6.897 Algorithmic Introduction to Coding Theory	October 24, 2001
Lecture 11	
Lecturer: Madhu Sudan	*Scribe: Matt Lepinski*

Today we will talk about:

1. Abstraction of the Decoding Algorithm for Reed-Solomon Codes
2. Decoding Concatenated Codes (specifically, the Forney Codes).

1 Abstraction of Reed-Solomon Decoding Algorithm

Our first goal today is to give a very abstract view of the Welch-Berlekamp decoding algorithm for Reed-Solomon codes. This abstraction will allow us to see its generality, and thus apply it to other families of error-correcting codes. The algorithm we describe here is from the works of Pellikaan [9], Kötter [6], and Duursma [1].

1.1 Reed-Solomon Decoding Review

Recall the Reed-Solomon decoding algorithm from last lecture. Here we have n distinct points $x_1, \ldots, x_n \in \mathbb{F}_q$ given implicitly, and we are explicitly given as input elements $y_1, \ldots, y_n \in \mathbb{F}_q$. The decoding algorithm consists of the following two steps:

1. FIND: Polynomials $a(x)$ and $b(x)$ such that:
   - For all $i \in [n]$, $a(x_i)y_i = b(x_i)$.
   - Degree of a is small (at most t).
   - Degree of b is small (less than $k + t$).
2. OUTPUT: $\frac{b(x)}{a(x)}$

1.2 Special Structure of Reed-Solomon Codes

The above algorithm and its proof of correctness used the properties of Reed-Solomon codes in several ways. Below we list the different aspects that seem specific to Reed-Solomon codes.

1. We used the fact that the indices of the codewords (i.e., $x_1, \ldots, x_n$) are field elements.
2. We used the fact that two low degree polynomials cannot agree in very many places, several times (in the construction of the code, as well as the analysis of the decoding algorithm).
3. We used the fact that multiplication of two low degree polynomials is a low degree polynomial.
4. We also used the fact that under the right conditions, the ratio of two polynomials is a polynomial.

Not very many of these above facts were really critical to the proof — they were just the simplest way to get to the end. Some of the above properties (like (2)) are just facts that hold for any error-correcting code. Others, in particular (3), are somewhat special, but can still be abstracted with care.

1.3 Multiplication of vectors

One of the critical operations in the decoding algorithm is that of multiplying the error-locator polynomial $a(x)$ with the message polynomial $p(x)$ and considering the evaluations of the product at $x_1, \dots, x_n$. This is essentially a coordinatewise product of vectors, defined below.

Definition 1 *For $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$, their coordinatewise product, denoted $\mathbf{u} \star \mathbf{v}$, is given by*

$$\mathbf{u} \star \mathbf{v} = \langle u_1 v_1, \dots, u_n v_n \rangle.$$

If the coordinatewise product is a strange operation in linear algebra, then the following product of sets is even stranger, but is critical to the working of the Reed-Solomon decoding algorithm.

Definition 2 *For $U, V \subseteq \mathbb{F}_q^n$ their product, denoted $U \star V$ is given by*

$$U \star V = \{\mathbf{u} \star \mathbf{v} | \mathbf{u} \in U, \mathbf{v} \in V\}.$$

Why are these operations interesting? To motivate them, let us look back at the RS decoding algorithm:

- Let U be the set of vectors obtained by evaluations of polynomials of degree at most t.
- Let V be the set of vectors obtained by evaluations of polynomials of degree less than k.
- Then $U \star V$ is the set of evaluations of polynomials of degree less than $k+t$ that factor into a polynomial of degree less than k and a polynomial of degree at most t.
- In particular $U \star V$ is a subset of the set of evaluations of polynomials of degree less than $k+t$.
- To see that this is special, note that U is a vector space of dimension $t+1$ and V is a vector space of dimension k. What we have noticed is that their product is contained in a vector space of dimension $k+t$. This is *very* special. In general if we take two arbitrary vector spaces of dimension k and t, their product would not be contained in any vector space of dimension less than kt — so polynomials end up being very special!

In what follows we will show that this is the *only* speciality of polynomial based codes. We will show that if any code ends up having nice properties with respect to product with some other codes, then it can be decoded.

1.4 Error-Locating Pairs

Let C be a $[n, k, ?]_q$ code and suppose, we wish to decode up to e errors with C. The following definition describes a simple combinatorial object, whose existence suffices to give a decoding algorithm for C.

Definition 3 *A pair of linear codes (A, B), with $A, B \subseteq \mathbb{F}_q^n$, form an* e-error-correcting pair *for a linear code $C \subseteq \mathbb{F}_q^n$ if they satisfy the following conditions:*

1. *$A \star C \subseteq B$.*
2. *The dimension of A is sufficiently large: Specifically,* $\dim(A) > e$.
3. *The minimum distance of B is sufficiently large: Specifically, $\Delta(B) > e$.*
4. *The minimum distance of C is sufficiently large: $\Delta(C) > n - \Delta(A)$.*

1.5 The Generalized Algorithm

In this algorithm, we assume that we are given generator matrices for linear codes A, B, and C, where $(A, B$ form an e-error-correcting pair for C.

Abstract decoding algorithm

Given: Matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ generating codes $A, B, C \subseteq \mathbb{F}_q^n$, with (A, B) forming e-error-correcting pair for C. Received vector $\mathbf{y} \in \mathbb{F}_q^n$.

Step 1: Find $\mathbf{a} \in A$ and $\mathbf{b} \in B$ such that $\mathbf{a} \star \mathbf{y} = \mathbf{b}$ and $(\mathbf{a}, \mathbf{b}) \neq (\mathbf{0}, \mathbf{0})$, if such a pair exists.

Step 2: Compute $\mathbf{z} \in (\mathbb{F}_q \cup \{?\})^n$ as follows: If $a_i = 0$, then $z_i =?$ else $z_i = y_i$.

Step 3: Output the result of performing erasure decoding on $\mathbf{z}$ for code C, if this results in a unique codeword.

As usual we argue efficiency first and correctness later. Efficiency is obvious for Step 2. For Step 3, recall that we observed in the last lecture that erasure-decoding can be done in $O(n^3)$ time for every linear code. So it suffices to argue efficiency of Step 1. We do so by claiming this is a task of finding a solution to a linear system. Note that we are searching for unknowns $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{F}_q$. The condition that $\mathbf{a} \in A$ places linear constraints on $a_1, \ldots, a_n$. Similarly, the condition $\mathbf{b} \in B$ turns into linear constraints on $b_1, \ldots, b_n$. Also, the constraints $a_i y_i = b_i$ is also linear in a_i and b_i (since y_i is fixed). Lastly, the condition $(\mathbf{a}, \mathbf{b}) \neq (\mathbf{0}, \mathbf{0})$ is just asking for a non-zero solution to this linear system. So the task at hand is that of finding a non-trivial solution to a homogenous linear system. Again this can be done efficiently, in $O(n^3)$ time. It remains to prove correctness of the above algorithm, and we do so next.

1.6 Correctness

The proof of correctness goes through the usual steps. We assume below that there exists a codeword $\mathbf{c} \in C$ that is close to (within a Hamming distance of e of) the received vector $\mathbf{y}$. We fix this codeword and use it to argue correctness.

Below we argue that a solution pair $(\mathbf{a}, \mathbf{b})$ to Step 1 does exist. We argue that any solution pair $(\mathbf{a}', \mathbf{b}')$ to Step 1 satisfies $\mathbf{a}' \star \mathbf{c} = \mathbf{b}'$. Next we show that for any pair $\mathbf{a}' \in A$ and $\mathbf{b}' \in B$ there is at most one $\mathbf{c}' \in C$ such that $\mathbf{a}' \star \mathbf{c}' = \mathbf{b}'$. The correctness follows by noticing that the solution $\mathbf{c}'$ output by the algorithm satisfies $\mathbf{a}' \star \mathbf{c}' = \mathbf{b}'$, if $(\mathbf{a}', \mathbf{b}')$ is the solution found by Step 1. Details below.

Claim 4 *There exists a pair $(\mathbf{a}, \mathbf{b})$ as required in Step 1 exists. Furthermore they satisfy $\mathbf{a} \star \mathbf{c} = \mathbf{b}$.*

Proof By peeking back at the analogous claim in the Reed-Solomon decoder, we realize we want $\mathbf{a}$ to be the "error-locator", i.e., satisfying $a_i = 0$ if $y_i \neq c_i$. Can we find such a vector that is non-zero? Turns out we can, if we know A has sufficiently large dimension. In particular, the constraints $a_i = 0$ give e homogenous linear constraints on $a_1, \ldots, a_n$. Since $\mathbf{A}$ has dimension $e+1$ or larger (Condition (2) of the definition of an error-correcting pair), it contains a non-zero vector that satisfies all these constraints!

Now take $\mathbf{a}$ to be any non-zero vector satisfying $a_i = 0$ if $c_i \neq y_i$. Take $\mathbf{b}$ to be the vector $\mathbf{a} \star \mathbf{c}$. Note that $\mathbf{b} \in B$ since $A \star C \subseteq B$. Furthermore, for every i, we have either $c_i = y_i$ and so $b_i = a_i c_i = a_i y_i$, or we have $a_i = 0$ and hence $b_i = a_i c_i = 0 = a_i y_i$. Furthermore the pair is non-zero since $\mathbf{a} \neq \mathbf{0}$. This concludes the proof of the claim. ■

Claim 5 *If $\mathbf{a}', \mathbf{b}'$ are any pair of solutions to Step 1, then $\mathbf{a}' \star \mathbf{c} = \mathbf{b}'$.*

Proof Since $\mathbf{a}', \mathbf{b}'$ are outputs of Step 1, they satisfy the condition, $\mathbf{a}' \star \mathbf{y} = \mathbf{b}'$. Let $\mathbf{a}' \star \mathbf{c} = \mathbf{b}^*$. To prove the claim we need to show $\mathbf{b}' = \mathbf{b}^*$. Since $A \star C \subseteq B$, we know that $\mathbf{b}^* \in B$. Therefore, $\mathbf{b}'$ and $\mathbf{b}^*$ are two codewords of B that agree on every coordinate i for which $y_i = c_i$ ($b'_i = a_i y_i$ and $b^*_i = a_i c_i$). But $y_i = c_i$ on at least $n - e$ coordinates, and so $\Delta(\mathbf{b}', \mathbf{b}^*) \leq e$. But $\Delta(B) > e$ (Condition (3) of the definition of an error-correcting pair), implying $\mathbf{b}' = \mathbf{b}^*$ as required. ■

Claim 6 *For any $(\mathbf{a}', \mathbf{b}') \in A \times B - \{(\mathbf{0}, \mathbf{0})\}$ there exists at most one $\mathbf{c} \in C$ such that $\mathbf{a}' \star \mathbf{c} = \mathbf{b}'$.*

Proof Let $\mathbf{c}, \mathbf{c}' \in C$ satisfy $\mathbf{a}' \star \mathbf{c} = \mathbf{b}' = \mathbf{a}' \star \mathbf{c}'$. First, let us note that we actually have $\mathbf{a}' \neq \mathbf{0}$. (If not, then $\mathbf{b}' = \mathbf{a}' \star \mathbf{c}'$ would also be $\mathbf{0}$ and this contradicts the condition that together then are non-zero.)

To prove the claim, we wish to show $\mathbf{c} = \mathbf{c}'$. Since both are codewords of C, it suffices to show that $\Delta(\mathbf{c}, \mathbf{c}') < \Delta(C)$. But note that $c_i = c'_i$ for every i, where $a'_i \neq 0$. Further, since $\mathbf{a}' \neq \mathbf{0}$, we have $a'_i \neq 0$ on at least $\Delta(A)$ coordinates. Thus we have, $\Delta(\mathbf{c}, \mathbf{c}') \leq n - \Delta(A)$. But Condition (4) in the definition of an error-locating pair ensures that $n - \Delta(A) < \Delta(C)$. Thus we get $\Delta(\mathbf{c}, \mathbf{c}') < \Delta(C)$ as desired. ■

We can now formally prove the correctness of the decoding algorithm.

Lemma 7 *If (A, B) form an e-error-correcting pair for C and $\mathbf{c} \in C$ and $\mathbf{y} \in \mathbb{F}_q^n$ satisfy $\Delta(\mathbf{c}, \mathbf{y}) \leq e$, then the* **Abstract decoding algorithm** *outputs $\mathbf{c}$ on input $A, B.C$, and $\mathbf{y}$.*

Proof By Claim 4, we have that there exists a pair $\mathbf{a}, \mathbf{b}$ satisfying the conditions of Step 1. So some such pair $\mathbf{a}', \mathbf{b}'$ will be found. By Claim 5, $\mathbf{c}$ will satisfy $\mathbf{a}' \star \mathbf{c} = \mathbf{b}'$. Since, $\mathbf{a}' \star \mathbf{c} = \mathbf{b}' = \mathbf{a}' \star \mathbf{y}$, we have $c_i = y_i$ whenever $a_i \neq 0$ and thus c_i is a valid solution to Step 3 of the algorithm. To conclude, we need to ensure that $\mathbf{c}$ is the only solution to Step 3 of the algorithm. But this is also clear, since any solution $\mathbf{c}'$ to this step must satisfy $a'_i c'_i = a'_i y_i = b'_i$ for every i, and thus must be a codeword of C satisfying $\mathbf{a}' \star \mathbf{c}' = \mathbf{b}'$ and by Claim 6, $\mathbf{c}$ is the unique vector with this property. ■

We conclude with the following theorem:

Theorem 8 *Any code C that has an e-error-correcting pair has an efficient ($O(n^3)$ time) algorithm solving the bounded distance decoding problem for up to e errors.*

1.7 Applications

Exercise: Verify that the **Welch-Berlekamp** algorithm from the last lecture is an instantiation of the **Abstract decoding algorithm** given today.

We now move on to a more interesting application. Recall the construction of algebraic-geometry codes. (To be more precise, recall that we know very little about them to recall much.)

Algebraic-geometry codes. These codes were constructed by finding n points in $\mathbb{F}_q^m$ and evaluating all polynomials of "order" at most ℓ at all n places. The following properties of order were used in asserting that these gave good codes:

1. There exists an integer g such that for every ℓ, the evaluations of polynomials of order at most ℓ formed a subspace of $\mathbb{F}_q^n$ of dimension at least $\ell - g + 1$.
2. Two distinct polynomials of order at most ℓ can agree on at most ℓ out of the n evaluation points.
3. The product of two polynomials of order ℓ_1 and ℓ_2 has order at most $\ell_1 + \ell_2$.

These properties suffice to prove that codes obtained by the evaluation of polynomials of order at most $n-d+1$ give an $[n,k,d]_q$ code for some $k \geq n-d-g+1$. As we see below, the same properties also give us an $\lfloor \frac{n-g-\ell-1}{2} \rfloor$-error-correcting pair for these codes.

Lemma 9 *If C is an algebraic-geometry code obtained by evaluating polynomials of order at most ℓ, then it has an $\lfloor \frac{n-g-\ell-1}{2} \rfloor$-error-correcting pair.*

Proof Let $e \leq \frac{n-g-\ell-1}{2}$. Below all references to the "Conditions" are to the four conditions in the definition of an e-error-correcting pair.

To get an e-error-correcting pair, we need $\dim(A) > e$ (to satisfy Condition 2). We will pick A to be the algebraic-geometry code obtained by evaluations of all polynomials of order at most $e+g$. Since we need $A \star C \subseteq B$, we pick B to be the algebraic-geometry code obtained by all evaluations of polynomials of order at most $e+g+\ell$, and thus satisfy Condition 1. To satisfy Condition 3, we need $\Delta(B) > e$. We know from the properties of algebraic-geometry codes, that B has distance at least $n-e-g-\ell$. From the choice of e, it follows that $n-e-g-\ell > e$. Finally, to get Condition 4, we need to verify that $\Delta(A)+\Delta(C) > n$. Since $\Delta(A) \geq n-e-g$ and $\Delta(C) \geq n-\ell$, this amount to verifying that $2n-\ell-e-g > n$ which is equivalent to verifying that $e < n-\ell-g$. But in fact e is less than half the RHS. Thus (A,B) as chosen above give an e-error-correcting code for C. ∎

As a corollary we see that we get a pretty decent decoding algorithm for algebraic-geometry codes, correcting about $(n-\ell-g)/2$ errors. However, it does not decode up to half the minimum distance, since the distance is $n-\ell$ and we are only correcting $(n-\ell-g)/2$ errors. Later in the course we will see a better algorithm.

Before concluding this section, we mention one more case where the abstract decoding algorithm provides an inspiration for a decoding algorithm. This is the case of the Chinese Remainder Codes, described next.

Chinese Remainder Codes. The Chinese Remainder Codes are number-theoretic codes defined as follows:

- Fix primes $p_1, \ldots, p_n$ such that $p_1 < p_2 < \cdots < p_n$
- A *message* is an integer $m \in [0 \ldots K-1]$ where $K = \prod_{i=1}^{k} p_i$.
- The encoding of m is the vector $\langle m(\mod p_1), \ldots, m(\mod p_n)\rangle$.

By the Chinese Remainder Theorem, residues of m modulo *any* k of the n primes, suffices to specify m. Thus specifying its residue modulo n primes, makes the above a redundant encoding. This is not one of our usual algebraic codes — in fact it is not even linear! However, one can apply our usual notions of distance, error-detection and correction to this code. We note the code has distance $n-k+1$ (since specifying the residues modulo k primes, specifies the message). So one can as the question - is it possible to correct $(n-k)/2$ errors. The first decoding algorithm was given by Mandelbaum [8]. Later work of Goldreich, Ron, and Sudan [2] showed how to interpret this algorithm as a relative of the abstract decoding algorithm given here. Turns out both algorithms correct slightly less than $(n-k)/2$ errors in polynomial time — this was fixed later by Guruswami, Sahai, and Sudan [3] using the algorithm of [2] in combination with an algorithm known as the "Generalized Minimum Distance Algorithm" that we will talk about in the next section.

2 Decoding Concatenated Codes

We now move on to an elegant solution for the unambiguous decoding problem for some families of concatenated codes. Let us start by recalling concatenated codes.

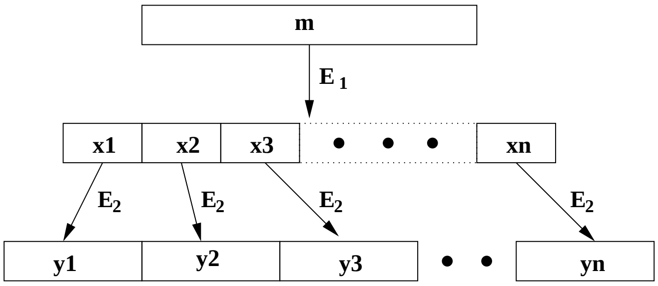

Figure 1: Encoding in concatenated codes

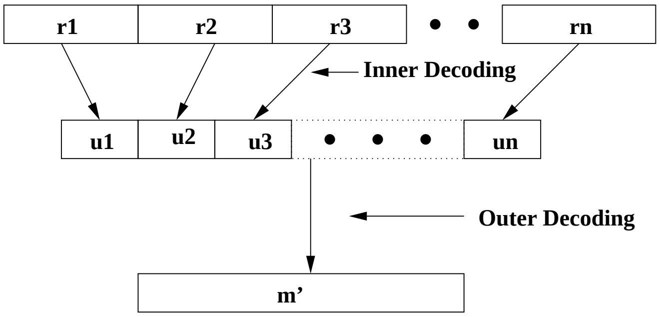

Figure 2: Decoding concatenated codes

Concatenated Codes. Given an $[n, k, d]_Q$ outer code C_1 with encoding function E_1 and an $[n_2, k_2, d_2]_q$ inner code C_2, with encoding function E_2, where $Q = q^{k_2}$, their concatenation, denoted $C_1 \diamond C_2$, is the $[nn_2, kk_2, dd_2]_q$ code obtained as follows: Start with a message $\mathbf{m} \in \mathbb{F}_Q^k$ and encode it using E_1 to get a vector $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$, where $x_i \in \mathbb{F}_Q$. Now, viewing the x_i's as elements of $\mathbb{F}_q^{k_2}$ encode them using E_2 to get $\mathbf{y} = \langle \mathbf{y}_1, \ldots, \mathbf{y}_n \rangle$ where $\mathbf{y}_i \in \mathbb{F}_q^{n_2}$ ad the encoding of $\mathbf{m}$. (See also Figure 1.)

Getting some reasonable algorithms for decoding concatenated codes is not so hard, under some reasonable assumptions. Such an algorithm would take a received vector $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_n \rangle$, where $\mathbf{r}_i \in \mathbb{F}_q^{n_2}$ and decode them in two steps, inverting the encoding steps. So first it would decode the $\mathbf{r}_i$'s individually to their nearest codewords. For $i \in [n]$, let $E_2(u_i)$ be the codeword of C_2 nearest to $\mathbf{y}_i$, where we view $u_i \in \mathbb{F}_Q$ as an element of $\mathbb{F}_q^{k_2}$. Now we treat $\mathbf{u} = \langle u_1, \ldots, u_n \rangle$ as a corrupted codeword of C_1 and decode it using a decoding algorithm for E_1. (See Figure 2.)

First, let us note that for the concatenated codes we have considered so far (Forney codes and Justesen codes), the above is actually efficient. Recall that in these applications the outer code is a well-studied code such as the Reed-Solomon code with efficient decoding algorithms. On the other hand the inner code is not well-understood — the only thing we know about it is that it has good minimum distance properties. So it is not reasonable to expect a sophisticated decoding algorithm for the inner code. But then the inner codes are so small the brute-force decoding only takes polynomial time in the length of the concatenated code, so we don't *need* a sophisticated inner decoding algorithm. So the entire decoding process described above takes time polynomial in the length of the concatenated code, for the typical codes.

However, this doesn't give us an algorithm decoding upto half the minimum distance of the concatenated code! It may only decode $\frac{dd_2}{4}$ errors. We won't prove that it can correct so many errors, or that it can't correct more. But to see a plausibility argument, note that to get a decoding failure, the adversary only has to ensure $d/2$ of the symbols $u_i \neq x_i$ and to get any such decoding error for the inner decoder it may only need to flip $d_2/2$ symbols of the inner alphabet. Thus a total of $dd_2/4$ errors may lead to a decoding failure. Below we will show a clever algorithm which gets around this problem with relatively little extra information about the outer decoding algorithm. This algorithm is the **Generalized Minimum Distance (GMD) Decoding Algorithm**, due to Forney [5] (see also [4]).

Exercise: Prove that the decoding algorithm outlined above does indeed correct at least $(d-1)(d_2-1)/4$ errors.

2.1 Decoding errors and erasures

Let us start by looking at the decoding algorithm we have for the outer code. We already seem to be making full use of it — we assume it can correct $(d-1)/2$ errors, and it can't possibly correct more errors unambiguously. Turns out, there is one additional feature of the decoding algorithm for the outer code that comes in quite handy. This is the feature that it can deal with erasures quite naturally and benefit from it.

Proposition 10 *Let C be an $[n, n-d+1, d]_q$ Reed-Solomon code. Suppose $\mathbf{r} \in (\mathbb{F}_q \cup \{?\})^n$ is a vector derived from a codeword $\mathbf{c} \in C$ by s erasures and t errors. I.e., $|\{i | r_i = ?\}| = s$ and $|\{i | c_i \neq r_i \text{ and } r_i \neq ?\}| = t$. Then $\mathbf{c}$ can computed efficiently given $\mathbf{r}$, s, t provided $s + 2t < d$.*

Proof The proposition is straightforward, given the observation that the code C', obtained by puncturing C on the coordinates where there are erasures, is an $[n-s, n-d+1, d-s]$ Reed-Solomon code and thus $(d-s-1)/2$ errors can be corrected efficiently. ■

How can we use the option to declare erasures? In the process of decoding the inner code, we ignored some obvious information that was available to us. We not only could find out the encoding of which message word u_i was closest to the ith received block $\mathbf{r}_i$, but we also know how many errors have (potentially) occurred in each block. Somehow we should make use of this information. A natural idea is to to declare blocks with large number of errors to be erasures. This works out roughly correct. We will see that a simple probabilistic interpretation of the distances give a right strategy for declaring erasures.

2.2 A randomized decoding algorithm

Let us fix some notation, that we have been introducing as we went along. The message vector is $\mathbf{m}$, its encoding under the outer code is $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$, where $x_i \in \mathbb{F}_Q \cong \mathbb{F}_q^{k_2}$. The encoding of x_i under the inner code is $\mathbf{y}_i$ and thus the final codeword is $\mathbf{y} = \langle \mathbf{y}_1, \ldots, \mathbf{y}_n \rangle$. The noisy channel corrupts $\mathbf{y}$ and a vector $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_n \rangle$ is received. We now describe the decoding algorithm for this code.

Random-Concat-Decoder

Given: $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_n \rangle$.

Step 1: For $i \in [n]$, compute u_i that minimizes $\Delta(E_2(u_i), \mathbf{r}_i)$.

Step 2: Set $e_i' = \min\{d_2/2, \Delta(E_2(u_i).\mathbf{r}_i)$.

Step 3: For every $i \in [n]$ repeat the following: With probability $2e_i'/d_2$, set $v_i = ?$, else set $v_i = u_i$.

Step 4: Perform errors and erasures decoding of the vector $\mathbf{v} = \langle v_1, \ldots, v_n \rangle$.

Step 5: If a vector $\mathbf{m}' \in \mathbb{F}_Q^k$ is obtained in Step 4, check to see if $\Delta(E_2 \diamond E_1(\mathbf{m}), \mathbf{r}) \leq (dd_2)/2$ and if so output $\mathbf{m}'$.

The above algorithm clearly works in time that is $O(nn_2Q + T(n))$, where $T(n)$ is the time taken by the errors and erasures decoding algorithm. The main issue is how many errors it corrects. We will claim that it "essentially" solves the unambiguous decoding algorithms for the concatenated code. We do so via a claim that is weak in that it only shows that the expected number of errors and erasures fall within the decoding capability of the outer code.

Lemma 11 *Let* $e_i = \Delta(\mathbf{r}_i, \mathbf{y}_i)$ *and* $e = \sum_{i=1}^{n} e_i$. *When* $\mathbf{v}$ *is picked at random by* **Random-Concat-Decoder**, *then we have:*

$$\mathrm{Exp}\left[\#\ \textit{erasures in}\ \mathbf{v}\ + 2 \cdot (\#\ \textit{errors in}\ \mathbf{v})\right] \leq 2e/d_2.$$

Remark: Note that if $e < dd_2/2$, the RHS above is less than d as one would hope for.
Proof Note that is suffices to argue that for every i

$$\Pr\left[\text{erasure in } i\text{th coordinate}\right] 2 \cdot \Pr\left[\text{error in } i\text{th coordinate}\right] \leq 2e_i/d_2, \tag{1}$$

and then the lemma will follow by linearity of expectations. We prove this in cases.
Case 1: $u_i = x_i$: In this case, the probability of an error in the ith coordinate is 0 and the probability of an erasure in the ith coordinate is $2e_i'/d_2$. Since $u_i = x_i$, we also have $e_i = e_i'$, and so we find that the LHS of (1) is indeed equal to $2e_i/d_2$.
Case 2: $u_i \neq x_i$. In this case, we could have an error in the ith coordinate — this happens with probability $1 - 2e_i'/d_2$, while with probability $2e_i'/d_2$ we get an error. We need to express these quantities as a function of e_i (rather than e_i') and so we note that $e_i \geq d_2 - e_i'$ (since $\Delta(E_2(x_i), \mathbf{r}_i) \geq \Delta(E_2(x_i), E_2(u_i)) - \Delta(\mathbf{r}_i, E_2(u_i))$). Now we have that the probability of an error is at most $2e_i/d_2 - 1$ and the probability of an erasure is at most $2 - 2e_i/d_2$. Plugging these into the LHS of (1), again we find that the LHS is bounded by $2e_i/d_2$. ■

The above lemma implies that there is positive probability associated with the event that $\mathbf{v}$ has small number of errors and erasures. However we did prove not a high probability result. We won't do so; instead we will just derandomize the algorithm above to get a deterministic algorithm for the unambiguous decoding problem.

2.3 Deterministic Decoder for Concatenated Codes

We will develop the deterministic decoder in two (simple) steps. First note that we didn't really need the random choices in Step 3 of **Random-Concat-Decoder** do not need to independent and expectation bound (Lemma 11) also holds if these events are completely dependent. So, in particular, the following algorithm would work as well.

Modified-Random-Concat-Decoder

Given: $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_n \rangle$.

Step 1: For $i \in [n]$, compute u_i that minimizes $\Delta(E_2(u_i), \mathbf{r}_i)$.

Step 2: Set $e_i' = \min\{d_2/2, \Delta(E_2(u_i).\mathbf{r}_i)$.

Step 3.1: Pick $p \leftarrow [0, 1]$ uniformly at random.

Step 3.2: For every $i \in [n]$ if $2e_i'/d_2 > p$, set $v_i = ?$, else set $v_i = u_i$.

Step 4: Perform errors and erasures decoding of the vector $\mathbf{v} = \langle v_1, \ldots, v_n \rangle$.

Step 5: If a vector $\mathbf{m}' \in \mathbb{F}_Q^k$ is obtained in Step 4, check to see if $\Delta(E_2 \diamond E_1(\mathbf{m}), \mathbf{r}) \leq (dd_2)/2$ and if so output $\mathbf{m}'$.

As in the analysis of **Random-Concat-Decoder** we see that the random variable $\mathbf{v}$ as defined in Steps 3.1 and 3.2 above satisfies the condition:

$$\text{Exp}\left[\# \text{ erasures in } \mathbf{v} \ + 2 \cdot (\# \text{ errors in } \mathbf{v})\right] \leq 2e/d_2.$$

In particular, there exists a choice of p in Step 3.1, such that for this choice of p, the vector $\mathbf{v}$ obtained in Step 3.2 satisfies the condition:

$$(\# \text{ erasures in } \mathbf{v}) \ + 2 \cdot (\# \text{ errors in } \mathbf{v}) \leq 2e/d_2.$$

The only interesting choices of p are from the set $S = \{0,1\} \cup \{2e_i'/d_2 | i \in [n]\}$ since for every other choice of p, there is some $p' \in S$ for which the vector $\mathbf{v}$ obtained is identical for p' and p. This gives us the deterministic algorithm below:

Deterministic-Concat-Decoder

Given: $\mathbf{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_n \rangle$.

Step 1: For $i \in [n]$, compute u_i that minimizes $\Delta(E_2(u_i), \mathbf{r}_i)$.

Step 2: Set $e_i' = \min\{d_2/2, \Delta(E_2(u_i).\mathbf{r}_i)$.

Step 3.1: Let $S = \{0,1\} \cup \{2e_i'/d_2 | i \in [n]\}$. Repeat Steps 3.2 to 5 for every choice of $p \in S$.

Step 3.2: For every $i \in [n]$ if $2e_i'/d_2 > p$, set $v_i = ?$, else set $v_i = u_i$.

Step 4: Perform errors and erasures decoding of the vector $\mathbf{v} = \langle v_1, \ldots, v_n \rangle$.

Step 5: If a vector $\mathbf{m}' \in \mathbb{F}_Q^k$ is obtained in Step 4, check to see if $\Delta(E_2 \diamond E_1(\mathbf{m}), \mathbf{r}) \leq (dd_2)/2$ and if so output $\mathbf{m}'$.

The running time of this algorithm is $O(nn_2Q + nT(n))$ where $T(n)$ is the running time of the errors and erasures decoder of the outer code. Correctness follows from the arguments developed so far. We summarize the discussion with the following theorem:

Theorem 12 *Let C be the concatenation of an $[n,k,d]_Q$ outer code C_1 and an $[n_2,k_2,d_2]_q$ inner codee C_2 with $Q = q^{k_2}$. Suppose C_1 has an "errors and erasures decoding algorithm" running in time $T(n)$ decoding up to s erasures and t errors provided $s + 2t < d$. Then C has an unambiguous decoding algorithm running in time $O(nn_2Q + nT(n))$.*

Again, the running times can be improved with some effort. In particular, Kötter [7] shows how to cut down the run time to just $O(nn_2Q + T(n))$ for some families of concatenated codes.

References

[1] Iwan M. Duursma. *Decoding Codes from Curves and Cyclic Codes.* PhD thesis, Eindhoven University of Technology, 1993.

[2] Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. *IEEE Transactions on Information Theory*, 46(5):1330–1338, July 2000. Extended version appears as ECCC Technical Report TR98-062 (Revision 4), `http://www.eccc.uni-trier.de/eccc`.

[3] Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. Soft-decision decoding of Chinese Remainder codes. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, pages 159–168, Redondo Beach, California, 12-14 November 2000.

[4] G. David Forney Jr. *Concatenated Codes*. MIT Press, Cambridge, MA, 1966.

[5] G. David Forney Jr. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12(2):125–131, April 1966.

[6] Ralf Kötter. A unified description of an error locating procedure for linear codes. In *Proceedings of the International Workshop on Algebraic and Combinatorial Coding Theory*, pages 113–117, Voneshta Voda, Bulgaria, 1992.

[7] Ralf Kötter. Fast generalized minimum distance decoding of algebraic geometry and Reed-Solomon codes. *IEEE Transactions on Information Theory*, 42(3):721–737, May 1996.

[8] David M. Mandelbaum. On a class of arithmetic codes and a decoding algorithm. *IEEE Transactions on Information Theory*, 21(1):85–88, January 1976.

[9] Ruud Pellikaan. On decoding linear codes by error correcting pairs. Preprint, Eindhoven University of Technology, 1988.