We discuss an alternative view of the proof of the Goldreich-Levin Theorem.

## 1 Error-Correcting Codes: The Walsh-Hadamard Code

An error-correcting code is a function $E : \{0, 1\}^n \rightarrow \{0, 1\}^m$ that maps $n$-bit strings into $m$-bit codewords. A codeword $E(x)$ should contain redundant information about $x$, in the sense that even if we have a string $\hat{E}(x)$ corrupted at a few locations, we are still able to recover the original string $x$ from $\hat{E}(x)$.

For two string $x, r \in \{0, 1\}^n$, we define $x \odot r = \sum_{j=1}^{n} x_j r_j \pmod{2}$. The Walsh-Hadamard code (WH) is an exponentially long error-correcting code: each input string $x$ is mapped to a $2^n$-bit string $\text{WH}(x)$, where the $i$-th bit of $\text{WH}(x)$ is given by $x \odot r$, where $r$ is the $i$-th $n$-bit string.

Given $\text{WH}(x)$, a corrupted codeword, we would like to recover $x$. Remember that $\text{WH}(x)$ is a $2^n$-bit string, and we would like to have an efficient decoding procedure to obtain the original string $x$. Hence we assume that we have only local access to a polynomial number of bit positions of the corrupted codeword, and we would like to recover the correct $x$ with high probability.

Finally, if $\text{WH}(x)$ is corrupted on more than a $1/4$ fraction of bit positions, it may be the case that the decoding of $\text{WH}(x)$ is not necessarily unique. In this case we are happy if the decoding procedure outputs in polynomial time (and with good probability) a list $L$ of candidate inputs $x$ for this corrupted codeword (this list may also contain irrelevant strings; in applications we usually have external information that allow us to narrow the list). It is possible to show that any corrupted codeword is $(1/2 + 1/p(n))-close$ to at most a polynomial number of correct codewords (i.e, the original list $L$ is not very large).

## 2 The Goldreich-Levin Theorem

The proof of the Goldreich-Levin theorem is equivalent to the existence of an efficient local list decoding procedure for the Walsh-Hadamard code. We try to explain this connection in this section.

**Theorem 1** (Goldreich-Levin). Suppose that $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a one-way function such that $f$ is one-to-one and $|f(x)| = |x|$ for every string $x \in \{0, 1\}^*$. Then for every probabilistic polynomial-time algorithm $A$ there is a negligible function $\epsilon : \mathbb{N} \rightarrow [0, 1]$ such that

$$\Pr_{x,r \in \{0,1\}^n} [A(f(x),r) = x \odot r] \leq 1/2 + \epsilon(n),$$

where $x \odot r$ is defined as before.
Remember that to prove this result we assume the existence of an algorithm \( A \) such that:
\[
\Pr_{x,r \in \{0,1\}^n} [A(f(x), r) = x \oplus r] \geq 1/2 + 2/p(n)
\]
ininitely often (of course using \( 2/p(n) \) instead of \( 1/p(n) \) is inessential, since \( p(n) \) is arbitrary), and use this assumption to prove the existence of an algorithm \( B \) that is able to invert \( f \) with noticeable probability. By an averaging argument, there exist at least an \( 1/p(n) \) fraction of the \( x \)'s such that:
\[
\Pr_{r \in \{0,1\}^n} [A(y, r) = x \oplus r] \geq 1/2 + 1/p(n),
\]
where \( y = f(x) \).

In other words, for the good input strings \( x \)'s we can view the sequence of output bits of \( A(f(x), r) \) for all \( r \in \{0,1\}^n \) as a corrupted version of the Walsh-Hadamard codeword \( WH(x) \). The Goldreich-Levin algorithm uses \( A(y,.) \) as a black-box, i.e., it makes local queries to this corrupted codeword and obtain a list \( L \) of candidate inputs that invert \( y = f(x) \). We know that with noticeable probability the original \( x \) will be on the list. The algorithm uses the fact that \( f \) is efficiently computable to test if some \( x \) in \( L \) satisfies \( y = f(x) \).

References