Binary Linear Codes

Coding Technology

Illés Horváth

2025/09/17

Shannon's limit

Previously, we looked at upper bounds of binary codes in terms of n, k and d_{min} (Singleton bound, Hamming bound).

Shannon's limit

Previously, we looked at upper bounds of binary codes in terms of n, k and d_{min} (Singleton bound, Hamming bound).

Next we look at an upper bound that takes into account the p_b bit error probability of the channel, too. Intuitively, for a noisier channel, only a lower code rate can be obtained with low block error probability.

Shannon's limit

Previously, we looked at upper bounds of binary codes in terms of n, k and d_{min} (Singleton bound, Hamming bound).

Next we look at an upper bound that takes into account the p_b bit error probability of the channel, too. Intuitively, for a noisier channel, only a lower code rate can be obtained with low block error probability.

This is indeed the case, and is known as the Noisy-channel coding theorem or Shannon's limit.

Shannon's limit

Define the channel entropy function as

$$H(p_b) = -(p_b \log_2 p_b + (1 - p_b) \log_2 (1 - p_b)).$$

Shannon's limit

Define the channel entropy function as

$$H(p_b) = -(p_b \log_2 p_b + (1 - p_b) \log_2 (1 - p_b)).$$

Theorem (Shannon's limit)

Assume we have a BSC with bit error probability p_b.

(a) *For any $\varepsilon > 0$ and any $r < 1 - H(p_b)$, there is an n_0 such that for any $n > n_0$, there exists a $C(n, k)$ code with code rate $\frac{k}{n} \geq r$ and block error probability $< \varepsilon$.*

Shannon's limit

Define the channel entropy function as

$$H(p_b) = -(p_b \log_2 p_b + (1 - p_b) \log_2 (1 - p_b)).$$

Theorem (Shannon's limit)

Assume we have a BSC with bit error probability p_b.

(a) *For any $\varepsilon > 0$ and any $r < 1 - H(p_b)$, there is an n_0 such that for any $n > n_0$, there exists a $C(n, k)$ code with code rate $\frac{k}{n} \geq r$ and block error probability $< \varepsilon$.*

(b) *For any $r > 1 - H(p_b)$ there exists an $\varepsilon > 0$ such that any $C(n, k)$ code with code rate $\frac{k}{n} > r$ has block error probability $\geq \varepsilon$.*

Shannon's limit

Define the channel entropy function as

$$H(p_b) = -(p_b \log_2 p_b + (1 - p_b) \log_2 (1 - p_b)).$$

Theorem (Shannon's limit)

Assume we have a BSC with bit error probability p_b.

(a) *For any $\varepsilon > 0$ and any $r < 1 - H(p_b)$, there is an n_0 such that for any $n > n_0$, there exists a $C(n, k)$ code with code rate $\frac{k}{n} \geq r$ and block error probability $< \varepsilon$.*

(b) *For any $r > 1 - H(p_b)$ there exists an $\varepsilon > 0$ such that any $C(n, k)$ code with code rate $\frac{k}{n} > r$ has block error probability $\geq \varepsilon$.*

We omit the proof here, but the proof is non-constructive anyway, which means that it does not design any specific code, just shows that it exists.

Shannon's limit

Shannon's limit essentially states that the code rate $1 - H(p_b)$ can be reached with arbitrarily small block error probability.

Shannon's limit

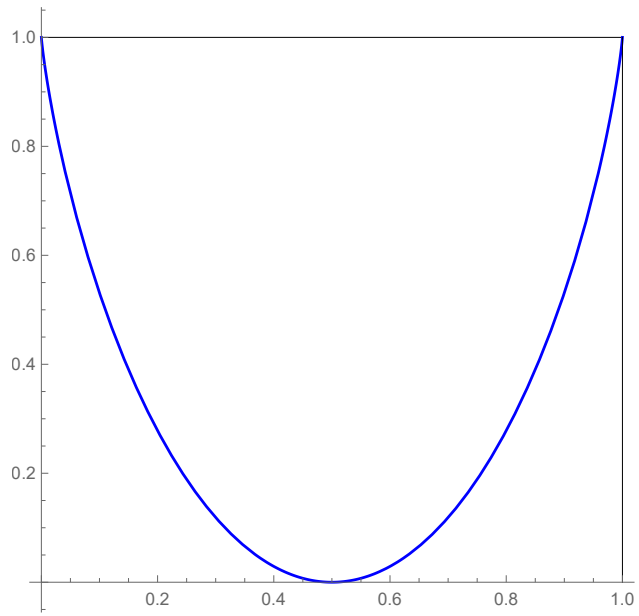
Shannon's limit essentially states that the code rate $1 - H(p_b)$ can be reached with arbitrarily small block error probability.

On the other hand, for any code with code rate $> 1 - H(p_b)$, there will inherently be errors.

Shannon's limit

Shannon's limit essentially states that the code rate $1 - H(p_b)$ can be reached with arbitrarily small block error probability.

On the other hand, for any code with code rate $> 1 - H(p_b)$, there will inherently be errors.



$1 - H(p_b)$

Linear functions

A function ψ between two linear spaces is linear if for any u_1, u_2 vectors and s_1, s_2 scalars,

$$\psi(s_1 u_1 + s_2 u_2) = s_1 \psi(u_1) + s_2 \psi(u_2).$$

Linear functions

A function ψ between two linear spaces is linear if for any u_1, u_2 vectors and s_1, s_2 scalars,

$$\psi(s_1 u_1 + s_2 u_2) = s_1 \psi(u_1) + s_2 \psi(u_2).$$

The above definition is for general linear spaces. For binary spaces, it can be simplified a little.

Linear functions

A function ψ between two linear spaces is linear if for any u_1, u_2 vectors and s_1, s_2 scalars,

$$\psi(s_1 u_1 + s_2 u_2) = s_1 \psi(u_1) + s_2 \psi(u_2).$$

The above definition is for general linear spaces. For binary spaces, it can be simplified a little.

A function $\psi : \{0,1\}^n \to \{0,1\}^k$ is called linear if for any u_1, u_2 vectors,

$$\psi(u_1 + u_2) = \psi(u_1) + \psi(u_2).$$

Linear functions

A function ψ between two linear spaces is linear if for any u_1, u_2 vectors and s_1, s_2 scalars,

$$\psi(s_1 u_1 + s_2 u_2) = s_1 \psi(u_1) + s_2 \psi(u_2).$$

The above definition is for general linear spaces. For binary spaces, it can be simplified a little.

A function $\psi : \{0,1\}^n \to \{0,1\}^k$ is called linear if for any u_1, u_2 vectors,

$$\psi(u_1 + u_2) = \psi(u_1) + \psi(u_2).$$

(For binary vectors, s_1 and s_2 could only be either 0 or 1, with only $s_1 = s_2 = 1$ meaningful.)

Linear functions

Example. This is the coding function of the C(5,2) code seen before. Is ψ linear?

ψ :

u	c
00	00000
01	00111
10	11100
11	11011

Linear functions

Example. This is the coding function of the C(5,2) code seen before. Is ψ linear?

ψ :

u	c
00	00000
01	00111
10	11100
11	11011

Actually, the only thing we really need to check is

$$\psi(01) + \psi(10) = \psi(11)$$
$$(00111) + (11100) = (11011),$$

which holds, so ψ is linear.

Linear functions

Theorem

If ψ is a $\psi : \{0,1\}^n \to \{0,1\}^k$ linear function, then there exists a $k \times n$ binary matrix G such that

$$\psi(u) = uG.$$

Linear functions

Theorem

If ψ is a $\psi : \{0,1\}^n \to \{0,1\}^k$ linear function, then there exists a $k \times n$ binary matrix G such that

$$\psi(u) = uG.$$

The rows of G are the vectors $\psi(e_1), \ldots, \psi(e_k)$, where the e_i's are the unit vectors, that is,

$$e_i = [0 \ldots 0 \underbrace{1}_{i} 0 \ldots 0].$$

Linear functions

Theorem

If ψ is a $\psi : \{0,1\}^n \to \{0,1\}^k$ linear function, then there exists a $k \times n$ binary matrix G such that

$$\psi(u) = uG.$$

The rows of G are the vectors $\psi(e_1), \ldots, \psi(e_k)$, where the e_i's are the unit vectors, that is,

$$e_i = [0 \ldots 0 \underbrace{1}_{i} 0 \ldots 0].$$

Vice versa, for any $k \times n$ binary matrix G, the function $u \to uG$ is linear.

Linear maps

Proof.

$$u = \sum_{i=1}^{k} u_i e_i,$$

where u_i denotes the i-th coordinate of u.

Linear maps

Proof.

$$u = \sum_{i=1}^{k} u_i e_i,$$

where u_i denotes the i-th coordinate of u.

$$\psi(u) = \sum_{i=1}^{k} u_i \psi(e_i);$$

for coordinates where $u_i = 1$, we use the linear property of ψ, and terms where $u_i = 0$ just cancel out.

Linear maps

Proof.

$$u = \sum_{i=1}^{k} u_i e_i,$$

where u_i denotes the i-th coordinate of u.

$$\psi(u) = \sum_{i=1}^{k} u_i \psi(e_i);$$

for coordinates where $u_i = 1$, we use the linear property of ψ, and terms where $u_i = 0$ just cancel out.

For the matrix G with rows $\psi(e_1), \ldots, \psi(e_k)$, computing the matrix-vector product uG gives exactly

$$uG = \sum_{i=1}^{k} u_i \psi(e_i).$$

Linear maps

Example. Compute the G matrix for the ψ function.

ψ :

u	c
00	00000
01	00111
10	11100
11	11011

Linear maps

Example. Compute the G matrix for the ψ function.

ψ :

u	c
00	00000
01	00111
10	11100
11	11011

According to the theorem, we have to put together the codewords corresponding to the unit vectors (01) and (10):

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Linear maps

Example. Compute the G matrix for the ψ function.

ψ :

u	c
00	00000
01	00111
10	11100
11	11011

According to the theorem, we have to put together the codewords corresponding to the unit vectors (01) and (10):

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Linear codes inherently map the all 0 message to the all 0 codeword, e.g. $(00) \rightarrow (00000)$.

Linear codes

If, for some (binary) error correction code, the $\psi : u \to c$ function is linear, then we call it a (binary) linear code.

Linear codes

If, for some (binary) error correction code, the $\psi : u \to c$ function is linear, then we call it a (binary) linear code.

The matrix G is called the generator matrix of the code.

Linear codes

If, for some (binary) error correction code, the $\psi : u \to c$ function is linear, then we call it a (binary) linear code.

The matrix G is called the generator matrix of the code.

For linear codes, the matrix G is a very efficient representation of the code. The table containing all (u, c) pairs has 2^k rows; instead, G is a $k \times n$ matrix, which is significantly smaller.

Linear codes

If, for some (binary) error correction code, the $\psi : u \to c$ function is linear, then we call it a (binary) linear code.

The matrix G is called the generator matrix of the code.

For linear codes, the matrix G is a very efficient representation of the code. The table containing all (u, c) pairs has 2^k rows; instead, G is a $k \times n$ matrix, which is significantly smaller.

Vector-matrix multiplication is fast (often with a dedicated Digital Signal Processor (DSP) unit involving multiply-accumulate operations), so we can compute $c = uG$ online, there is no need to store all (u, c) pairs.

Parity-check matrix

For a $C(n, k)$ linear code with generator matrix G, we call an $(n - k) \times n$ matrix H a parity-check matrix if the rows of H are linearly independent, and

$$G \cdot H^T = 0.$$

Parity-check matrix

For a $C(n, k)$ linear code with generator matrix G, we call an $(n - k) \times n$ matrix H a parity-check matrix if the rows of H are linearly independent, and

$$G \cdot H^T = 0.$$

Theorem

For any generator matrix G, there always exists an H parity-check matrix.

Parity-check matrix

For a $C(n, k)$ linear code with generator matrix G, we call an $(n - k) \times n$ matrix H a parity-check matrix if the rows of H are linearly independent, and

$$G \cdot H^T = 0.$$

Theorem

For any generator matrix G, there always exists an H parity-check matrix.

Proof (sketch). In the linear space $\{0, 1\}^n$, consider the k-dimensional subspace spanned by the rows of G. The orthogonal complement of this subspace is $(n - k)$-dimensional. Any $n - k$ linearly independent vectors from this subspace are suitable as the rows of H.

Parity-check matrix

For a $C(n, k)$ linear code with generator matrix G, we call an $(n - k) \times n$ matrix H a parity-check matrix if the rows of H are linearly independent, and

$$G \cdot H^T = 0.$$

Theorem

For any generator matrix G, there always exists an H parity-check matrix.

Proof (sketch). In the linear space $\{0, 1\}^n$, consider the k-dimensional subspace spanned by the rows of G. The orthogonal complement of this subspace is $(n - k)$-dimensional. Any $n - k$ linearly independent vectors from this subspace are suitable as the rows of H.

The parity check matrix H will be useful for decoding.

Systematic linear codes

For systematic linear codes, G and H have a nice structure.

Theorem

Assume we have a linear code with generator matrix G. The following three properties are equivalent:

- *the code is systematic;*
- *the leftmost $k \times k$ block of G is the identity matrix;*
- *the rightmost $(n-k) \times (n-k)$ block of H is the identity matrix.*

Systematic linear codes

For systematic linear codes, G and H have a nice structure.

Theorem

Assume we have a linear code with generator matrix G. The following three properties are equivalent:

- *the code is systematic;*
- *the leftmost $k \times k$ block of G is the identity matrix;*
- *the rightmost $(n-k) \times (n-k)$ block of H is the identity matrix.*

Moreover,

$$G = [I_k | B] \qquad \Longrightarrow \qquad H = [B^T | I_{n-k}].$$

(B is of size $k \times (n-k)$).

Systematic linear codes

Proof. If $G = [I_k|B]$, then

$$u \cdot G = [u \cdot I_k|uB] = [u|uB],$$

so the code is systematic.

Systematic linear codes

Proof. If $G = [I_k|B]$, then

$$u \cdot G = [u \cdot I_k|uB] = [u|uB],$$

so the code is systematic.

The other way round, the only $k \times k$ matrix that leaves every vector unchanged is I_k.

Systematic linear codes

Proof. If $G = [I_k|B]$, then

$$u \cdot G = [u \cdot I_k|uB] = [u|uB],$$

so the code is systematic.

The other way round, the only $k \times k$ matrix that leaves every vector unchanged is I_k.

For the second part, we compute $G \cdot H^T$ in block form:

$$G \cdot H^T = [I_k|B] \cdot [B|I_{n-k}] = [I_k \cdot B + B \cdot I_{n-k}] = [B + B] = [0].$$

Systematic linear codes

Proof. If $G = [I_k|B]$, then

$$u \cdot G = [u \cdot I_k|uB] = [u|uB],$$

so the code is systematic.

The other way round, the only $k \times k$ matrix that leaves every vector unchanged is I_k.

For the second part, we compute $G \cdot H^T$ in block form:

$$G \cdot H^T = [I_k|B] \cdot [B|I_{n-k}] = [I_k \cdot B + B \cdot I_{n-k}] = [B + B] = [0].$$

Finally, the rows of H are linearly independent because the rows of the I_{n-k} block are already linearly independent.

Example

Consider the binary linear code with generator matrix

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Is the code systematic?

Example

Consider the binary linear code with generator matrix

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Is the code systematic?

No, because the leftmost 2×2 block of G is not $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Example

Consider the binary linear code with generator matrix

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Is the code systematic?

No, because the leftmost 2×2 block of G is not $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

This can be fixed by exchanging columns 1 and 5 in G:

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Example

Compute a good parity-check matrix H for the generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Example

Compute a good parity-check matrix H for the generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Since the code is now systematic, we write

$$G = \left[\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{array}\right],$$

I_2 B

Example

Compute a good parity-check matrix H for the generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Since the code is now systematic, we write

$$G = \left[\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{array}\right],$$

I_2 B

and then H can be computed as

$$H = (B^T, I_{n-k}) = \left[\begin{array}{cc|ccc} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

B^T I_3

$d_{\min}$

Theorem

For linear codes, the codewords form a linear subspace of $\{0,1\}^n$.

$d_{\min}$

Theorem

For linear codes, the codewords form a linear subspace of $\{0,1\}^n$.

For linear codes,

$$d_{\min} = \min_{c \neq 0} w(c).$$

$d_{\min}$

Theorem

For linear codes, the codewords form a linear subspace of $\{0,1\}^n$.

For linear codes,

$$d_{\min} = \min_{c \neq 0} w(c).$$

Proof. First part. If $c = uG$ and $c' = uG'$, then

$$c + c' = (u + u')G.$$

$d_{\min}$

Theorem

For linear codes, the codewords form a linear subspace of $\{0,1\}^n$.

For linear codes,

$$d_{\min} = \min_{c \neq 0} w(c).$$

Proof. First part. If $c = uG$ and $c' = uG'$, then

$$c + c' = (u + u')G.$$

For the second part,

$$d_{\min} = \min_{c' \neq c''} d(c', c'') = \min_{c' \neq c''} w(c' - c'') = \min_{c} w(c).$$

Syndrome vector

For general binary block codes, decoding was done by minimizing $d(v, c)$ for the received vector v.

For binary linear codes, decoding is done using a different method, known as syndrome decoding.

Syndrome vector

For general binary block codes, decoding was done by minimizing $d(v, c)$ for the received vector v.

For binary linear codes, decoding is done using a different method, known as syndrome decoding.

For any received vector v, the corresponding syndrome vector (or just syndrome) s is defined as

$$s = vH^T.$$

Syndrome vector

For every codeword c,

$$cH^T = uGH^T = u \cdot 0 = 0.$$

Syndrome vector

For every codeword c,

$$cH^T = uGH^T = u \cdot 0 = 0.$$

This implies that a syndrome vector s corresponding to a received vector v depends only on the error vector e, but not the codeword c (hence the name). That is, for $v = c + e$,

$$vH^T = (c + e)H^T = cH^T + eH^T = eH^T.$$

Syndrome vector

For every codeword c,

$$cH^T = uGH^T = u \cdot 0 = 0.$$

This implies that a syndrome vector s corresponding to a received vector v depends only on the error vector e, but not the codeword c (hence the name). That is, for $v = c + e$,

$$vH^T = (c + e)H^T = cH^T + eH^T = eH^T.$$

For decoding of linear codes, we are going to replace finding the c' with minimal $d(v, c')$ by syndrome decoding: we compute the syndrome vector s, then try to guess what the error vector e was based on the syndrome s.

Syndrome vector

Syndrome vectors have length $n - k$. For a $C(n, k)$ linear code,

- the number of possible syndrome vectors is 2^{n-k}, while
- the number of possible error vectors is 2^n,

so some of the error vectors will give the same syndrome vector.

Syndrome vector

Syndrome vectors have length $n - k$. For a $C(n, k)$ linear code,

- the number of possible syndrome vectors is 2^{n-k}, while
- the number of possible error vectors is 2^n,

so some of the error vectors will give the same syndrome vector.

We look to group error vectors according to what syndrome they give.

Syndrome vector

Syndrome vectors have length $n - k$. For a $C(n, k)$ linear code,

- the number of possible syndrome vectors is 2^{n-k}, while
- the number of possible error vectors is 2^n,

so some of the error vectors will give the same syndrome vector.

We look to group error vectors according to what syndrome they give.

The naive approach to do that is to compute the syndrome vectors $s = eH^T$ for every possible error vector e, then group the error vectors according to the value of eH^T.

Example

Example. For the $C(5,2)$ code with

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

compute the syndrome vectors corresponding to the error vectors $(10000), (00001), (11010)$ respectively.

Example

Example. For the $C(5,2)$ code with

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

compute the syndrome vectors corresponding to the error vectors $(10000), (00001), (11010)$ respectively.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Example

Example. For the $C(5,2)$ code with

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

compute the syndrome vectors corresponding to the error vectors $(10000), (00001), (11010)$ respectively.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

So (00001) and (11010) belong to the same error group, but (10000) belongs to a different error group.

Example

Overall, the syndromes and their corresponding error groups for this code are the following:

$$
\begin{array}{lcl}
(000) & \rightarrow & \{(00000), (10110), (01101), (11011)\} \\
(001) & \rightarrow & \{(00001), (10111), (01100), (11010)\} \\
(010) & \rightarrow & \{(00010), (10100), (01111), (11001)\} \\
(100) & \rightarrow & \{(00100), (10010), (01001), (11111)\} \\
(101) & \rightarrow & \{(01000), (11110), (00101), (10011)\} \\
(110) & \rightarrow & \{(10000), (00110), (11101), (01011)\} \\
(011) & \rightarrow & \{(00011), (10101), (01110), (11000)\} \\
(110) & \rightarrow & \{(01010), (11100), (00111), (10001)\}
\end{array}
$$

Example

Overall, the syndromes and their corresponding error groups for this code are the following:

$$
\begin{array}{lcl}
(000) & \rightarrow & \{(00000), (10110), (01101), (11011)\} \\
(001) & \rightarrow & \{(00001), (10111), (01100), (11010)\} \\
(010) & \rightarrow & \{(00010), (10100), (01111), (11001)\} \\
(100) & \rightarrow & \{(00100), (10010), (01001), (11111)\} \\
(101) & \rightarrow & \{(01000), (11110), (00101), (10011)\} \\
(110) & \rightarrow & \{(10000), (00110), (11101), (01011)\} \\
(011) & \rightarrow & \{(00011), (10101), (01110), (11000)\} \\
(110) & \rightarrow & \{(01010), (11100), (00111), (10001)\}
\end{array}
$$

Due to $cH^T = 0$ for any codeword c, the error groups have a nice structure: for any e error vector, its error group is

$$\{e + c_1, \ldots, e + c_{2^k}\},$$

where $c_1, \ldots, c_{2^k}$ are the codewords.

Example

When doing syndrome decoding, we first compute the syndrome s from the received vector v; from the syndrome, we know that the actual error vector of the channel e must belong to the syndrome group corresponding to s.

Example

When doing syndrome decoding, we first compute the syndrome s from the received vector v; from the syndrome, we know that the actual error vector of the channel e must belong to the syndrome group corresponding to s.

Example. If the received vector is $v = (11010)$, then

$$s = vH^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

and the error group corresponding to $s = (001)$ is $\{(00001), (10111), (01100), (11010)\}$.

Example

When doing syndrome decoding, we first compute the syndrome s from the received vector v; from the syndrome, we know that the actual error vector of the channel e must belong to the syndrome group corresponding to s.

Example. If the received vector is $v = (11010)$, then

$$s = vH^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

and the error group corresponding to $s = (001)$ is $\{(00001), (10111), (01100), (11010)\}$.

One of these 4 vectors is the actual error of the channel; we need to make a guess. Which one should we pick?

Error groups

The best guess is the error vector with minimal weight, because that has the highest probability.

Error groups

The best guess is the error vector with minimal weight, because that has the highest probability.

Accordingly, in every error group, mark the error vector with minimal weight (red):

$$
\begin{array}{lcl}
(000) & \rightarrow & \{(00000), (10110), (01101), (11011)\} \\
(001) & \rightarrow & \{(00001), (10111), (01100), (11010)\} \\
(010) & \rightarrow & \{(00010), (10100), (01111), (11001)\} \\
(100) & \rightarrow & \{(00100), (10010), (01001), (11111)\} \\
(101) & \rightarrow & \{(01000), (11110), (00101), (10011)\} \\
(110) & \rightarrow & \{(10000), (00110), (11101), (01011)\} \\
(011) & \rightarrow & \{(00011), (10101), (01110), (11000)\} \\
(110) & \rightarrow & \{(01010), (11100), (00111), (10001)\}
\end{array}
$$

The red vectors are called the group leaders (or coset leaders).

Syndrome decoding

Ties are possible for the group leaders (blue). In case of a tie, we may select either error vector tied for minimal weight as the group leader, but we generally fix one.

Syndrome decoding

Ties are possible for the group leaders (blue). In case of a tie, we may select either error vector tied for minimal weight as the group leader, but we generally fix one.

The code can correct $\geq t$ errors $\iff$ in groups with minimal weight $\leq t$, the group leaders are unique (no ties).

Syndrome decoding

Ties are possible for the group leaders (blue). In case of a tie, we may select either error vector tied for minimal weight as the group leader, but we generally fix one.

The code can correct $\geq t$ errors $\iff$ in groups with minimal weight $\leq t$, the group leaders are unique (no ties).

The code is perfect $\iff$ there are no ties for group leader in any of the groups.

Syndrome decoding

Ties are possible for the group leaders (blue). In case of a tie, we may select either error vector tied for minimal weight as the group leader, but we generally fix one.

The code can correct $\geq t$ errors $\iff$ in groups with minimal weight $\leq t$, the group leaders are unique (no ties).

The code is perfect $\iff$ there are no ties for group leader in any of the groups.

For perfect codes, the decoding will be correct if there $\lfloor \frac{d_{\min}}{2} \rfloor$ errors, but the decoding will be erroneous if there are more errors.

For non-perfect codes, the decoding may still be correct with some probability even if there are more than $\lfloor \frac{d_{\min}}{2} \rfloor$ errors.

Syndrome decoding

Overall, syndrome decoding consists of the following steps. These are done offline in advance:

- Compute the error groups and select the group leader from each group.

Syndrome decoding

Overall, syndrome decoding consists of the following steps. These are done offline in advance:

- Compute the error groups and select the group leader from each group. (For large k, this can be too much computation; there are methods to find the group leader without computing the entire group, but we will not discuss these right now.)
- Store the syndromes and the corresponding group leaders in a lookup table.

Syndrome decoding

Overall, syndrome decoding consists of the following steps. These are done offline in advance:

- Compute the error groups and select the group leader from each group. (For large k, this can be too much computation; there are methods to find the group leader without computing the entire group, but we will not discuss these right now.)
- Store the syndromes and the corresponding group leaders in a lookup table.

During actual decoding:

- From the received vector v, compute the syndrome vector s.
- The detected error vector e' is the group leader corresponding to s from the lookup table.
- Compute the detected codeword as $c' = v - e'$.
- Compute u' (if the code is systematic, this step is just truncation).

Syndrome decoding table

Example. For the previous $C(5, 2)$ code, the syndrome decoding table is:

s	e
(000)	(00000)
(001)	(00001)
(010)	(00010)
(100)	(00100)
(101)	(01000)
(110)	(10000)
(011)	(00011)
(110)	(01010)

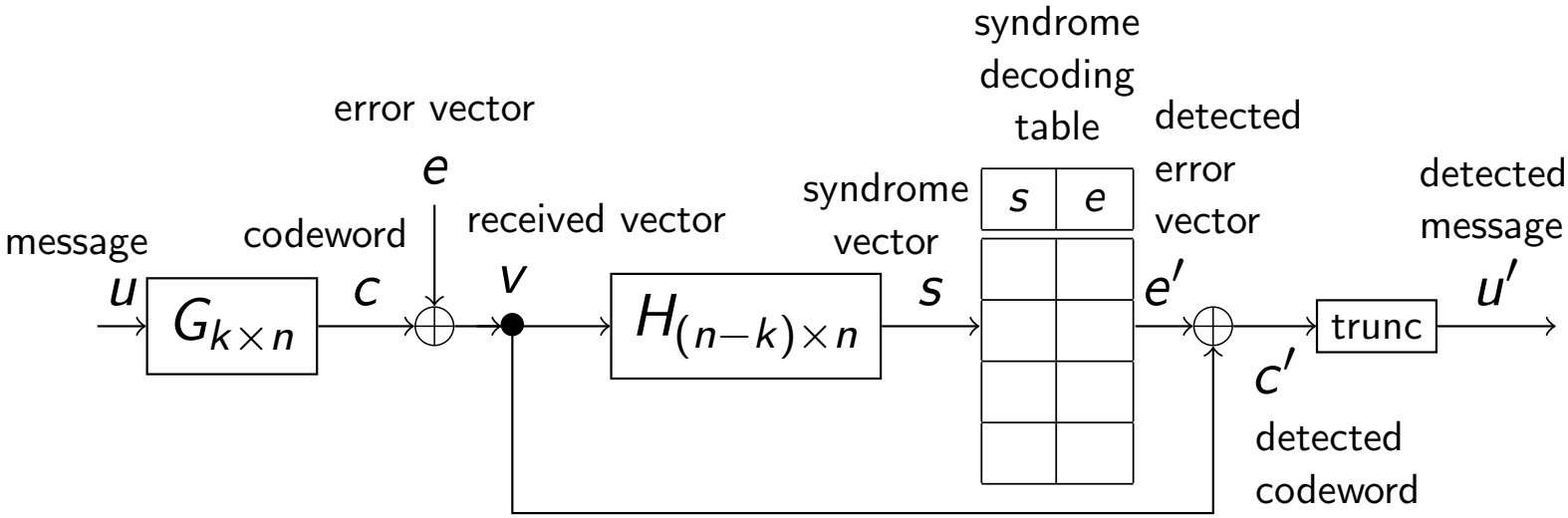
Syndrome decoding table

Example. For the previous $C(5,2)$ code, the syndrome decoding table is:

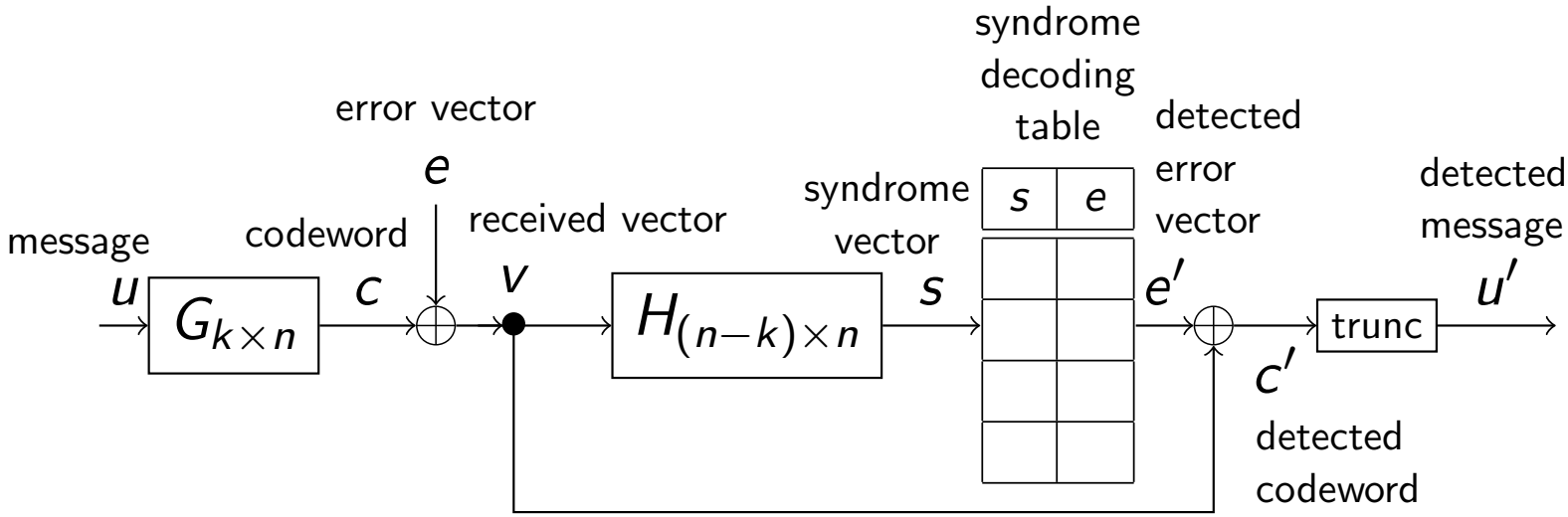
s	e
(000)	(00000)
(001)	(00001)
(010)	(00010)
(100)	(00100)
(101)	(01000)
(110)	(10000)
(011)	(00011)
(110)	(01010)

Errors in the same error group are called indistuingishable. If the actual error was the group leader, then decoding will be correct, but if it was another error from the group, decoding will be erroneous.

Syndrome decoding

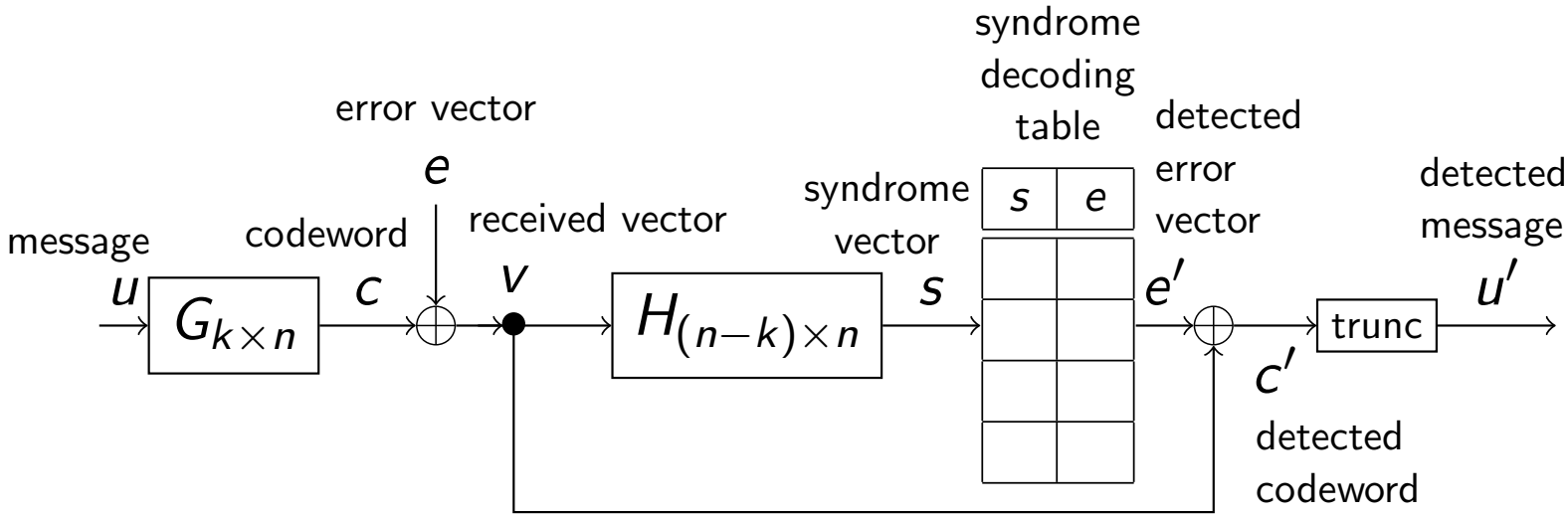


Syndrome decoding



For any binary linear code, this decoding scheme always gives the same u' as $\min d(v, c)$ in the binary block coding scheme previously.

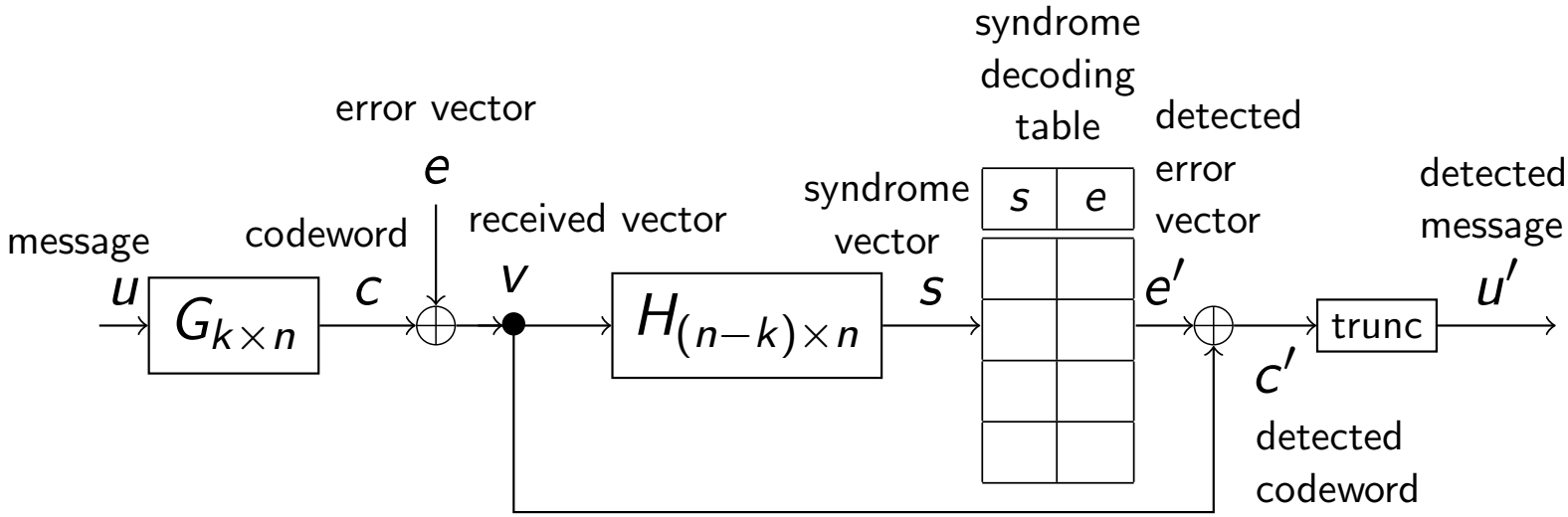
Syndrome decoding



For any binary linear code, this decoding scheme always gives the same u' as $\min d(v, c)$ in the binary block coding scheme previously.

So did we accomplish anything? That is, is this decoding scheme more efficient to compute?

Syndrome decoding

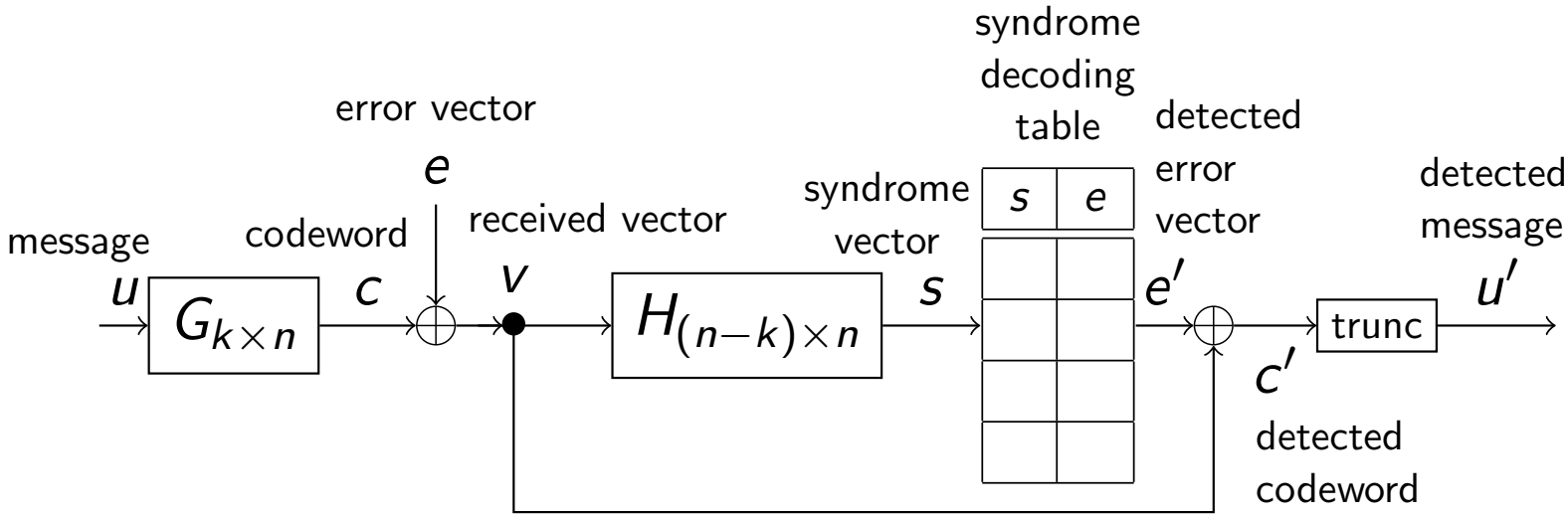


For any binary linear code, this decoding scheme always gives the same u' as $\min d(v, c)$ in the binary block coding scheme previously.

So did we accomplish anything? That is, is this decoding scheme more efficient to compute?

It is! The syndrome decoding table has 2^{n-k} rows, while for finding $\min d(v, c)$, we needed to do 2^k steps online.

Syndrome decoding



For any binary linear code, this decoding scheme always gives the same u' as $\min d(v, c)$ in the binary block coding scheme previously.

So did we accomplish anything? That is, is this decoding scheme more efficient to compute?

It is! The syndrome decoding table has 2^{n-k} rows, while for finding $\min d(v, c)$, we needed to do 2^k steps online.

$n - k$ is typically much smaller than k for codes that are used in practice (the good stuff, coming soon. . .)

Communication engineering

Quality-of-Service (QoS) approach: using error correction codes, we can decrease the probability of decoding error.

Communication engineering

Quality-of-Service (QoS) approach: using error correction codes, we can decrease the probability of decoding error.

Errors which are one of the group leaders can be corrected:

$$\begin{array}{cccc} (00000) & (00001) & (00010) & (00100) \\ (01000) & (10000) & (00011) & (01010) \end{array}$$

These include all error vectors with weight 0 and 1, and 2 error vectors with weight 2. All other error vectors cannot be corrected, so the block error probability is

$$p_e = \left(\binom{5}{2} - 2\right) p_b^2 (1-p_b)^3 + \binom{5}{3} p_b^3 (1-p_b)^2 + 5p_b^4(1-p_b) + p_b^5;$$

for $p_b = 0.1$,

$$p_e \approx 0.0669.$$

Improvement in QoS

What do we gain by using an error correcting code?

Improvement in QoS

What do we gain by using an error correcting code?

Consider what happens when we transmit messages with no coding on the same channel. For a 2-bit block, the probability of erroneous decoding is

$$1 - (1 - p_b)^2 = 0.19,$$

while it is 0.0669 when using error correcting code. (On the other hand, the code rate is $2/5$, so using the code effectively reduces the channel capacity to $2/5$ of the original capacity. Tradeoff.)

Improvement in QoS

Another possible comparison is to compare to another channel with different bit error probability p_b' where messages of block length 5 are transmitted without any coding. Compute p_b' so that the block error probability is the same as for the original channel with coding.

Improvement in QoS

Another possible comparison is to compare to another channel with different bit error probability p_b' where messages of block length 5 are transmitted without any coding. Compute p_b' so that the block error probability is the same as for the original channel with coding.

For the original channel, block error probability is 0.0669; for the channel with no coding, it is

$$1 - (1 - p_b')^5 = p_e = 0.0669 \quad \rightarrow \quad p_b' \approx 0.0137.$$

By using the error correcting code, we can obtain the same block error probability over a noisier channel ($p_b = 0.1 > 0.0137 = p_b'$).

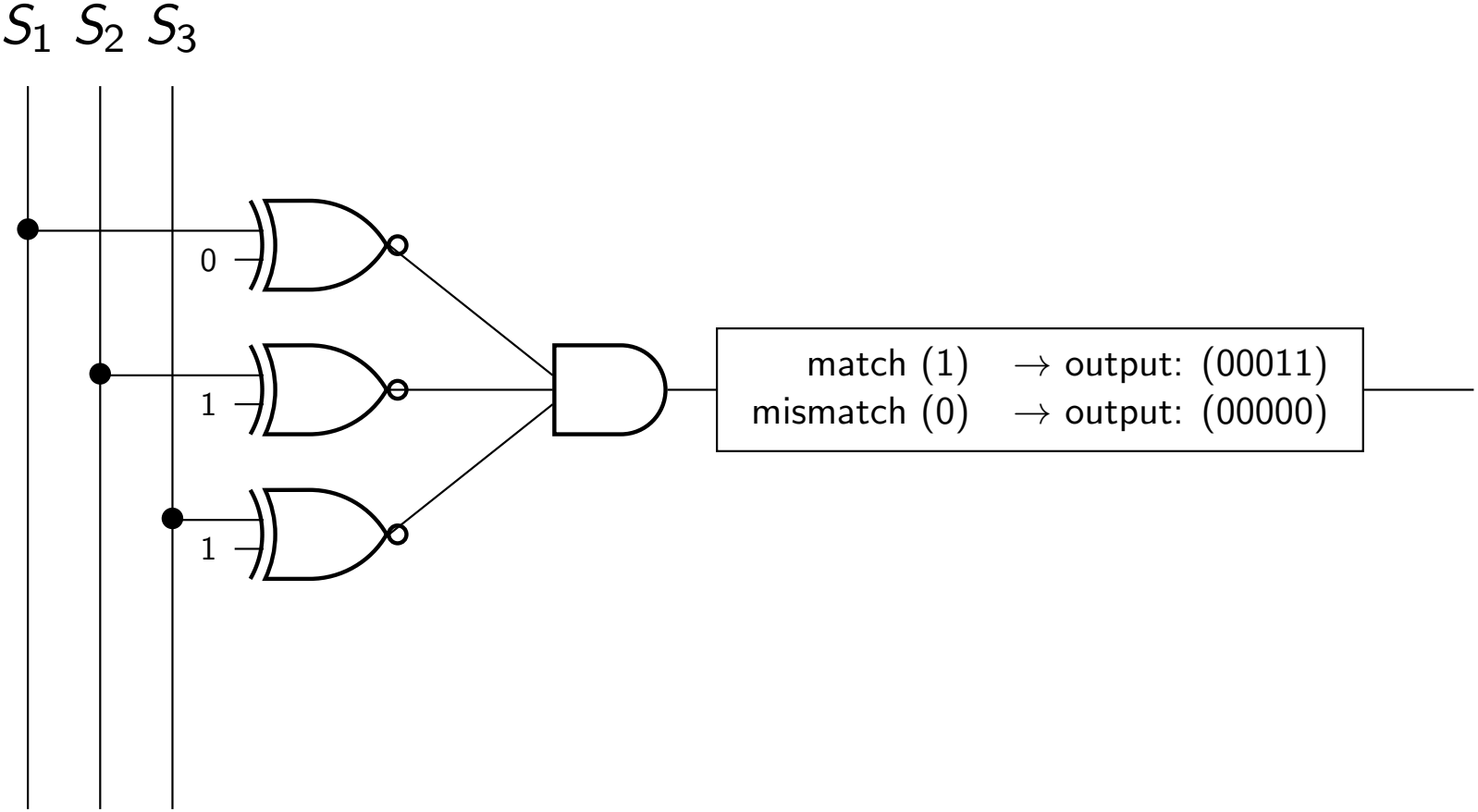
Architectural implementation

Next we look at how the syndrome decoding table (or any lookup table) can be implemented using an Arithmetic-Logic Unit (ALU).

Architectural implementation

Next we look at how the syndrome decoding table (or any lookup table) can be implemented using an Arithmetic-Logic Unit (ALU).

First, for every row of the lookup table (e.g. $s = (011)$) take this ALU:



Architectural implementation

Syndrome bits are directed into each such unit, and outputs are added as binary vectors (only 1 output will be nonzero).