Comments about Polar Codes:

1) Polar Codes give an alternate proof of
   Shannon's Theorem for Symmetric Channels.

   NB: Polar Codes are linear.

2) Method has been generalized to prime
   alphabets. (and maybe also general ones?)

   Source Coding:

   Example: Take $Y = \text{null}$

   $X_0^{N-1} \rightarrow M_n X_0^{N-1} = U_0^{N-1}$

   but $H(U_i | U_0^{i-1}, Y_0^{N-1}) \rightarrow 0$ (for most indices)

   $\sum_{i=0}^{N-1} H(U_i | U_0^{i-1}) = H(U_0^{N-1}) = H(X_0^{N-1}) = N \cdot H(X)$.

   Only reveal $U_i$ for which $H(U_i | U_0^{i-1})$ is not $\approx 0$ \Rightarrow We get Compression $\approx n \cdot H(X)$.

   \Rightarrow Another proof for the Source Coding Theorem.
3) Which indices polarize to 0?

\[ \sum H(U_i | U_0^{i-1}) = N \cdot H(X) \]

i.e., find the "Frozen bits".

It might "feel" reasonable that \( H(U_i | U_0^{i-1}) \)

is \( \approx 1 \) for small \( i \) and \( \approx 0 \) for large \( i \).

* But life is not so simple.

* There is an algorithm to figure out

which bits to freeze.

4) Polar Codes are Versatile!

Useful for Slepian-Wolf, Wyner-Ziv,
Gelfand-Finsker, etc.

References: Arikan's original paper,
Şasoglu's thesis (Chap 2)
Upcoming writeup with P. Xia.
Moser's "entropy compression" argument.

Of course, if \( \Pr(\overline{E_i}) \) is small

\[ \Rightarrow \Pr(U \overline{E_i}) \text{ is small by union bound.} \]

\[ \Rightarrow \bigcap E_i \text{ can happen.} \]

Lovasz Local Lemma (LLL) gives another criterion. (under limited dependence)

Boolean

**k-SAT**: \( n \) var's \( x_1, \ldots, x_n \), \( m \) clauses \( c_1, \ldots, c_m \)

\[ c_i : x_{i_1} \lor x_{i_2} \lor \cdots \lor x_{i_k} \]

(possibly with negations)

Q: Given a k-SAT instance, is there a way of assigning \( x_i \) s.t. all clauses are satisfied?

Sufficient Condition for Satisfiability of k-SAT:

1) Clauses are disjoint. (don't share var's)
2) If clauses overlap a lot, may not be satisfiable.

(e.g., take all $2^k$ clauses) over $k$ bits.

Theorem: Suppose an instance of $k$-SAT where each clause overlaps with $\leq \frac{k-c}{2}$ clauses is always satisfiable. (for some constant $c$) ($c=3$ works).

Algorithmic Proof (Moser 2009)

1) Pick a random assignment to the $n$ vars.

If all satisfied, done!

else let $T$ be the set of unsat clauses.

2) For each clause set $S$, Fix($S$).

Fix($S$) 1) If $A$ satisfies $S$, do nothing.

2) Replace the bits of $A$ on support of $S$ by $k$ random bits.

3) Find all clauses $S'$ which overlap with $S$ & which the new $A$ violates. Call the set $B$. 
(s may be still in B).

For each $S' \in B$ (in some fixed order),

$\text{Fix}(S')$.

* Claim: If $\text{fix}(S)$ terminates, then
  the new $A$ satisfies $S$, and
  it will continue the old
  if $S$ was satisfied by $A$ before
  $\text{fix}(S)$ was called, then the
  new $A$ will also satisfy $S$.

$(\Rightarrow$ each application of $\text{fix}(\cdot)$ decreases
the # of unsat clauses $)$.

Pf: Obvious!

* Corollary: If the main algo. terminates,
  the instance was satisfiable ($A$
  being a satisfying assignment).

* All that remains is to prove $\text{fix}(\cdot)$
terminates early enough.
Idea: (Entropy Compression)

Give the alg., a long tape $R$ of random bits.

Each step consumes about $k$ bits of randomness.

Q: Given $A'$ and $R'$, can you recover $(A, R)$?

$$A \rightarrow \boxed{\text{Step 2 of } \text{fix}()} \rightarrow A'$$

$$R \rightarrow \text{Step 2 of } \text{fix}() \rightarrow R' \text{ (rest of the tape)}$$

But if we knew which $S$ was being fixed, we could reverse the process.

(since only a unique assignment fails to satisfy $S$).

* Now imagine a log file where you record the sequence of clauses on which fix is called:

Step 2 of Fix(): Suppose the alg. runs for more than $M$ runs of Fix(): In $M$ calls of Fix() (Step 2 of)

We consume $|R| = Mk$ bits of randomness.
Suppose the information in the log and $A'$ (and $R' = \emptyset$ in the end) can be used to recover $(A, R)$.

A Naive coding of history takes $M(\log m)$ bits. To get an "impossible compression" we need $M(\log m) < M \cdot k$

$\Rightarrow m < 2^k$. Not so good! (true but trivial to prove)

More clever way?

Knowing $s$, specifying $s'$ needs only $k-c$ bits!

Big insight: Start by using $m \log m$ bit to encode clauses in $T$ in the start. For other clauses $s'$ called from $\text{fix}(s)$, record $s'$ via $k-c$ bits which encodes neighbors of $s$. 
* We also need termination symbols when Fix() returns.

\[ |H'| = O(m \log m) + M(k - C) + O(1) \]

\[ \Rightarrow \text{Total size of } \log = \]

\[ n + M k \text{ bits} \leq n + O(m \log m) + M(k - C + O(1)) \]

\[ \Rightarrow \text{We compress random } (A+R) \text{ to } (A', H') \]

\[ \Rightarrow \text{We get an upper bound on } M. \]