A NOTE ON THE HARDNESS OF VERIFYING AND GENERATING REDUCTIONS

MASON DICICCO

Abstract. This note studies *reductions* between decision problems, commenting on the complexity of two tasks: Verification (is a candidate reduction correct?) and generation (does a correct reduction exist?). We show that (our formulations of) both tasks lie at higher levels of the polynomial hierarchy than the problems themselves.

1. Preliminaries

We work with binary strings over the alphabet $\{0,1\}$. Let $\{0,1\}^n$ denote the set of all n-bit strings, and $\{0,1\}^* = \bigcup_{n\geq 0}\{0,1\}^n$ denote the set of all finite binary strings. For a string x, we write $|x|$ for its length.

Definition 1. A *language* (or *decision problem*) is a subset $L \subseteq \{0,1\}^*$. The *membership problem* for L is: given $x \in \{0,1\}^*$, decide whether $x \in L$.

Definition 2. A language L is in P (polynomial time) if there exists a deterministic Turing machine M (a "decider") and a polynomial p such that for all $x \in \{0,1\}^*$:

- $M(x)$ halts in at most $p(|x|)$ steps, and
- $M(x) = 1$ if and only if $x \in L$.

Definition 3. A language L is in NP (nondeterministic polynomial time) if there exists a polynomial-time computable predicate R (a "verifier") and a polynomial s such that

$$x \in L \iff \exists w \in \{0,1\}^{\leq s(|x|)} : R(x,w) = 1.$$

Here, we call w a *witness* for x. A language is in coNP if its complement is in NP.

Definition 4. The *polynomial hierarchy* is defined inductively:

- $\Sigma_0^p = \Pi_0^p = \mathsf{P}$.
- $\Sigma_{k+1}^p = \mathsf{NP}^{\Sigma_k^p} = \mathsf{NP}^{\Pi_k^p}$ (NP with oracle access to Σ_k^p or, equivalently, Π_k^p).
- $\Pi_{k+1}^p = \mathsf{coNP}^{\Sigma_k^p} = \mathsf{coNP}^{\Pi_k^p}$ (coNP with oracle access to Σ_k^p or Π_k^p).

Equivalently, $L \in \Sigma_k^p$ iff there exist a poly-time predicate R and polynomials $s_1, \ldots, s_k$ such that

$$x \in L \iff \exists w_1 \forall w_2 \exists w_3 \cdots Q_k w_k : R(x, w_1, \ldots, w_k),$$

where $|w_i| \leq s_i(|x|)$ and the quantifiers strictly alternate starting with $\exists$. For Π_k^p, the quantifiers start with $\forall$.

Note that $\Sigma_1^p = \mathsf{NP}$ and $\Pi_1^p = \mathsf{coNP}$.

Definition 5. A *polynomial-time reduction* from F to G is a poly-time computable function M satisfying:

$$x \in F \iff M(x) \in G \quad \text{for all } x.$$

Definition 6. A *predicate schema* of depth k is a tuple

$$S = (Q_1 w_1, \ldots, Q_k w_k : R(x, \boldsymbol{w});\ p, s_1, \ldots, s_k)$$

where:

- Each $Q_i \in \{\exists, \forall\}$ with **strict alternation**: $Q_{i+1} \neq Q_i$ for all $i < k$.
- $R : \{0,1\}^* \times \{0,1\}^* \times \cdots \times \{0,1\}^* \to \{0,1\}$ is a predicate computed by a deterministic Turing machine in time $p(|x|)$ on inputs $(x, w_1, \ldots, w_k)$ where $|w_i| \leq s_i(|x|)$.
- $p, s_1, \ldots, s_k : \mathbb{N} \to \mathbb{N}$ are polynomials.

The schema defines the language

$$L_S = \{x \in \{0,1\}^* : Q_1 w_1 \in \{0,1\}^{\leq s_1(|x|)} \cdots Q_k w_k \in \{0,1\}^{\leq s_k(|x|)}\ R(x, w_1, \ldots, w_k)\}.$$

Examples:

- Depth 0 reduces to $(R(x); p)$ with no quantifiers, so $L_S = \{x : R(x)\}$ where R is computable in time $p(|x|)$. This captures P.
- Depth 1 with leading $\exists$ captures NP: "is there a witness w such that $R(x, w)$ holds?"
- Depth 1 with leading $\forall$ captures coNP: "for all w, does $R(x, w)$ hold?"
- A depth-k schema with leading $\exists$ defines a Σ_k^p language; with leading $\forall$, a Π_k^p language.

Conventions. Bounds n, t, ℓ, s are given in unary. Machines and circuits have polynomial-size encodings. All completeness results are under polynomial-time many-one reductions.

2. Verification

Given a candidate reduction M, bounds $1^n, 1^t$, and specifications of languages F and G, does M correctly reduce F to G on all n-bit inputs in time t? Verifying correctness requires checking a universal property—M must preserve membership for *every* input—which introduces one quantifier alternation beyond the complexity of the language specifications themselves.

2.1. General Verification Theorem.

We first establish the general result for languages specified by predicate schemas. The important special cases of circuits and NP verifiers then follow as immediate corollaries.

Definition 7 (Correct)**. Input:** Tuple $\langle M, S_F, S_G, 1^n, 1^t, 1^\ell \rangle$.

Question: For all $y \in \{0,1\}^n$, does $M(y)$ halt in $\leq t$ steps with $|M(y)| = \ell$, and satisfy

$$y \in L_{S_F} \iff M(y) \in L_{S_G}\,?$$

Theorem 8. *Let* $k = \max(\mathrm{depth}(S_F), \mathrm{depth}(S_G))$. *Then:*

$$\boxed{\textsc{Correct} \text{ is } \Pi_{k+1}^p\text{-complete.}}$$

Proof. Write $A(y) \equiv y \in L_{S_F}$ and $B(z) \equiv z \in L_{S_G}$. Each is a depth-$k$ predicate, lying in Σ_k^p or Π_k^p depending on its leading quantifier.

Membership. The correctness condition is $\forall y\,(A(y) \Leftrightarrow B(M(y)))$. Negating gives

$$\exists y\,\big[(A(y) \wedge \neg B(M(y))) \vee (\neg A(y) \wedge B(M(y)))\big].$$

Each disjunct is a conjunction of a Σ_k^p predicate and a Π_k^p predicate; by merging quantifier prefixes, each conjunction remains in Σ_k^p. A disjunction of Σ_k^p predicates stays in Σ_k^p. The outer $\exists y$ adds one alternation, so the negation lies in Σ_{k+1}^p, and the original problem is in Π_{k+1}^p.

Hardness. Reduce from Π_{k+1}-QSAT. Given $\Phi = \forall u\, Q_1 w_1 \cdots Q_k w_k : \psi(u, \boldsymbol{w})$, set S_F to encode $Q_1 w_1 \cdots Q_k w_k : \psi(u, \boldsymbol{w})$ (with u as input), S_G the trivial true language, and $M = \mathrm{id}$. Correctness becomes $\forall u\,(u \in L_{S_F} \Leftrightarrow \mathsf{true}) = \Phi$. Since S_F has leading quantifier opposite to $\forall$, its complement is Π_k^p, so $\exists u\,(u \notin L_{S_F}) \in \Sigma_{k+1}^p$, confirming Π_{k+1}^p-hardness. □

2.2. **Corollaries for Circuits and NP Verifiers.** The most natural ways to specify decision problems—Boolean circuits and NP verifiers—correspond to depth 0 and depth 1 schemas, respectively. Applying the general theorem immediately yields:

Definition 9 (Correct-Circ)**. Input:** Tuple $\langle M, C_F, C_G, 1^n, 1^t, 1^\ell \rangle$ where C_F is an n-input Boolean circuit and C_G is an ℓ-input Boolean circuit.

Question: For all $y \in \{0,1\}^n$: $M(y)$ halts in $\leq t$ steps, $|M(y)| = \ell$, and $C_F(y) = C_G(M(y))$?

Corollary 10. Correct-Circ *is* coNP*-complete.*

Proof. A Boolean circuit defines a depth-0 predicate (no quantifiers, just a polynomial-time computation). By Theorem 8 with $k = 0$, verification is $\Pi_1^p = \mathsf{coNP}$-complete. Concretely, the correctness condition $\forall y : C_F(y) = C_G(M(y))$ is a single universal quantifier over a polynomial-time predicate. For hardness, reduce from Circuit-Tautology by setting $M = M_{\mathrm{id}}$, $C_G \equiv 1$, so correctness becomes "C_F is a tautology." □

Definition 11 (Correct-NP)**. Input:** Tuple $\langle M, V_F, V_G, 1^n, 1^t, 1^\ell, 1^{s_F}, 1^{s_G} \rangle$ where V_F, V_G are NP verifiers.

Question: For all $y \in \{0,1\}^n$: $M(y)$ halts in $\leq t$ steps, $|M(y)| = \ell$, and

$$(\exists w\, V_F(y, w) = 1) \iff (\exists w\, V_G(M(y), w) = 1)\,?$$

Corollary 12. Correct-NP *is* Π_2^p*-complete.*

Proof. An NP verifier defines a depth-1 predicate with leading $\exists$: $x \in L$ iff $\exists w\, V(x, w)$. By Theorem 8 with $k = 1$, verification is Π_2^p-complete. The correctness condition expands to

$$\forall y : (\exists w_F\, V_F(y, w_F)) \Leftrightarrow (\exists w_G\, V_G(M(y), w_G)).$$

To see this is Π_2^p: the biconditional $P \Leftrightarrow Q$ is equivalent to $(P \wedge Q) \vee (\neg P \wedge \neg Q)$. When $P = \exists w_F\, V_F(\cdot)$ and $Q = \exists w_G\, V_G(\cdot)$, each disjunct involves a conjunction of Σ_1^p and Π_1^p predicates, which remains in $\Sigma_1^p \cup \Pi_1^p \subseteq \Sigma_2^p$. The outer $\forall y$ then gives Π_2^p. Hardness follows from Theorem 8, or directly: reduce from Π_2-QSAT $\forall u\, \exists v : \varphi(u, v)$ by setting $V_F(u, v) = \varphi(u, v)$, $V_G \equiv 1$, and $M = M_{\mathrm{id}}$. □

3. Generation

Given specifications S_F, S_G, does there *exist* a correct reduction?

Definition 13 (Reducible)**. Input:** Tuple $\langle S_F, S_G, 1^n, 1^t, 1^\ell, 1^s \rangle$.

Question: Does there exist a machine M with $|M| \leq s$ such that

$$\langle M, S_F, S_G, 1^n, 1^t, 1^\ell \rangle \in \textsc{Correct}\,?$$

Theorem 14. *Let $k = \max(\text{depth}(S_F), \text{depth}(S_G))$. Then:*

$$\boxed{\textsc{Reducible} \textit{ is } \Sigma^p_{k+2}\textit{-complete.}}$$

Proof. Membership. The question is

$$\exists M\ (|M| \le s)\ \big(\forall y\ A(y) \Leftrightarrow B(M(y))\big).$$

By Theorem 8 the inner predicate "$\forall y\ A(y) \Leftrightarrow B(M(y))$" is in Π^p_{k+1}, and the outer $\exists M$ (over descriptions of polynomial length since s is unary) yields membership in Σ^p_{k+2}.

Hardness. Reduce from Σ_{k+2}-QSAT. Given

$$\Phi = \exists x\ \forall y\ Q_1 w_1 \cdots Q_k w_k : \psi(x, y, \boldsymbol{w}),$$

encode each candidate $x \in \{0,1\}^r$ by a machine M_x that outputs $x \| y$ on input y (so $|M_x| = r + O(1)$). Put $S_F = \{0,1\}^n$ (always true) and let S_G interpret its input z as (x, y) and test $Q_1 \cdots Q_k\ \psi(x, y, \boldsymbol{w})$. Then M_x is a valid reduction iff $\forall y\ Q_1 \cdots Q_k\ \psi(x, y, \boldsymbol{w})$, so $\exists M$ passing $\textsc{Correct}$ iff $\exists x\, \forall y\, Q_1 \cdots Q_k\ \psi$, i.e. iff Φ holds. This is a polynomial-time reduction, giving Σ^p_{k+2}-hardness. □

Remark. The unbounded problem ("does a poly-time reduction exist for all input lengths?") is undecidable. Our bounded version captures the complexity of finding reductions for a specific input length.

Definition 15 ($\textsc{Reducible-Unbounded}$)**. Input:** Turing machines M_F, M_G specifying languages $L_F = L(M_F)$ and $L_G = L(M_G)$.

Question: Does there exist a polynomial-time computable function f such that for all $x \in \{0,1\}^*$:

$$x \in L_F \iff f(x) \in L_G\,?$$

Theorem 16. $\textsc{Reducible-Unbounded}$ *is undecidable.*

Proof. We reduce from the language emptiness problem: given a Turing machine T, is $L(T) = \emptyset$? This is undecidable by Rice's theorem.

Given T, construct an instance of $\textsc{Reducible-Unbounded}$ with $L_F = L(T)$ and $L_G = \emptyset$ (the empty language, specified by a machine that immediately rejects).

We claim: a polynomial-time reduction from L_F to L_G exists if and only if $L_F = \emptyset$.

- ($\Rightarrow$) Suppose f is a reduction and $L_F \neq \emptyset$. Let $x \in L_F$. Then by correctness of f, we need $f(x) \in L_G = \emptyset$, a contradiction.
- ($\Leftarrow$) If $L_F = \emptyset$, then any function f is a valid reduction: the condition $x \in L_F \Leftrightarrow f(x) \in L_G$ becomes $\mathsf{false} \Leftrightarrow \mathsf{false}$, which holds for all x.

Thus $\textsc{Reducible-Unbounded}$ decides language emptiness, so it is undecidable. □