# Reductions for One-Way Functions

by

Mark Liu

A thesis submitted in partial fulfillment
of the requirements for degree of
Bachelor of Science
(Honors Computer Science)
from The University of Michigan
2013

## 1 Introduction

In computer science, reductions are transformations from one problem to another. The concept of reduction has proved fruitful in many areas, such as the theory of NP-completeness [1]. There, Karp reductions, or polynomial-time many-one reductions [2], allowed us to define the set of NP-complete problems within NP, which are problems that can be reduced to by all problems in NP. This has stimulated much research in the field. The idea of reduction has also been extended to other types of problems as well. Valiant gave a definition of reduction for enumeration problems [6], while Papadimitriou and Yannakakis similarly determined the appropriate definition for optimization problems [4]. Both of these discoveries were the start of many fruitful lines of research. Much of the difficulty lies in first defining the appropriate reduction.

One-way functions (OWF's) are functions that are easy to compute but “difficult” to invert. A variant, weak one-way functions, are also easy to compute and difficult to invert, but “less difficult”. We believe it will be fruitful to rigorously define a notion of reduction for the set of one-way functions or any of its variants. In this thesis we present a notion of reductions for the class of weak one-way functions and show it behaves as intended. Our definition is transitive and preserves the hardness condition; if we can reduce $f$ to $g$ and $f$ is ”difficult” to invert, $g$ is as well. We also show that a complete function already exists in Levin's Universal Function [3] by adapting his original proof to fit our definition.

## 2 Worst-case One-way functions

Before discussing weak one-way functions, which are probablistic in nature, we begin with a deterministic version, worst-case one-way functions, and a notion of reducibility for them.

**Definition 2.1.** A function $g$ is a left (right) inverse of $f$ if and only if $g \circ f$ $(f \circ g)$ is the identity function on the domain (codomain) of f.

**Definition 2.2.** The range of $F$, or $ran(F)$ is the set of elements that is mapped to by $F$. In other words, if $F : \Sigma^* \to \Sigma^*$ then $ran(F) = \{y \in \Sigma^* | \exists x F(x) = y\}$

**Definition 2.3.** Honest Function: A function $F : \Sigma^* \to \Sigma^*$ is honest if: $(\exists c > 0)$ $(\forall w)$ $(|F(w)| \geq |w|^c)$. We require that functions be honest so that a right inverse would even have a chance to be polynomial time computable.

**Definition 2.4.** Worst-case One-way: A function $F$ is worst-case one-way if:

1. $F$ is polynomial-time computable
2. $F$ is honest
3. $F$ has no polynomial-time computable right inverse

**Theorem 2.1.** *(Selman)[5] Worst-case one-way functions exist if and only if $P \neq NP$*

*Proof.* " $\Leftarrow$ " Assume $P \neq NP$. Let $A \in NP - P$. Since $A \in NP$, there is a non-deterministic Turing Machine (NDTM) M that recognizes A in polynomial time, with time bound $q(n)$. One characterization of non-determinism is that M, when able to take multiple paths, will "branch" into multiple copies, each of which tries a different path. Then M accepts if at least one branch accepts. Then let

$$f(w,b) = \begin{cases} \langle 1, w \rangle & \text{: if M following branch b accepts in time } q(n) \\ \langle 0, w \rangle & \text{: otherwise} \end{cases}$$

Then $f$ can be computed in polynomial time, since we only need to run one branch of $M$ for time $q(n)$.

$f$ is also honest, because we include $w$ in the output, and while f takes $(w, b)$ as input, $b$'s length is proportional to the running time of $M$, which is polynomial in $|w|$.

Finally, if $f$ had a polynomial time computable right inverse $f'$, we would have an efficient algorithm that would, given $\langle 1, w\rangle$, return $(w, b)$ such that $M$ would accept $\langle 1, w\rangle$ on branch b. From this, we could construct deterministic Turing Machine $M'$ to recognize $A$. For any string $w$, $M'$ would run $f'$ on input $\langle 1, w\rangle$. The output may or may not make sense, but if it is of the form $(w, b)$, we can run $w$ on $M$, taking branch $b$. If we end on an accept state then we know $w$ is in the language $A$. Thus $A \in P$ and we have a contradiction, so $f$ does not have a polynomial time computable right inverse and $f$ is a worst-case one-way function.
" $\Rightarrow$ " Now assume worst-case one-way function $f$ exists.
Define

$$A = \{\langle x, z\rangle : \exists y \text{ such that } f(xy) = z\}$$

Then $A \in NP$ because there is non-deterministic polynomial-time verifier for $A$. Recall that honest functions require that output $z$ must be greater than $|w|^c$ for some $c$, so we only need to try strings of length up to $L = |z|^c - |x|$. For each string $a$ s.t $|a| \leq L$ non-deterministically try $f(xa)$. This should take time polynomial in length of $xa$, since f is weak one-way. Then if any branch accepts, return true. If not, return false.

Show $A \notin P$ by assuming $A \in P$. We get a contradiction by finding polynomial time right inverse for $f$. Given $z$, test $\langle \epsilon, z\rangle \in A$. If it is, then $z$ has inverse image under $f$, otherwise it does not, so terminate. Now test $\langle 0, z\rangle \in A$. If it is, then we have found an element in preimage of $z$ whose first bit is 0, otherwise we know there is element in preimage of $z$ whose first bit is 1. Either way we are guaranteed an element in preimage whose first bit is $b_0$. Then we test $\langle b_0 0, z\rangle \in A$ and $\langle b_0 1, z\rangle \in A$.

If neither accept then we know $b_0$ was in the preimage of $z$. If one of them accepts then set $b_1$ equal to it. We continue this process, learning one bit at a time. Each check for inclusion in $A$ takes polynomial time by assumption, and we know this algorithm will terminate in a polynomial number of steps, since $f$ is honest. Thus we have a polynomial time inverse for $f$, which gives us a contradiction. So $A \notin P$ and $P \neq NP$. $\square$

**Definition 2.5.** We say $a \xrightarrow{F} b$ or "$F$ takes $a$ to $b$" if $F$ maps $a$ to $b$. This is equivalent to writing $F(a) = b$. We prefer this notation because it will generalize better to the case when $F$ is not a function but a $PPT$.

**Definition 2.6.** A reduction from $F : \Sigma^* \to \Sigma^*$ to $G : \Sigma^* \to \Sigma^*$ consists of two functions $A, B$ s.t

1. $B, A$ are polynomial time computable.
2. $(x \xrightarrow{G} B(y)) \implies (A(x) \xrightarrow{F} y)$.
3. $(x \xrightarrow{F} y) \implies (\exists x'$ such that $x' \xrightarrow{G} B(y))$, or equivalently,
   $y \in ran(F) \implies B(y) \in ran(G)$.

$$\begin{tikzcd}
\Sigma^* \arrow[d, "F"'] & \Sigma^* \arrow[l, "A"'] \arrow[d, shift right, "G"'] \\
\Sigma^* \arrow[r, "B"] & \Sigma^* \arrow[u, shift right, "G'"']
\end{tikzcd}$$

Figure 1: Necessary functions for reduction from $F$ to $G$

**Proposition 2.2.** *If $F$ reduces to $G$ and $F$ has no polynomial time right inverse, then $G$ has no polynomial time right inverse. Equivalently, if $F, G$ are polynomial time computable and honest, then $F$ worst-case one-way implies $G$ worst-case one-way.*

*Proof.* Let $F$ have no polynomial time right inverse, and suppose a reduction from $F$ to $G$ exists. Assume towards a contradiction that $G$ has some polynomial partial right inverse $G'$. Let

$$F' = A \circ G' \circ B$$

$F'$ is polynomial time computable because it is the composition of polynomial time computable functions. We claim that $F'$ is a right inverse of F. Let $y \in ran(F)$ be arbitrary. Then by condition (3), we have that $B(y) \in ran(G)$. So then, since $G'$ is a right inverse of $G$,

$$G' \circ B(y) \xrightarrow{G} B(y)$$

Then by condition (2),

$$A \circ G' \circ B(y) \xrightarrow{F} y$$

and by our earlier choice of $F'$, we see that $F \circ F'(y) = y$ so we have contradiction, so $F$ worst-case one-way implies $G$ worst-case one-way. □

## 3 Weak one-way functions

**Definition 3.1.** $\mathbb{P} : X \to [0,1]$ is a discrete probability measure on a space $X$ if:

1. $\mathbb{P}(X) = 1, \mathbb{P}(\varnothing) = 0$
2. For every countable collection $\{E_i\}$ of pairwise disjoint sets, $\mathbb{P}(\bigcup_i E_i) = \sum\limits_i \mathbb{P}(E_i)$ Then in particular, we have: $\forall S \subset X, \mathbb{P}(S) = \sum\limits_{x \in S} \mathbb{P}(\{x\})$

**Definition 3.2.** We use $\mathbb{P}^n$ to denote the uniform probability measure on strings of length $n$. Let $F : \Sigma^n \to Y$ and $\mathbb{P}^n$ be a probability measure on $\Sigma^n$. Then define $\mathbb{P}^n_F : Y \to [0,1]$ to be the probability measure on Y such that $\forall S \subset Y, \mathbb{P}^n_F(S) = \mathbb{P}^n(F^{-1}[S])$ where $F^{-1}[S]$ is the preimage of $S$, or $\{x \in \Sigma^n | F(x) \in S\}$.

**Definition 3.3.** Another convenient notation we use is $\mathbb{P}\{F = G\}$, where $F, G$ are functions. By this notation, we actually mean the set $\mathbb{P}[\{x | F(x) = G(x)\}]$

**Definition 3.4.** A probabilistic polynomial time (PPT) algorithm is one that runs in polynomial time, and also has the capability to "toss coins", meaning it has access to a source of unbiased random bits. Alternatively, we can think of a PPT algorithm as a deterministic algorithm that is given as input a random string of length $p(n)$ where $p$ is some polynomial and $n$ is the size of the input. This is the characterization we use throughout the rest of the paper.

**Definition 3.5.** A function $\epsilon : N \to R$ is *negligible* if $\forall c > 0, \exists n_0$ such that $\epsilon(n) < 1/n^c$ for all $n \geq n_0$.

We can now move to weak one-way functions. The main difference between weak one-way functions and worst-case one-way functions is that we move to a probablistic setting. Firstly, we allow adversaries to use PPT algorithms. Secondly, instead of requiring that any such adversary must always fail, we require that the adversary should fail, with non-negligible probability, to invert the function. Formally,

**Definition 3.6.** We will use the notion of "augmenting" $F$ and $G$ to "carry" information about the length of the input, so that for any $F : \Sigma^* \to \Sigma^*$, define $\overline{F} : \Sigma^* \to 1^* \times \Sigma^*$ such that $\overline{F}(x) = (1^{|x|}, F(x))$. This is neccessary for our reductions so that we do not "lose" information about the length of the string when reducing one string to another.

**Definition 3.7.** We say $\phi$ holds almost everywhere (a.e.) if $\exists M$ such that $\phi(n)$ holds for $n > M$. Similarly, we say $\phi$ holds infinitely often (i.o.) if $\forall M\ \exists n$ such that $\phi(n)$ holds.

**Definition 3.8.** A polynomial-time computable function $F : \Sigma^* \to \Sigma^*$ is weak-one way if: $\exists k > 0\ \forall PPT\ \overline{F}'$ :

$$\mathbb{P}^m_{\overline{F}}[\overline{F} \circ \overline{F}' = I] \leq 1 - 1/m^k \text{ a.e.}$$

or equivalently,

$$\mathbb{P}^m_{\overline{F}}[\overline{F} \circ \overline{F}' \neq I] \geq 1/m^k \text{ a.e.}$$

where $I(1^m, y, r) = \langle 1^m, y\rangle$ and $r$ is the random string used by PPT. Note that $\mathbb{P}^m_{\overline{F}}[\overline{F} \circ \overline{F}' = I]$ is a convenient abuse of notation, as $\{\overline{F} \circ \overline{F}' = I\} \subset 1^m \times \Sigma^* \times \Sigma^*$, while $\mathbb{P}^m_{\overline{F}}$ is defined on $1^m \times \Sigma^*$.
By $\{\overline{F} \circ \overline{F}' = I\}$ we mean the set of 3-tuples $\langle 1^m, y, r\rangle$ correctly inverted by $\overline{F}'$, where

$$\mathbb{P}^m_{\overline{F}}[1^m, y, r] = \mathbb{P}^m_{\overline{F}}[1^m, y]\mathbb{P}(r)$$

where $\mathbb{P}(r) = 1/2^{|r|}$ since $r$ is selected randomly from strings of a fixed length.

**Definition 3.9.** Now with the goal of preserving the weak one-way hardness condition, we will define a new reduction from a polynomial-time computable function $F : \Sigma^* \to \Sigma^*$ to another polynomial-time computable function $G : \Sigma^* \to \Sigma^*$. Our reduction consists of:
$\alpha : \mathbb{N} \to \mathbb{N}$ such that $\exists c, d > 0$ s.t $n^c \leq \alpha(n) \leq n^d$. $\alpha$ should be computable in time $O(n^g)$ for some $g$.
$A : \Sigma^* \to \Sigma^*$ such that $|A(x)| = \alpha(|x|)$, and a sequence of functions
$B_n : \langle 1^*, \Sigma^*\rangle \to \langle 1^n, \Sigma^*\rangle$ such that $B_n(1^m, x) = B(1^n, 1^m, x)$ for some function $B$.
and the following conditions must hold:

1. $A, B$ are polynomial-time computable.

2. Whenever $m = \alpha(n)$,
   $(\forall x \in \Sigma^n)$, $(\forall y \in \Sigma^*)$,
   $[(x \xrightarrow{\overline{G}} B_n(\langle 1^m, y\rangle)) \implies (A(x) \xrightarrow{\overline{F}} \langle 1^m, y\rangle)]$

3. Whenever $m = \alpha(n)$,
   $(\exists j > 0)$ $(\forall S \subset ran(\overline{F}))$,
   $\mathbb{P}^n_{\overline{G}}(B_n[S]) \geq \mathbb{P}^m_{\overline{F}}(S)/n^j$

Conditions 2) and 3) are analagous to conditions 2) and 3) for worst-case one way reductions, except more care is needed to handle this probablistic setting.

$$\begin{tikzcd}
\Sigma^m, \mathbb{P} \arrow[d, "\overline{F}"'] & \Sigma^n, \mathbb{P}^n \arrow[l, "A"'] \arrow[d, shift right, "\overline{G}"'] \\
\{1^m\} \times \Sigma^* \arrow[r, "B_n"'] & \{1^n\} \times \Sigma^* \arrow[u, shift right, "\overline{G'}"']
\end{tikzcd}$$

Figure 2: Necessary functions for reduction from $\overline{F}$ to $\overline{G}$

## 4 Goals

We will show that our definition meets the following goals:

- If a reduction can be constructed from poly-time computable $F$ to poly-time computable $G$, and $F$ is weak OWF, then $G$ must be as well.
- If we have a reduction from $F$ to $G$, and a reduction from $G$ to $H$, we should be able to construct a reduction from $F$ to $H$. (Transitivity)
- Every weak OWF is reducible to Levin's Complete Function.

### 4.1 Weak OWF Hardness Property Preserved

**Theorem 4.1.** *If $F, G$ poly-time computable and such a reduction exists from $F : \Sigma^* \to \Sigma^*$ to $G : \Sigma^* \to \Sigma^*$, and $F$ is a weak OWF, then $G$ is as well.*

*Proof.* Let $F : \Sigma^* \to \Sigma^*$, $G : \Sigma^* \to \Sigma^*$ be polynomial-time computable functions, and let $F$ be a weak OWF and assume a reduction exists from $F$ to $G$. Then $\exists k > 0$ such that for any PPT algorithm $\overline{F}'$

$$\mathbb{P}\{\overline{F} \circ \overline{F'} \neq I\} \geq 1/m^k \text{ a.e.}$$

Now let $\overline{G}'$ be an arbitrary PPT function that attempts to invert $\overline{G}$. We want to show the probability that $\overline{G}'$ fails to invert is high.

To do this, we construct a PPT $\overline{F}'$ with the goal of inverting $\overline{F}$, using $\overline{G}'$ as a subfunction. $\overline{F}'$ behaves as follows: On input $\langle 1^m, y, r\rangle$, where $y \in \Sigma^*$ and $r$ is the random string used by the inverter, for each $B_n$ such that $\alpha(n) = m$, first compute

$$(1^n, y') = B_n(1^m, y)$$

and then compute

$$x' = \overline{G}'(1^n, y', r)$$

In other words, we are using the random string that $\overline{F}'$ receives and using it as the random string for $\overline{G}'$. Finally let

$$x = A(x')$$

Then $\overline{F}'$ checks if $\overline{F}(x) = \langle 1^m, y\rangle$. If it is, then $\overline{F}'$ outputs x. If not, it continues to loop through each $B_n$ such that $\alpha(n) = m$. Since there are only polynomially many $n$ such that this holds, and each iteration computes a composition of polynomial-time computable functions, $\overline{F}'$ runs in polynomial time.
Recall that

$$\mathbb{P}^m_{\overline{F}}\{\overline{F} \circ \overline{F'} \neq I\} \geq 1/m^k \text{ a.e.}$$

Let $S = \{\overline{F} \circ \overline{F'} \neq I\}$. In other words, for each $\langle 1^m, y, r\rangle \in S$,

$$\overline{F} \circ \overline{F}'(\langle 1^m, y, r\rangle) \neq \langle 1^m, y\rangle$$

Then

$$\mathbb{P}^m_{\overline{F}}[S] = \mathbb{P}^m_{\overline{F}}\{\overline{F} \circ \overline{F'} \neq I\} \geq 1/m^k$$

Now for an arbitrary 3-tuple $\langle 1^m, y, r\rangle \in S$, consider $\langle B_n(1^m, y), r\rangle$.

$$\begin{aligned}
\mathbb{P}^n_{\overline{G}}[B_n(1^m, y), r] &= \mathbb{P}^n_{\overline{G}}[B_n(1^m, y)]\mathbb{P}(r) \text{ since r is independently chosen} \\
&\geq \frac{1}{m^j}(\mathbb{P}^m_{\overline{F}}[1^m, y]\mathbb{P}(r)) \text{ for some j, by condition 3)} \\
&= \frac{1}{m^j}\mathbb{P}^m_{\overline{F}}[1^m, y, r]
\end{aligned}$$

Then let $T = \{\langle B_n(1^m, y), r\rangle | \langle 1^m, y, r\rangle \in S\}$

$$\begin{aligned}
\mathbb{P}^n_{\overline{G}}[T] &= \sum_{\langle 1^m, y, r\rangle \in S} \mathbb{P}^n_{\overline{G}}[\langle B_n(1^m, y), r\rangle] \\
&\geq \frac{1}{m^j} \sum_{\langle 1^m, y, r\rangle \in S} \mathbb{P}^m_{\overline{F}}[\langle 1^m, y, r\rangle] \\
&= \frac{1}{m^j}\mathbb{P}^m_{\overline{F}}[S] \\
&\geq \frac{1}{m^{k+j}}
\end{aligned}$$

So our set $T$ is "large". We would like to show $T$ is not inverted by $\overline{G}'$. Now, for an arbitrary 3-tuple $\langle 1^n, y', r\rangle \in T$ (where $(1^n, y') = B_n(1^m, y)$ and $(1^m, y, r) \in S$), we show that the inverter $G'$ will fail. Assume not. Then

$$\begin{aligned}
\overline{G} \circ \overline{G}'(1^n, y', r) = (1^n, y') &\Rightarrow \overline{G}'(1^n, y', r) \xrightarrow{\overline{G}} B_n(1^m, y) \\
&\Rightarrow A \circ \overline{G}'(1^n, y', r) \xrightarrow{\overline{F}} \langle 1^m, y\rangle \\
&\Rightarrow A \circ \overline{G}'(B_n(1^m, y), r) \xrightarrow{\overline{F}} \langle 1^m, y\rangle
\end{aligned}$$

by condition 2). But the last statement would imply that

$$\overline{F} \circ \overline{F}'(1^m, y, r) = \langle 1^m, y\rangle$$

since it is one step of the inverter $\overline{F}'$. Then we have that

$$\overline{G} \circ \overline{G}'(1^n, y', r) = (1^n, y') \Rightarrow \overline{F} \circ \overline{F}'(1^m, y, r) = \langle 1^m, y \rangle$$

But $(1^m, y, r) \in S$, so

$$\overline{G} \circ \overline{G}'(1^n, y', r) \neq (1^n, y')$$

for all $(1^n, y', r) \in T$ and so we have

$$\mathbb{P}^n_{\overline{G}}\{\overline{G} \circ \overline{G}' \neq I\} \geq 1/m^{k+j} \text{ a.e.}$$

But $m = \alpha(n) < n^d$, so

$$\mathbb{P}^n_{\overline{G}}\{\overline{G} \circ \overline{G}' \neq I\} \geq 1/n^{d(k+j)} \text{ a.e.}$$

So $\overline{G}'$ fails to invert on a "large" set, and since $\overline{G}'$ was arbitrary, this shows that $G$ is weak one-way. □

## 4.2 Transitivity

**Theorem 4.2.** *If we have a reduction from $F$ to $G$ and from $G$ to $H$, there is a reduction from $F$ to $H$.*

Figure 3: Transitivity diagram

$$\begin{tikzcd}
\Sigma^\ell \arrow[d, "\overline{F}"'] & \Sigma^m \arrow[l, "A'"'] \arrow[d, "\overline{G}"'] & \Sigma^n \arrow[l, "A''"'] \arrow[d, "\overline{H}"'] \\
1^\ell \times \Sigma^* \arrow[r, "B'_m"] & 1^m \times \Sigma^* \arrow[r, "B''_n"] & 1^n \times \Sigma^*
\end{tikzcd}$$

*Proof.* Let $F : \Sigma^* \to \Sigma^*, G : \Sigma^* \to \Sigma^*, H : \Sigma^* \to \Sigma^*$, and assume we have reductions from $F$ to $G$, and from $G$ to $H$. So then for the first reduction, we have the functions $\beta : \mathbb{N} \to \mathbb{N}, A' : \Sigma^* \to \Sigma^*$ where $|A'(x)| = \beta(|x|)$ , and the sequence of functions

$B'_m : 1^* \times \Sigma^* \to 1^m \times \Sigma^*$ where $B'_m(1^\ell, x) = B'(1^m, 1^\ell, x)$ for some function $B'$.

For the second reduction, we have the functions $\alpha : \mathbb{N} \to \mathbb{N}$, $A'' : \Sigma^* \to \Sigma^*$ where $|A''(x)| = \alpha(|x|)$, and the sequence of functions $B''_n : 1^* \times \Sigma^* \to 1^n \times \Sigma^*$ where $B''_n(1^m, x) = B'(1^n, 1^m, x)$ for some function $B''$ and they satisfy all the aforementioned conditions.

Then, claim we have a reduction from $\overline{F}$ to $\overline{H}$, consisting of:

$\gamma$ where $\gamma = \beta \circ \alpha$

$A : \Sigma^* \to \Sigma^* = A' \circ A''$ and the sequence of functions

$B_n : 1^* \times \Sigma^* \to 1^n \times \Sigma^*$ where $B_n(1^\ell, x) = B''_n \circ B'_m(1^\ell, x))$ whenever $m = \alpha(n)$ and $\ell = \beta(m)$.

Now we check that all the conditions are satisfied, and this is a reduction. □

**Claim.** *$\exists e, f \in \mathbb{N}$ s.t $n^e \leq \gamma(n) \leq n^f$ and $|A(x)| = \gamma(|x|)$.*

*Proof.* $\gamma(n) = \beta \circ \alpha(n)$ and by conditions we placed on $\beta$,

$$\alpha(n)^g \leq \beta \circ \alpha(n) \leq \alpha(n)^h$$

for some $g, h > 0$. Then, by the conditions we placed on $\alpha$

$$\alpha(n)^{ig} \leq \beta \circ \alpha(n) \leq \alpha(n)^{jh}$$

for some $i, j > 0$. So we have

$$\alpha(n)^{ig} \leq \gamma(n) \leq \alpha(n)^{jh}$$

Now to see $|A(x)| = \gamma(|x|)$, remember that $A = A' \circ A''$. Then since $|A''(x)| = \alpha(|x|)$ and $|A'(x)| = \beta(|x|)$, we have that $|A(x)| = \beta \circ \alpha(|x|) = \gamma(|x|)$. □

**Claim.** *Condition (1) holds: $A, B$ are polynomial time computable.*

*Proof.* As compositions of polynomial time functions, $A, B$ are clearly also polynomial time computable $\square$

**Claim.** *Condition (2) holds: Whenever $\ell = \gamma(n)$,*
$(\forall x \in \Sigma^n)$, $(\forall y \in \Sigma^*)$
$(x \xrightarrow{\overline{H}} B_n(\langle 1^\ell, y\rangle)) \implies (A(x) \xrightarrow{\overline{F}} \langle 1^\ell, y\rangle)$

*Proof.* Let $\ell, n$ be such that $\ell = \gamma(n)$. Let $x \in \Sigma^n, y \in \Sigma^*$ be such that $x \xrightarrow{\overline{H}} B_n(\langle 1^\ell, y\rangle) = B''_n \circ B'_m(\langle 1^\ell, y\rangle)$ since we know each $B_n$ can be rewritten as $B''_n \circ B'_m$ for some $B''_n, B'_m$. Now let $T = B'_m(\langle 1^\ell, y\rangle)$, so $B_n(\langle 1^\ell, y\rangle) = B''_n(T)$, so $x \xrightarrow{\overline{H}} B''_n(T)$. Since we have a reduction from $G$ to $H$, condition (2) implies that $A''(x) \xrightarrow{\overline{G}} T$ or $A''(x) \xrightarrow{\overline{G}} B'_m(\langle 1^\ell, y\rangle)$ But then, since we have a reduction from $F$ to $G$, we again use condition (2) to see that $A' \circ A''(x) \xrightarrow{\overline{F}} \langle 1^\ell, y\rangle$, or $A(x) \xrightarrow{\overline{F}} \langle 1^\ell, y\rangle$ so condition (2) is preserved. $\square$

**Claim.** *Condition (3) holds: Whenever $\ell = \gamma(n)$,*
$(\exists j \in \mathbb{N})$ *s.t* $(\forall S \subset ran(\overline{F}))$
$\mathbb{P}^n_{\overline{H}}(B_n[S]) \geq \mathbb{P}^\ell_{\overline{F}}(S)/n^j$

*Proof.* Let $T \subset ran(\overline{F})$. Then $B_n[T] \subset \langle 1^n, \Sigma^*\rangle$ and we would like to know what $\mathbb{P}^n_{\overline{H}}(B_n[T])$ is. We know each $B_n$ can be rewritten as $B''_n \circ B'_m$ for some $B''_n, B'_m$.Then it is clear that $B''_n[T'] = B_n[T]$. Let $T' = B'_m[T]$. Since we have a reduction from $G$ to $H$, by condition (3), $\mathbb{P}^n_{\overline{H}}(B_n[T]) = \mathbb{P}^n_{\overline{H}}(B''_n[T']) \geq \mathbb{P}^m_{\overline{G}}(T')/n^c$ for some $c \in \mathbb{N}$. Also, since we have a reduction from $\overline{F}$ to $\overline{G}$, $\mathbb{P}^m_{\overline{G}}(T') = \mathbb{P}^m_{\overline{G}}(B'_m[T]) \geq \mathbb{P}^\ell_{\overline{F}}(T)/n^d$ for some $d \in \mathbb{N}$. Putting it all together, we have

$$\mathbb{P}^n_{\overline{H}}(B[T]) = \mathbb{P}^n_{\overline{H}}(B''[T']) \geq \mathbb{P}^m_{\overline{G}}(T')/n^k = \mathbb{P}^m_{\overline{G}}(B'[T])/n^k \geq \mathbb{P}^\ell_{\overline{F}}(T)/n^{c+d}$$

so condition (3) is preserved. $\square$

Thus the reduction is transitive.

## 4.3 Levin's Universal Function

Our initial motivation for this definition was to capture to essence of Levin's Universal Function and show that every weak one-way function is reducible to Levin's Universal Function under this new notion of reduction. Note that the existence of OWF's would imply Levin's Function is a OWF as well. Levin's Universal Function, $G : \Sigma^* \to \Sigma^*$ is the function that, on input string of length $n$, will interpret first $log(n)$ bits as TM description $\langle M \rangle$, and give the remaining bits as input to $M$, letting it run for $|x|^3$ time, where $x$ is the input to $M$. If $M$ has finished and outputs $y$, then $G$ outputs $(\langle M \rangle, y)$ (Otherwise $G$ outputs something nonsensical which could not be confused for valid output.) Since $|x| = log(n) \leq n$, $G$ is clearly computable in polynomial time.

Analagously to the original proof, we first show that every weak one-way function can be reduced to a weak one-way function that runs in quadratic time. Then we show that arbitrary quadratic time weak one-way functions can be reduced to Levin's OWF. Then, since our reductions are transitive, that shows that every weak OWF is reducible to Levin's OWF, so that Levin's OWF is complete.

**Theorem 4.3.** *Every weak OWF is reducible to a weak OWF that runs in quadratic time.*

*Proof.* In the original proof, Levin's idea is to "pad" the input so that the running time will be quadratic in the size of the new input size. This is enough to show that the existence of a weak OWF implies the existence of a quadratic time OWF, but the idea needs to be subtly modified for our purposes. Let $F : \Sigma^* \to \Sigma^*$ be an arbitrary weak OWF, and computable in time $m^k$ for some $k \in \mathbb{N}$ where $m$ is the input size. If $k \leq 2$ then $F$ is already quadratic time, so assume that $k > 2$. Then I claim the function $F$ is reducible to $G : \Sigma^* \to \Sigma^*$, which drops all but the last $\lfloor n^{2/k} \rfloor$ bits (where $n$ is the input size), runs the remaining bits as input to $F$, then outputs the output of $F$. Furthermore, I claim that $G$ runs in quadratic time. To prove this, we will construct functions $\alpha, A, B_n$ and show that the necessary conditions of the reduction are satisfied.

Let $\alpha(n) = \lfloor n^{2/k} \rfloor$,
$A : \Sigma^* \to \Sigma^*$ be the function that peels off all but the last $\lfloor n^{2/k} \rfloor$ bits, and
$B_n : \langle 1^*, \Sigma^* \rangle \to \langle 1^n, \Sigma^* \rangle$ be the sequence of trivial functions such that $B_n(1^m, y) = \langle 1^n, y \rangle$. Now we check that the conditions for a reduction are satisfied.

**Claim.** *$\exists c, d > 0$ s.t $n^c \leq \alpha(n) \leq n^d$ and $|A(x) = \alpha(|x|)$*

*Proof.* $\alpha(n) = \lfloor n^{2/k} \rfloor$. Since $k > 2$, $n^{2/k} < \alpha(n) < n^1$ □

**Claim.** *$A, B,$ are polynomial time computable.*

*Proof.* Since $A, B$ are both trivial functions, they are clearly polynomial time computable □

**Claim.** *Whenever $m = \alpha(n)$,*
*$(\forall x \in \Sigma^n)$, $(\forall y \in \Sigma^*)$,*
*$[(x \xrightarrow{\overline{G}} B_n(\langle 1^m, y \rangle)) \implies (A(x) \xrightarrow{\overline{F}} \langle 1^m, y \rangle)]$*

*Proof.* Let $m, n$ be such that $m = \alpha(n)$ and assume $x \xrightarrow{\overline{G}} B_n(\langle 1^m, y \rangle)$. Then, since $B_n$ is only changing the number of 1's, $x \xrightarrow{\overline{G}} \langle 1^n, y \rangle$. But by definition, $\overline{G}$ works by removing the padding, or applying $A$ and running $\overline{F}$ on what remains. So then $A(x) \xrightarrow{\overline{F}} \langle 1^m, y \rangle$.

□

**Claim.** *Whenever $m = \alpha(n)$,*
*$(\exists j > 0)$ $(\forall S \subset ran(\overline{F}))$,*
*$\mathbb{P}^n_{\overline{G}}(B_n[S]) \geq \mathbb{P}^m_{\overline{F}}(S)/n^j$*

*Proof.* Let $S \subset \langle 1^m, \Sigma^* \rangle$ and let $\langle 1^m, x \rangle \in S$. Then $B_n[1^m, x] = \langle 1^n, x \rangle$. Then since $\overline{G}$ consists of dropping the padding and running $\overline{F}$, $\mathbb{P}^n_{\overline{G}}(B_n[S]) = \mathbb{P}^m_{\overline{F}}$ □

So a reduction exists. Now,we have that $G$, on input of size $n$, runs in time $\alpha(n)^k$ (since $G$ runs by dropping all but $\alpha(n)$ bits and computing $F$) . But we also have that $\lfloor n^{2/k} \rfloor = \alpha(n)$ and so $n \geq \alpha(n)^{k/2}$ or $\alpha(n)^k \leq n^2$. So $G$ runs in quadratic time, and therefore every weak OWF is reducible to a weak OWF that runs in quadratic time. □

**Theorem 4.4.** *Every quadratic-time weak OWF is reducible to Levin's OWF.*

*Proof.* Let $F : \Sigma^* \to \Sigma^*$ be an arbitrary quadratic-time weak OWF and $G$ be Levin's Universal Function (as described earlier). Then let $\alpha(n) = n - log(n)$, $A : \Sigma^* \to \Sigma^*$ be the function that, on an input of size $n$, simply removes the first $log(n)$ bits from its input (so clearly $|A(x)| = \alpha(|x|)$), and $B_n : \langle 1^*, \Sigma^* \rangle \to \langle 1^n, \Sigma^* \rangle$ be the sequence of functions that, on input $\langle 1^m, w \rangle$, strips off the first $m$ ones, puts back on $n$ 1's, and then appends the TM description that computes $\overline{F}$, $\langle \overline{F} \rangle$, to $w$. As before, we need to verify the conditions necessary for a reduction hold.

**Claim.** $\exists c, d > 0 \ s.t \ n^c \leq \alpha(n) \leq n^d$

*Proof.* $n/2 \leq n - log(n) \leq n.$ □

**Claim.** *Condition (1) holds. $A, B$ are polynomial time computable.*

*Proof.* $A$ is only removing bits and so runs in linear time. $B$ removes bits, then appends at most polynomially many 1's, and then a TM description of $\overline{F}$, which will be of fixed length. So $B$ is computable in polynomial time as well. □

**Claim.** *Condition (2) holds. Whenever $m = \alpha(n)$,*
$(\forall x \in \Sigma^n)$, $(\forall y \in \Sigma^*)$,
$[(x \xrightarrow{\overline{G}} B_n(\langle 1^m, y \rangle)) \implies (A(x) \xrightarrow{\overline{F}} \langle 1^m, y \rangle)]$

*Proof.* Let $m, n$ be such that $\alpha(n) = m$. Assume $x \xrightarrow{\overline{G}} B_n(\langle 1^m, y \rangle)$. Note that this implies that the TM described by the first $log(n)$ bits of $x$ halts when run on the remaining bits. Then, since $B_n$ is only changing the number of 1's and appending a TM description, $x \xrightarrow{\overline{G}} \langle 1^n, \langle \overline{F} \rangle, y \rangle$. But since the output of $\overline{G}$ indicates the TM description used, we know the first $log(n)$ bits, $\langle M \rangle$, are exactly the bits of $\langle \overline{F} \rangle$. Then, since $\overline{G}$ works by applying $A$ and running $\overline{F}$ on what remains, $A(x) \xrightarrow{\overline{F}} \langle 1^m, y \rangle$. □

**Claim.** *Condition (3) holds. Whenever* $m = \alpha(n)$,
$(\exists j > 0)\ (\forall S \subset ran(\overline{F}))$,
$\mathbb{P}^n_{\overline{G}}(B_n[S]) \geq \mathbb{P}^m_{\overline{F}}(S)/n^j$

*Proof.* Let $m, n$ be such that $\alpha(n) = m$ and let $S = \langle 1^m, y \rangle \subset \langle 1^m, \Sigma^* \rangle$. Consider $B_n[S] = \langle 1^n, \langle \overline{F} \rangle, y \rangle$. By the definition of $\overline{G}$, this set will be mapped to only by strings whose first $log(n)$ bits correspond to the machine description of $\overline{F}$ and whose remaining bits are mapped by $TM_x$ to $y$. The probabililty that the first $log(n)$ bits, $\langle M \rangle$, corresponds to the correct TM description is $2^{-log(n)} = 1/n$. So $\mathbb{P}^m_{\overline{G}}(B_n[S]) = \mathbb{P}^n_{\overline{F}}(S)/n$ □

This shows that every quadratic-time computable weak OWF can be reduced to Levin's Universal Function. □

**Theorem 4.5.** *Every weak OWF is reducible to Levin's OWF*

*Proof.* This is clear from the previous two theorems, and the transitivity of reductions. □

## 5 Open Questions

We hope this formalization of reductions for one-way functions will stimulate new interest in looking for other complete one-way functions, much as searching for NP-complete problems has been a huge part of complexity theory. In particular, it would be nice to have a more "natural" complete one-way function, such as SAT is for decision problems. We would also like to see if this definition could be extended (perhaps with slight modifications) to similar types of problems, perhaps including the class of collision resistant functions.

## References

[1] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[2] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[3] L. A. Levin. The tale of one-way functions. *Probl. Inf. Transm.*, 39(1):92–103, January 2003.

[4] C. Papadimitriou and Mihalis Yannakakis. Optimization, approximization and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[5] Alan L. Selman. A survey of one-way functions in complexity theory. *Mathematical systems theory*, 25(3):203–221, 1992.

[6] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.