π, the set $\{\mathrm{pad}_L(x, \pi) \mid x \in L,\ \Pi(x, \pi) = 1\} \subseteq L$ can be accepted in a polynomial time.

Applicability:

- Messner's proof goes for randomized algorithms.

From acceptors to proof systems

Optimal proof [Messner, 99]:

- A proof π of x in some system Π;
- padding.

Verification:

- run optimal acceptor on $\mathrm{pad}_L(x, \pi)$;
- for a correct proof π, it accepts in a polynomial time because for a correct system Π, the set $\{\mathrm{pad}_L(x, \pi) \mid x \in L,\ \Pi(x, \pi) = 1\} \subseteq L$ can be accepted in a polynomial time.

Applicability:

- Messner's proof goes for randomized algorithms.
- Does not go for heuristic, average-case algorithms.

Heuristic proof systems

- Allow probabilistic proof verification (with bounded error).
- Allow small number of false theorems (unbounded error there) according to D.
- Heuristic computation: gets d on input and makes at most $\frac{1}{d}$ errors.

Heuristic proof systems

- Allow probabilistic proof verification (with bounded error).
- Allow small number of false theorems (unbounded error there) according to D.
- Heuristic computation: gets d on input and makes at most $\frac{1}{d}$ errors.

Definition

Heuristic proof system for (D, L) is a polynomial-time Π such that

(completeness) $\forall x \in L\ \forall d \in \mathbb{N}\ \exists w \quad \Pr\{\Pi(x, w, d) = 1\} > \frac{1}{2}$.

(Such w is a Π-proof with confidence d.)

(correctness) $\Pr_{r \leftarrow D_n}\{\exists w\ \{\Pr\{\Pi(r, w, d) = 1\} > \frac{1}{8}\} < \frac{1}{d}\}$.

Heuristic proof systems

- Allow probabilistic proof verification (with bounded error).
- Allow small number of false theorems (unbounded error there) according to D.
- Heuristic computation: gets d on input and makes at most $\frac{1}{d}$ errors.

Definition

Heuristic proof system for (D, L) is a polynomial-time Π such that

(completeness) $\forall x \in L \ \forall d \in \mathbb{N} \ \exists w \quad \Pr\{\Pi(x, w, d) = 1\} > \frac{1}{2}$.

(Such w is a Π-proof with confidence d.)

(correctness) $\Pr_{r \leftarrow D_n}\{\exists w \ \{\Pr\{\Pi(r, w, d) = 1\} > \frac{1}{8}\} < \frac{1}{d}\}$.

Open question: Devise an interesting heuristic p.s.,
i.e., distinguish between distributions hard for heuristic acceptors and heuristic proof systems.

Open questions

- $\exists$ optimal proof system $\iff$ $\exists$ optimal heuristic acceptor;
- $\exists$ optimal heuristic proof system $\overset{?}{\iff}$ $\exists$ optimal heuristic acceptor;
- $\exists$ optimal proof system with advice $\overset{?}{\iff}$ $\exists$ optimal acceptor with advice;
- $\exists$ average-case optimal acceptor?
- $\exists$ optimal acceptor for GNI or any other $\mathbf{co}\text{-}\mathbf{NP} \setminus \mathbf{P}$ problem?
- $\exists$ optimal proof system for any problem outside $\mathbf{P}$?
- $\exists (D, L) \in (\mathbf{co}\text{-}\mathbf{NP}, \mathrm{PSamplable})$ with no polynomially-bounded heuristic proof system $\iff$?