Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

PART II

Outline

The Setting (I)

Perfect environment: No storage or communication errors or excessive message delivery delays

The Setting (I)

Perfect environment: No storage or communication errors or excessive message delivery delays

Goals:

- represent information in a systematic way to allow storage or transmission
- do it unambiguously and efficiently

The Setting (I)

Perfect environment: No storage or communication errors or excessive message delivery delays

Goals:

- represent information in a systematic way to allow storage or transmission
- do it unambiguously and efficiently

Imperfect environment: Storage or communication errors occur with some positive probability

The Setting (I)

Perfect environment: No storage or communication errors or excessive message delivery delays

Goals:

- represent information in a systematic way to allow storage or transmission
- do it unambiguously and efficiently

Imperfect environment: Storage or communication errors occur with some positive probability

Goals:

- represent information with some redundance allowing error detection and correction
- without loosing too much efficiency

Basic Concepts

Information is usually represented as a sequence of symbols in a fixed finite alphabet (e.g., natural languages, sampled audio or digital images)

Basic Concepts

Information is usually represented as a sequence of symbols in a fixed finite alphabet (e.g., natural languages, sampled audio or digital images)

Encoding: "A systematic method to translate sequences from one alphabet to another"
Decoding: "Recovering the original sequence from the encoded one"

Basic Concepts

Information is usually represented as a sequence of symbols in a fixed finite alphabet (e.g., natural languages, sampled audio or digital images)

Encoding: "A systematic method to translate sequences from one alphabet to another"
Decoding: "Recovering the original sequence from the encoded one"
Source alphabet: $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$
Target alphabet: $\mathcal{B} = \{\beta_1, \ldots, \beta_r\}$
Encoding map: $E : \mathcal{A} \longrightarrow \mathcal{B}^*$
(r-ary) **Code:** $\mathcal{C} = E(\mathcal{A}) \subset \mathcal{B}^*$
Codeword: Element in the code $\mathcal{C}$

Basic Concepts

Information is usually represented as a sequence of symbols in a fixed finite alphabet (e.g., natural languages, sampled audio or digital images)

Encoding: "A systematic method to translate sequences from one alphabet to another"
Decoding: "Recovering the original sequence from the encoded one"
Source alphabet: $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$
Target alphabet: $\mathcal{B} = \{\beta_1, \ldots, \beta_r\}$
Encoding map: $E : \mathcal{A} \longrightarrow \mathcal{B}^*$
(r-ary) **Code:** $\mathcal{C} = E(\mathcal{A}) \subset \mathcal{B}^*$
Codeword: Element in the code $\mathcal{C}$

E extends naturally to the set $\mathcal{A}^*$ by concatenation:

$$E(\emptyset) = \emptyset \qquad E(xy) = E(x)E(y)$$

Some Examples (I)

Unary representation of natural numbers (digits)

'1' $\rightarrow$ 1	'4' $\rightarrow$ 1111	'7' $\rightarrow$ 1111111
'2' $\rightarrow$ 11	'5' $\rightarrow$ 11111	'8' $\rightarrow$ 11111111
'3' $\rightarrow$ 111	'6' $\rightarrow$ 111111	'9' $\rightarrow$ 111111111

Some Examples (I)

Unary representation of natural numbers (digits)

'1' $\to$ 1	'4' $\to$ 1111	'7' $\to$ 1111111
'2' $\to$ 11	'5' $\to$ 11111	'8' $\to$ 11111111
'3' $\to$ 111	'6' $\to$ 111111	'9' $\to$ 111111111

Not good for concatenation!

Some Examples (I)

Unary representation of natural numbers (digits)

'1' $\to$ 1	'4' $\to$ 1111	'7' $\to$ 1111111
'2' $\to$ 11	'5' $\to$ 11111	'8' $\to$ 11111111
'3' $\to$ 111	'6' $\to$ 111111	'9' $\to$ 111111111

Not good for concatenation! Needs a separator or delimiter!

'0' $\to$ 0	'5' $\to$ 111110
'1' $\to$ 10	'6' $\to$ 1111110
'2' $\to$ 110	'7' $\to$ 11111110
'3' $\to$ 1110	'8' $\to$ 111111110
'4' $\to$ 11110	'9' $\to$ 1111111110

Some Examples (I)

Unary representation of natural numbers (digits)

'1' $\to$ 1	'4' $\to$ 1111	'7' $\to$ 1111111
'2' $\to$ 11	'5' $\to$ 11111	'8' $\to$ 11111111
'3' $\to$ 111	'6' $\to$ 111111	'9' $\to$ 111111111

Not good for concatenation! Needs a separator or delimiter!

'0' $\to$ 0	'5' $\to$ 111110
'1' $\to$ 10	'6' $\to$ 1111110
'2' $\to$ 110	'7' $\to$ 11111110
'3' $\to$ 1110	'8' $\to$ 111111110
'4' $\to$ 11110	'9' $\to$ 1111111110

Now $23501 = (\text{'2'},\text{'3'},\text{'5'},\text{'0'},\text{'1'}) \to \underbrace{110}_{2}\,\underbrace{1110}_{3}\,\underbrace{111110}_{5}\,\underbrace{0}_{0}\,\underbrace{10}_{1}$

Some Examples (II)

Binary representation of natural numbers (digits)

'0' $\to$ 0	'4' $\to$ 100	'8' $\to$ 1000
'1' $\to$ 01	'5' $\to$ 101	'9' $\to$ 1001
'2' $\to$ 10	'6' $\to$ 110	
'3' $\to$ 11	'7' $\to$ 111	

Some Examples (II)

Binary representation of natural numbers (digits)

'0' $\to$ 0	'4' $\to$ 100	'8' $\to$ 1000
'1' $\to$ 01	'5' $\to$ 101	'9' $\to$ 1001
'2' $\to$ 10	'6' $\to$ 110	
'3' $\to$ 11	'7' $\to$ 111	

Also needs a separator

Some Examples (II)

Binary representation of natural numbers (digits)

'0' $\to$ 0	'4' $\to$ 100	'8' $\to$ 1000
'1' $\to$ 01	'5' $\to$ 101	'9' $\to$ 1001
'2' $\to$ 10	'6' $\to$ 110	
'3' $\to$ 11	'7' $\to$ 111	

Also needs a separator or turn it into a fixed length code

'0' $\to$ 0000	'4' $\to$ 0100	'8' $\to$ 1000
'1' $\to$ 0001	'5' $\to$ 0101	'9' $\to$ 1001
'2' $\to$ 0010	'6' $\to$ 0110	
'3' $\to$ 0011	'7' $\to$ 0111	

Some Examples (II)

Binary representation of natural numbers (digits)

'0' $\to$ 0	'4' $\to$ 100	'8' $\to$ 1000
'1' $\to$ 01	'5' $\to$ 101	'9' $\to$ 1001
'2' $\to$ 10	'6' $\to$ 110	
'3' $\to$ 11	'7' $\to$ 111	

Also needs a separator or turn it into a fixed length code

'0' $\to$ 0000	'4' $\to$ 0100	'8' $\to$ 1000
'1' $\to$ 0001	'5' $\to$ 0101	'9' $\to$ 1001
'2' $\to$ 0010	'6' $\to$ 0110	
'3' $\to$ 0011	'7' $\to$ 0111	

Now $23501 = (\text{'2'},\text{'3'},\text{'5'},\text{'0'},\text{'1'}) \to \underbrace{0010}_{2}\underbrace{0011}_{3}\underbrace{0101}_{5}\underbrace{0000}_{0}\underbrace{0001}_{1}$

Some Examples (III)

ASCII, another fixed-length binary code

' ' $\rightarrow$ 00100000	'0' $\rightarrow$ 00110000	'@' $\rightarrow$ 01000000
'!' $\rightarrow$ 00100001	'1' $\rightarrow$ 00110001	'A' $\rightarrow$ 01000001
'"' $\rightarrow$ 00100010	'2' $\rightarrow$ 00110010	'B' $\rightarrow$ 01000010
'#' $\rightarrow$ 00100011	'3' $\rightarrow$ 00110011	'C' $\rightarrow$ 01000011
'$' $\rightarrow$ 00100100	'4' $\rightarrow$ 00110100	'D' $\rightarrow$ 01000100
'%' $\rightarrow$ 00100101	'5' $\rightarrow$ 00110101	'E' $\rightarrow$ 01000101
'&' $\rightarrow$ 00100110	'6' $\rightarrow$ 00110110	'F' $\rightarrow$ 01000110
''' $\rightarrow$ 00100111	'7' $\rightarrow$ 00110111	'G' $\rightarrow$ 01000111
...	...	...

Some Examples (IV)

QR Codes:

"Hello World!!!"

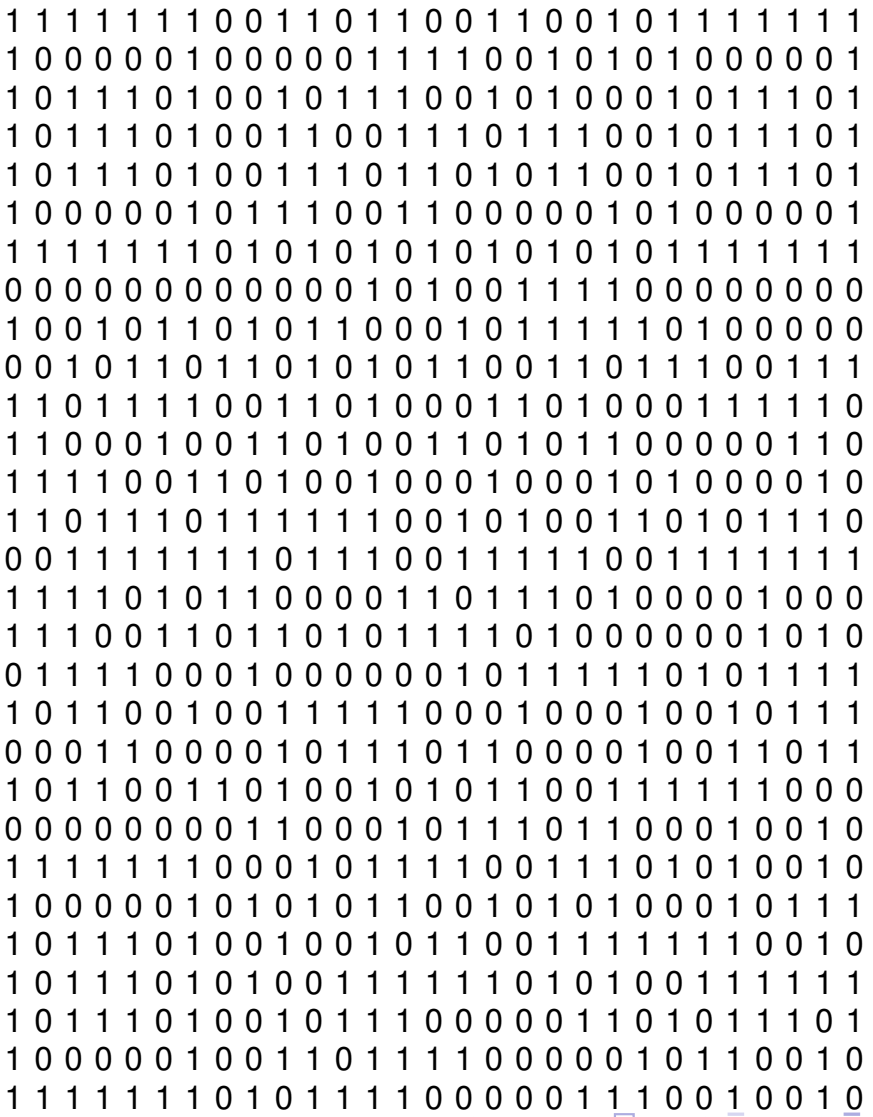
↓

Some Examples (IV)

QR Codes:

"Hello World!!!"

↓



Outline

An Example of Non-Uniquely Decodable Code (I)

Alphabets:

$\mathcal{A} = \{\alpha_1, \ldots, \alpha_5\}$

$\mathcal{B} = \{0, 1\}$

Code:

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

Encoding map:

$E(\alpha_1) = 01; \quad E(\alpha_2) = 001; \quad E(\alpha_3) = 010;$

$E(\alpha_4) = 10011; \quad E(\alpha_5) = 111$

An Example of Non-Uniquely Decodable Code (I)

Alphabets:

$\mathcal{A} = \{\alpha_1, \ldots, \alpha_5\}$

$\mathcal{B} = \{0, 1\}$

Code:

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

Encoding map:

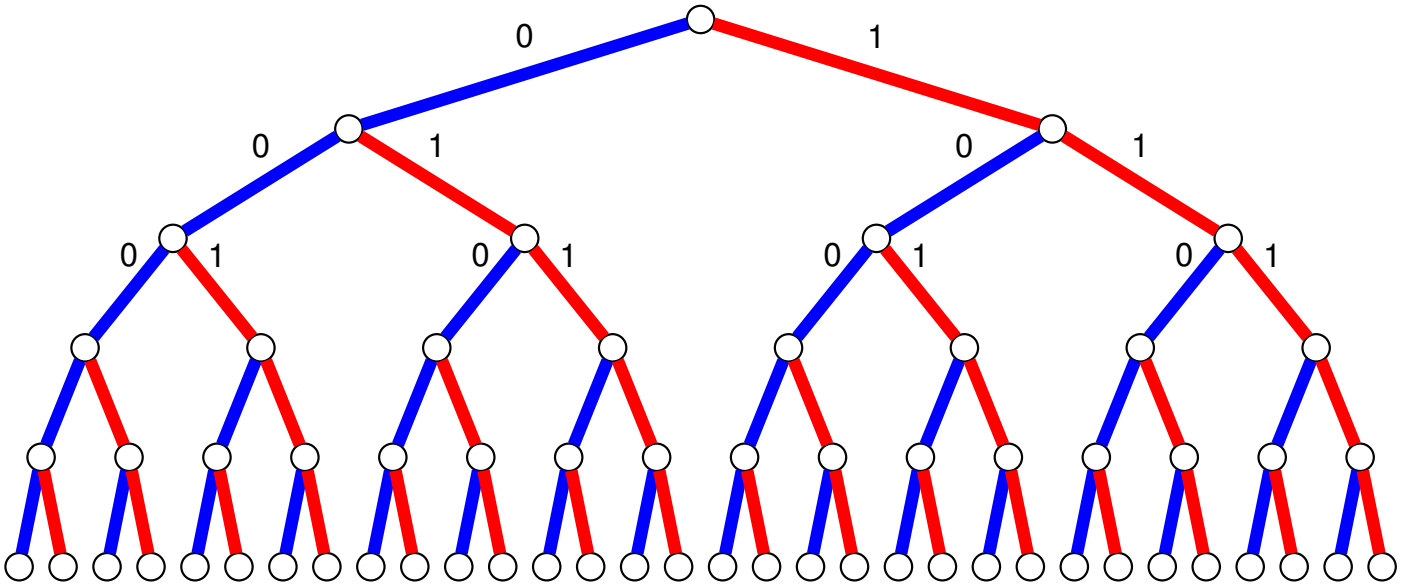
$E(\alpha_1) = 01; \quad E(\alpha_2) = 001; \quad E(\alpha_3) = 010;$

$E(\alpha_4) = 10011; \quad E(\alpha_5) = 111$

Problem: Decode the sequence

1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1

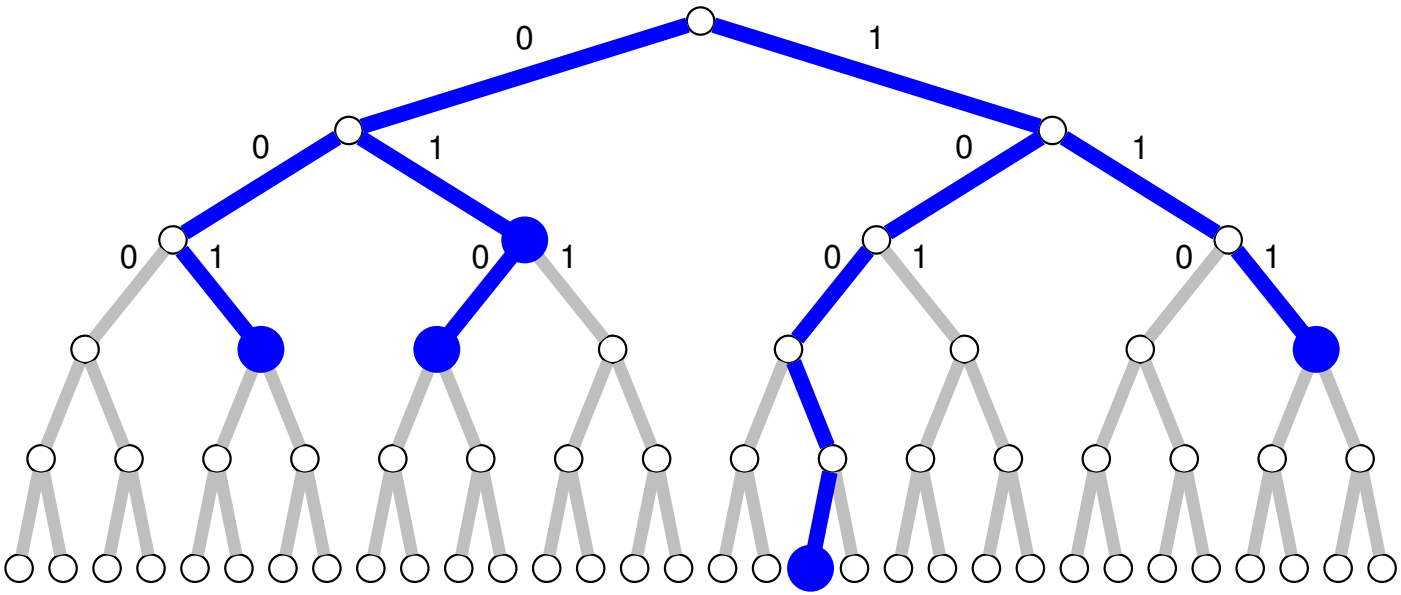
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1

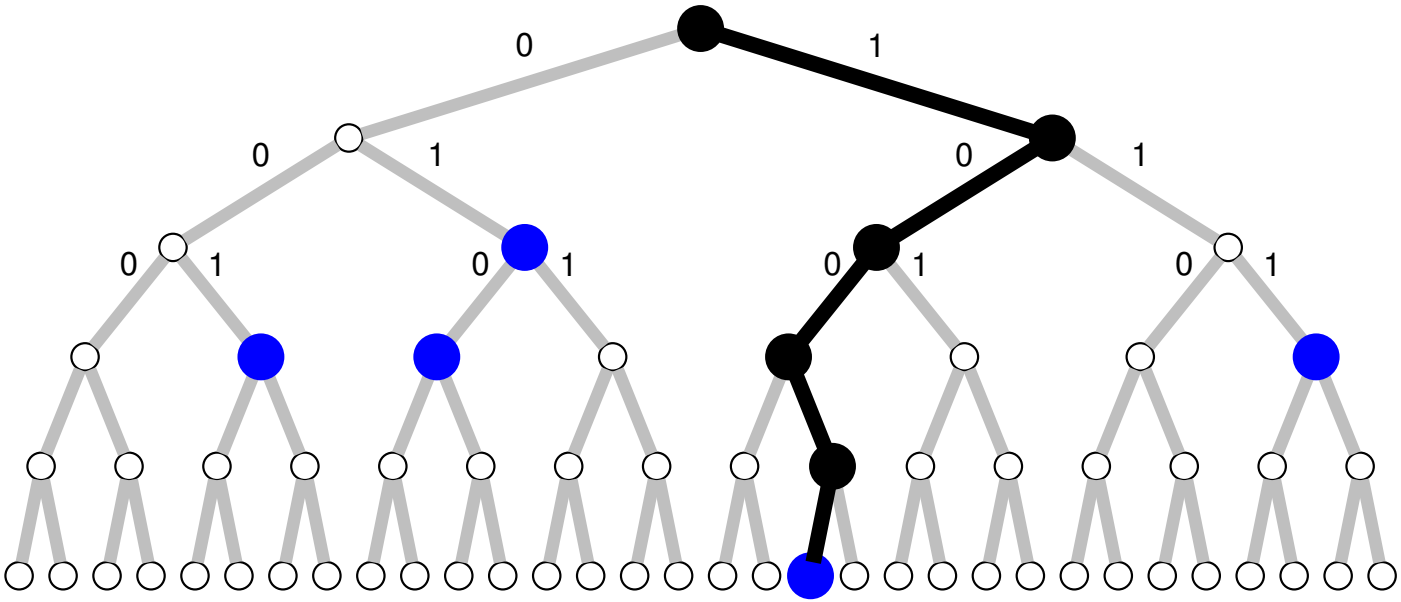
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1

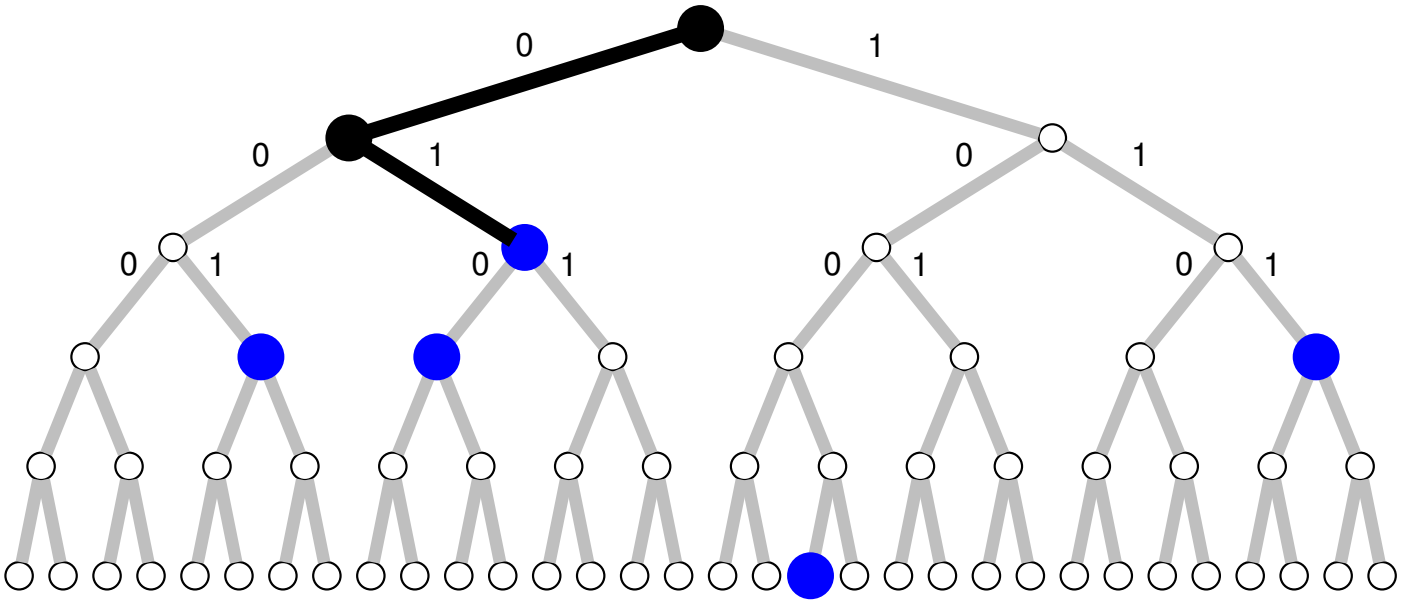
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}01100100010111$

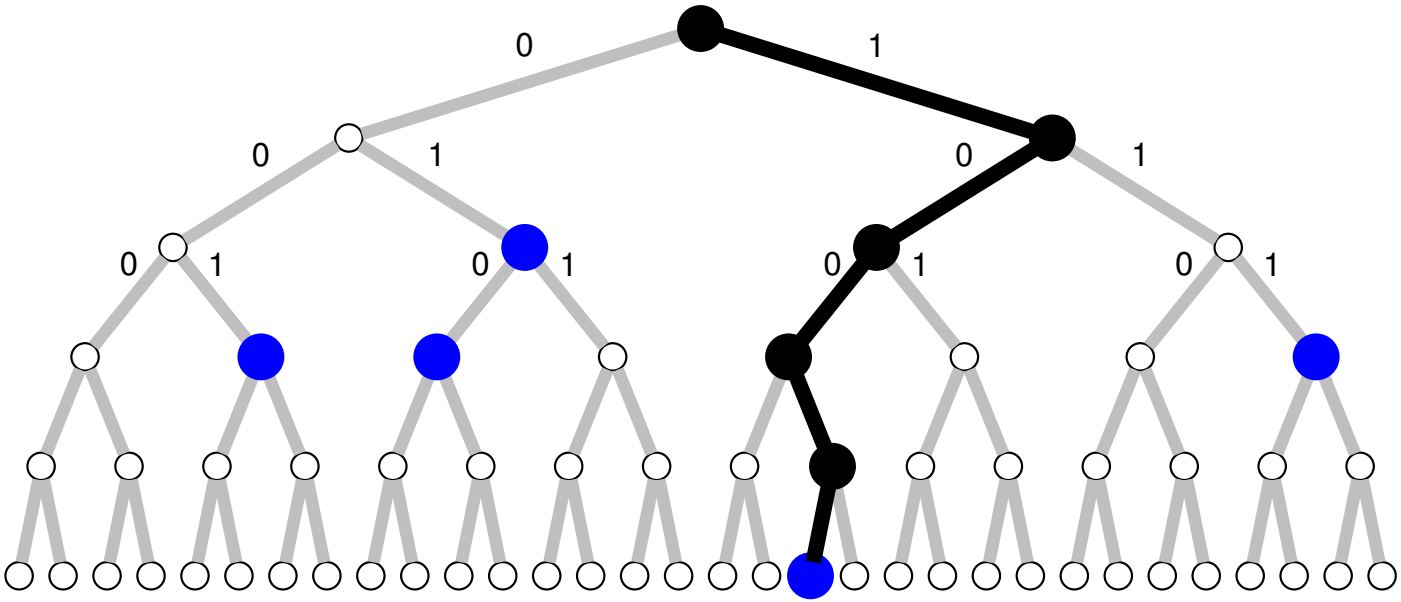
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ 10010010111$

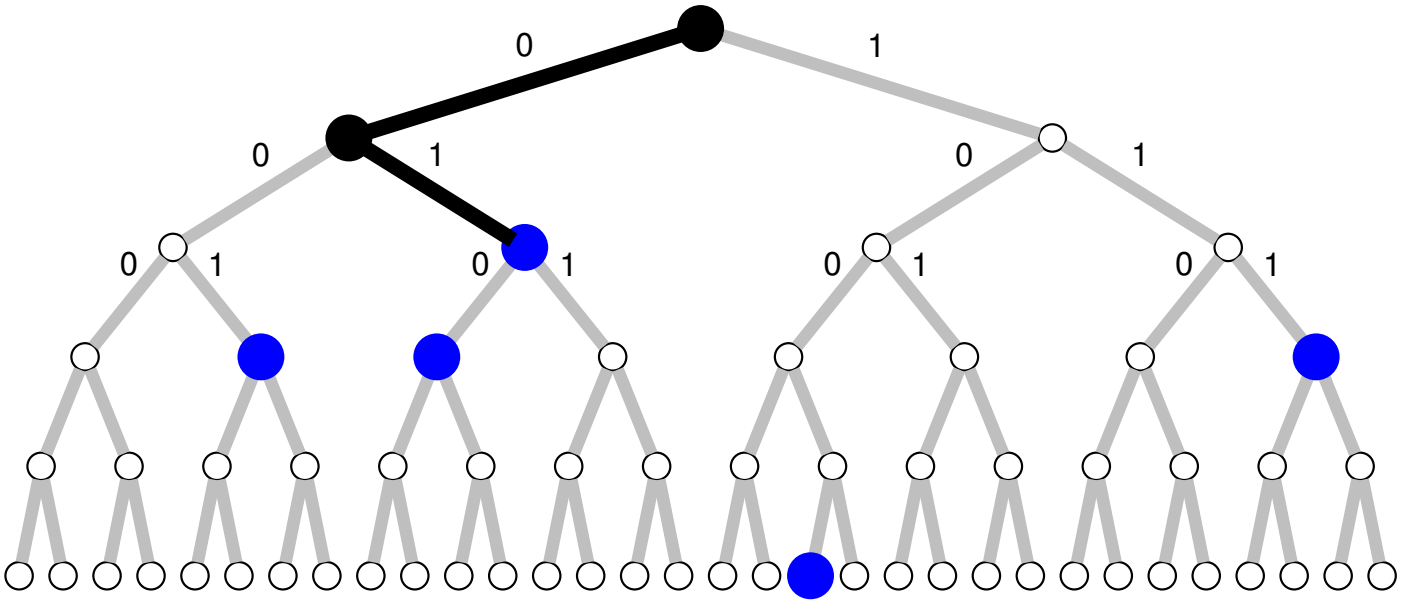
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ \underbrace{10010}010111$

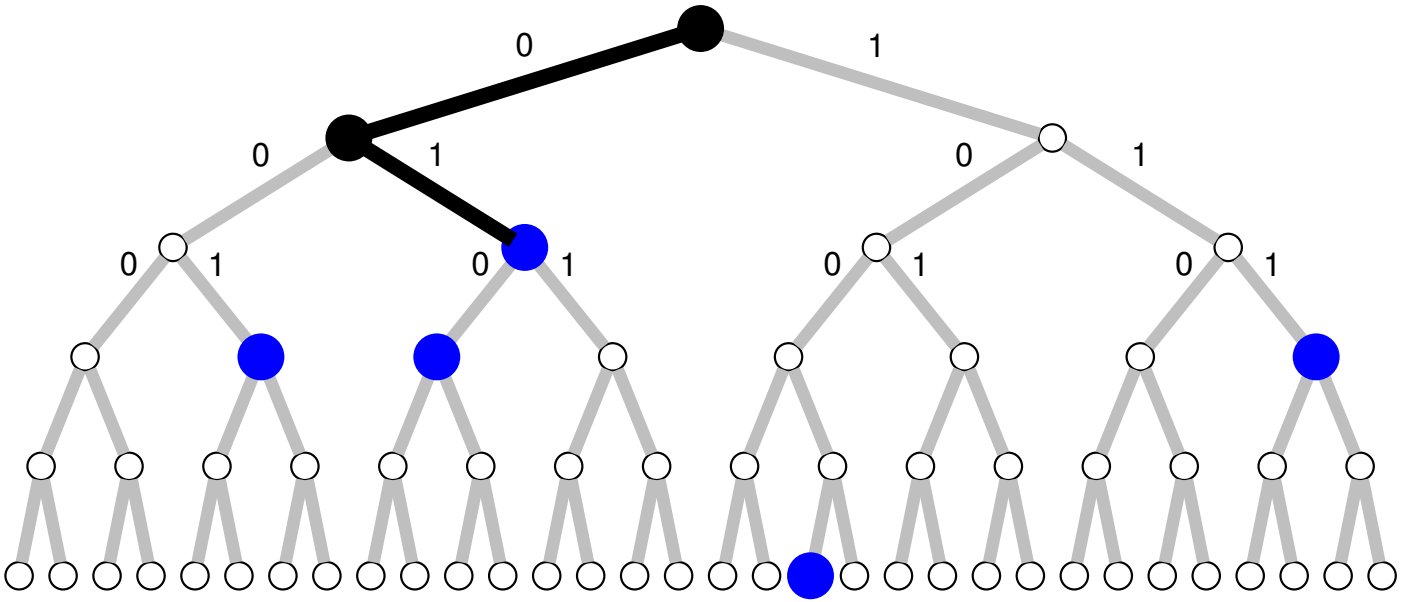
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ \underbrace{10010}\ \underbrace{01}\ 0111$

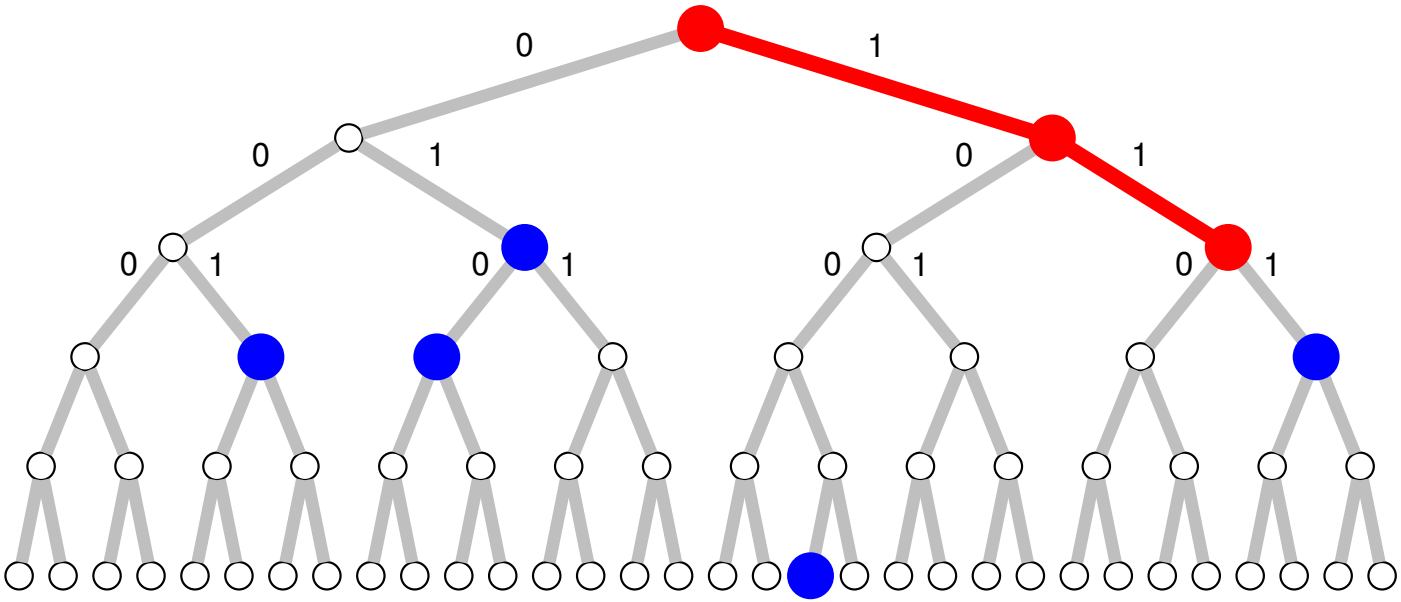
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ \underbrace{10010}\ \underbrace{01}\ \underbrace{01}\ 11$

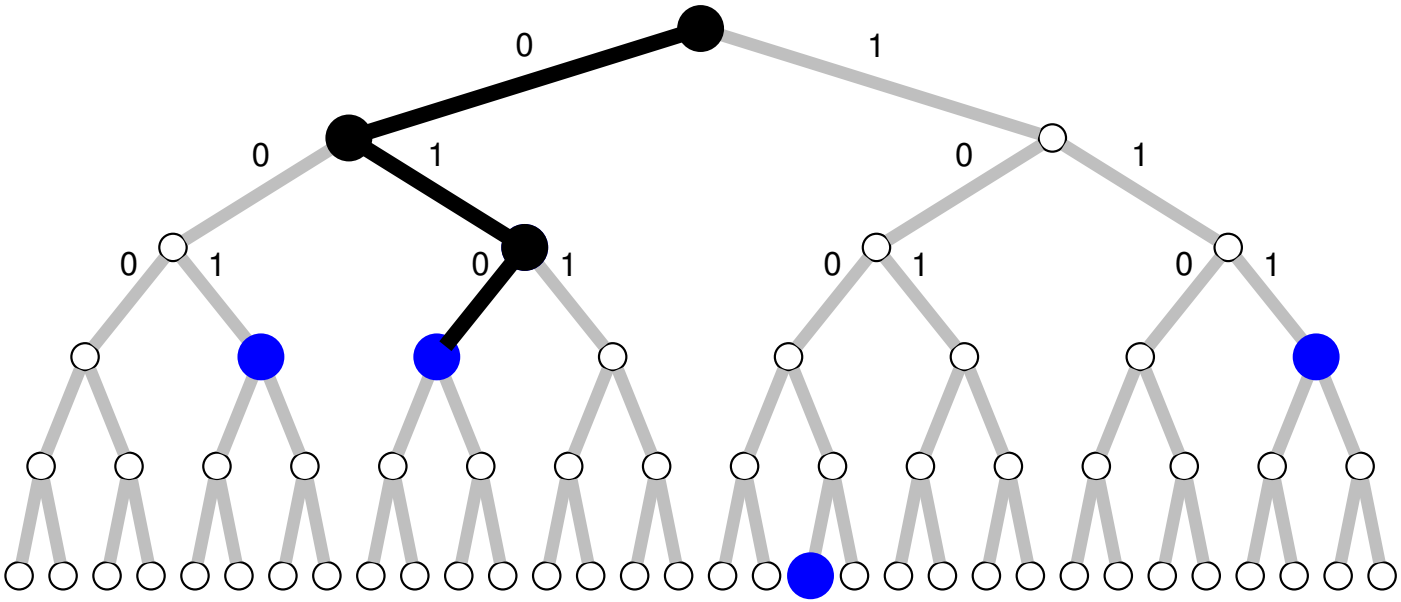
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ \underbrace{10010}\ \underbrace{01}\ \underbrace{01}\ 11$

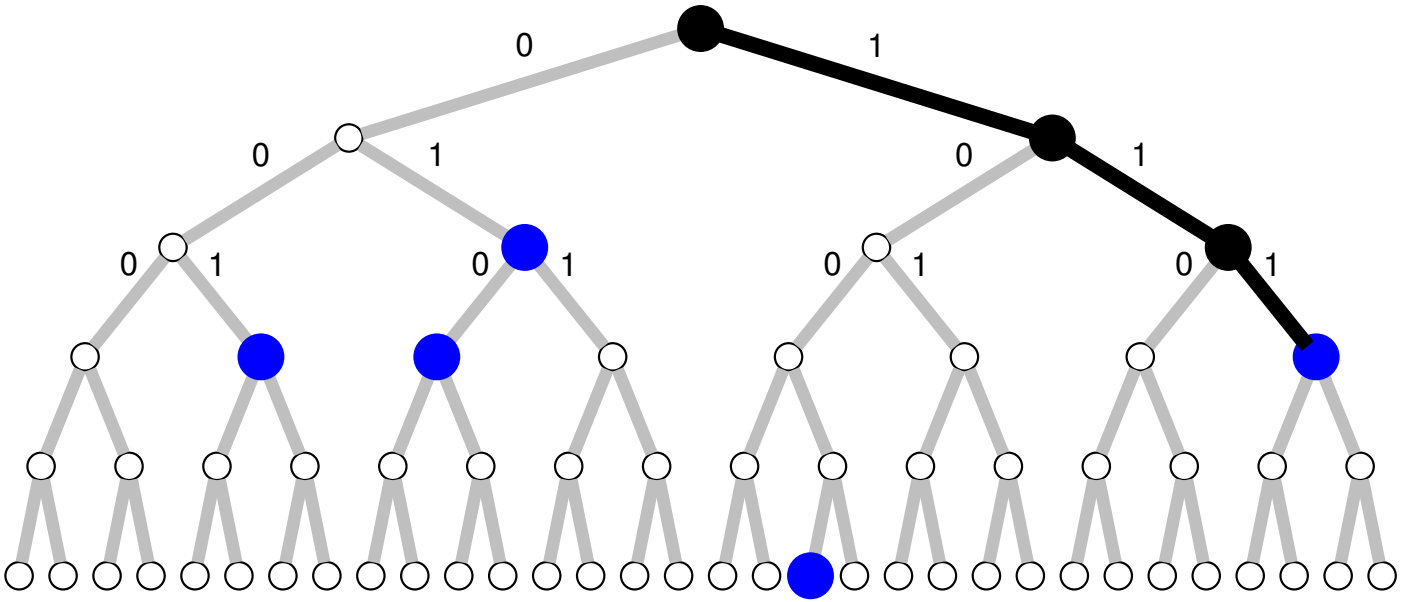
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ \underbrace{10010}\ \underbrace{010}111$

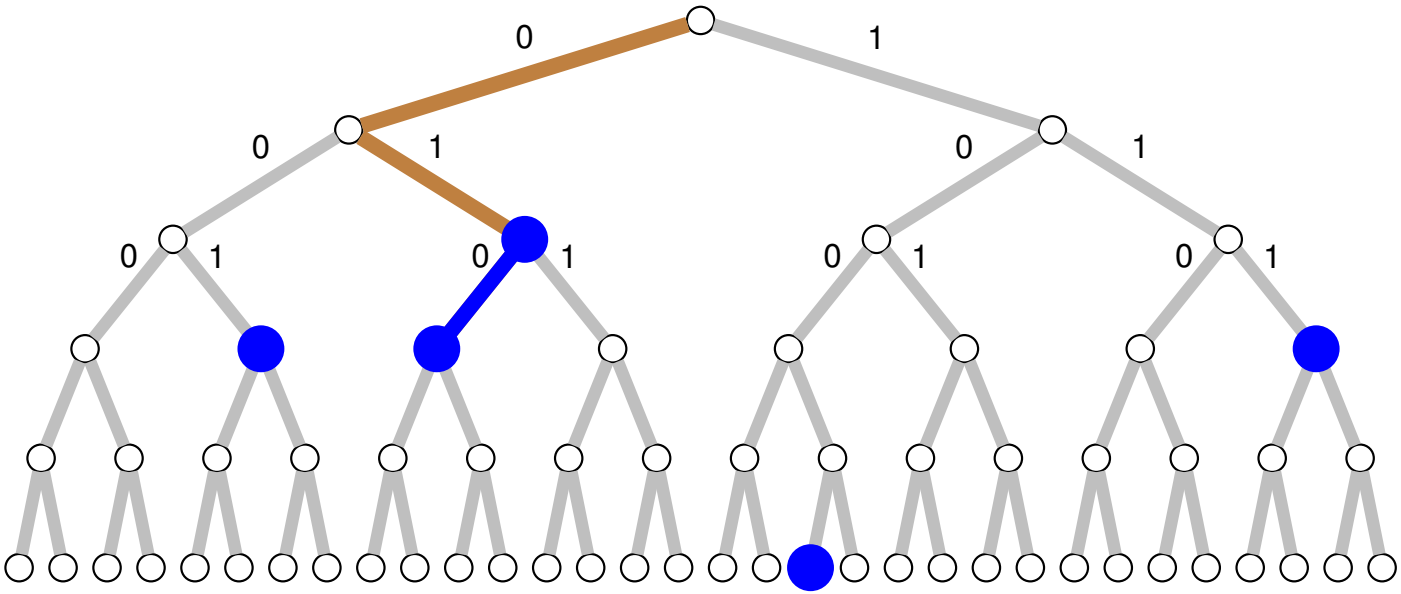
An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\underbrace{10010}\ \underbrace{01}\ \underbrace{10010}\ \underbrace{010}\ \underbrace{111}$

An Example of Non-Uniquely Decodable Code (II)



$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

01 is a prefix of 010!

An Example of Non-Uniquely Decodable Code (III)

The code is not **uniquely decodable** (i.e., the extended encoding function is not injective)

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$$\underbrace{01}\,\underbrace{001} \quad = \quad \underbrace{010}\,\underbrace{01}$$

An Example of Non-Uniquely Decodable Code (III)

The code is not **uniquely decodable** (i.e., the extended encoding function is not injective)

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$$\underbrace{0\,1}\,\underbrace{0\,0\,1} \quad = \quad \underbrace{0\,1\,0}\,\underbrace{0\,1}$$

$$\underbrace{0\,1}\,\underbrace{0\,1}\,\underbrace{0\,0\,1}\,\underbrace{0\,0\,1} \quad = \quad \underbrace{0\,1\,0}\,\underbrace{1\,0\,0\,1\,0}\,\underbrace{0\,1}$$

Sardinas-Patterson Theorem

Definition (Uniquely Decodable)

A code $\mathcal{C} \subset \mathcal{B}^*$ is **uniquely decodable** if any sequence in $\mathcal{C}^*$ splits uniquely into codewords, or equivalently, the extended encoding function $E : \mathcal{A}^* \to \mathcal{B}^*$ is injective.

Sardinas-Patterson Theorem

Definition (Uniquely Decodable)

A code $\mathcal{C} \subset \mathcal{B}^*$ is **uniquely decodable** if any sequence in $\mathcal{C}^*$ splits uniquely into codewords, or equivalently, the extended encoding function $E : \mathcal{A}^* \to \mathcal{B}^*$ is injective.

We recursively define the sets

$\mathcal{C}_1 = \mathcal{C} \diamond \mathcal{C} = \{y \in \mathcal{B}^{\geq 1} \mid \exists x \in \mathcal{C}\ xy \in \mathcal{C}\}$

$\mathcal{C}_{k+1} = \mathcal{C}_k \diamond \mathcal{C} = \{y \in \mathcal{B}^{\geq 1} \mid \exists x \in \mathcal{C}\ xy \in \mathcal{C}_k \text{ or } \exists x \in \mathcal{C}_k\ xy \in \mathcal{C}\}$

Sardinas-Patterson Theorem

Definition (Uniquely Decodable)

A code $\mathcal{C} \subset \mathcal{B}^*$ is **uniquely decodable** if any sequence in $\mathcal{C}^*$ splits uniquely into codewords, or equivalently, the extended encoding function $E : \mathcal{A}^* \to \mathcal{B}^*$ is injective.

We recursively define the sets

$$\mathcal{C}_1 = \mathcal{C} \diamond \mathcal{C} = \{y \in \mathcal{B}^{\geq 1} \mid \exists x \in \mathcal{C}\ xy \in \mathcal{C}\}$$

$$\mathcal{C}_{k+1} = \mathcal{C}_k \diamond \mathcal{C} = \{y \in \mathcal{B}^{\geq 1} \mid \exists x \in \mathcal{C}\ xy \in \mathcal{C}_k \text{ or } \exists x \in \mathcal{C}_k\ xy \in \mathcal{C}\}$$

Theorem (Sardinas-Patterson [1953])

A code $\mathcal{C}$ is uniquely decodable if and only if $\mathcal{C} \cap \left(\bigcup_{k=1}^{\infty} \mathcal{C}_k \right) = \emptyset$

▸ details...

Sardinas-Patterson Theorem: An Example

$$\mathcal{C} = \{01, 001, 010, 10010, 111\}$$

Sardinas-Patterson Theorem: An Example

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\mathcal{C}_1 = \{0\}$

Sardinas-Patterson Theorem: An Example

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\mathcal{C}_1 = \{0\}$

$\mathcal{C}_2 = \{1, 01, 10\}$

Sardinas-Patterson Theorem: An Example

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\mathcal{C}_1 = \{0\}$

$\mathcal{C}_2 = \{1, 01, 10\}$

Not a uniquely-decodable code!

Sardinas-Patterson Theorem: An Example

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\mathcal{C}_1 = \{0\}$

$\mathcal{C}_2 = \{1, 01, 10\}$

Not a uniquely-decodable code!

As $\mathcal{C}$ is finite, the sequence $\mathcal{C}_1, \mathcal{C}_2, \cdots$ becomes constant (e.g., $\emptyset$) or cyclic after a finite number of steps.

Sardinas-Patterson Theorem: An Example

$\mathcal{C} = \{01, 001, 010, 10010, 111\}$

$\mathcal{C}_1 = \{0\}$

$\mathcal{C}_2 = \{1, 01, 10\}$

Not a uniquely-decodable code!

As $\mathcal{C}$ is finite, the sequence $\mathcal{C}_1, \mathcal{C}_2, \cdots$ becomes constant (e.g., $\emptyset$) or cyclic after a finite number of steps.

In the example, $\mathcal{C}_1 \neq \mathcal{C}_2 \neq \ldots \neq \mathcal{C}_7 = \mathcal{C}_8 = \ldots$

Sardinas-Patterson Theorem: Another Example

$$\mathcal{C} = \{01, 0011, 011, 10010, 111\}$$

Sardinas-Patterson Theorem: Another Example

$\mathcal{C} = \{01, 0011, 011, 10010, 111\}$

$\mathcal{C}_1 = \{1\}$

Sardinas-Patterson Theorem: Another Example

$\mathcal{C} = \{01, 0011, 011, 10010, 111\}$

$\mathcal{C}_1 = \{1\}$

$\mathcal{C}_2 = \{0010, 11\}$

Sardinas-Patterson Theorem: Another Example

$\mathcal{C} = \{01, 0011, 011, 10010, 111\}$

$\mathcal{C}_1 = \{1\}$

$\mathcal{C}_2 = \{0010, 11\}$

$\mathcal{C}_3 = \{1\} = \mathcal{C}_1$

Sardinas-Patterson Theorem: Another Example

$$\mathcal{C} = \{01, 0011, 011, 10010, 111\}$$

$$\mathcal{C}_1 = \{1\}$$

$$\mathcal{C}_2 = \{0010, 11\}$$

$$\mathcal{C}_3 = \{1\} = \mathcal{C}_1$$

$$\bigcup_{i=1}^{\infty} \mathcal{C}_i = \mathcal{C}_1 \cup \mathcal{C}_2 = \{1, 0010, 11\}$$

Sardinas-Patterson Theorem: Another Example

$\mathcal{C} = \{01, 0011, 011, 10010, 111\}$

$\mathcal{C}_1 = \{1\}$

$\mathcal{C}_2 = \{0010, 11\}$

$\mathcal{C}_3 = \{1\} = \mathcal{C}_1$

$\bigcup_{i=1}^{\infty} \mathcal{C}_i = \mathcal{C}_1 \cup \mathcal{C}_2 = \{1, 0010, 11\}$

Empty intersection with $\mathcal{C}$

$\mathcal{C}$ is uniquely-decodable!

Prefix-Free Codes

Definition (Prefix-free)

$\mathcal{C}$ is **prefix-free** if $\mathcal{C}_1 = \emptyset$

Prefix-Free Codes

Definition (Prefix-free)

$\mathcal{C}$ is **prefix-free** if $\mathcal{C}_1 = \emptyset$

- Any prefix-free code is uniquely-decodable
- Any fixed-length code is prefix-free
- Subtrees rooted at codewords of a prefix-free code are disjoint

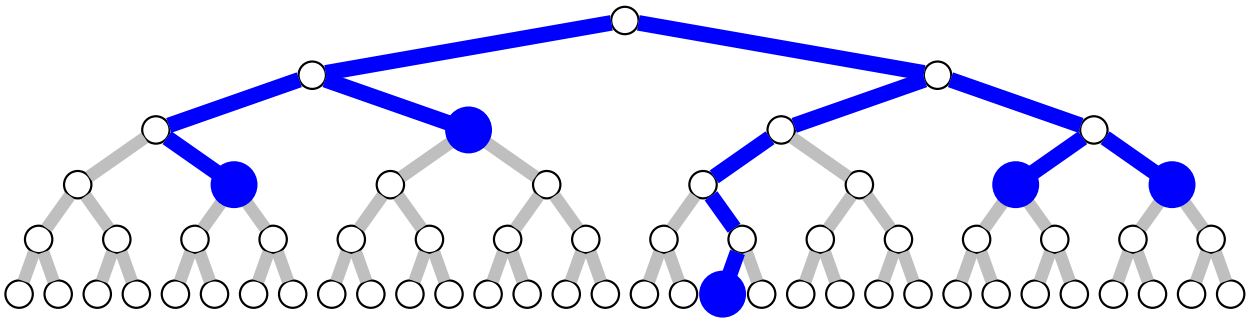
Prefix-Free Codes

Definition (Prefix-free)

$\mathcal{C}$ is **prefix-free** if $\mathcal{C}_1 = \emptyset$

- Any prefix-free code is uniquely-decodable
- Any fixed-length code is prefix-free
- Subtrees rooted at codewords of a prefix-free code are disjoint

An example: $\mathcal{C} = \{001, 01, 10010, 110, 111\}$



Outline

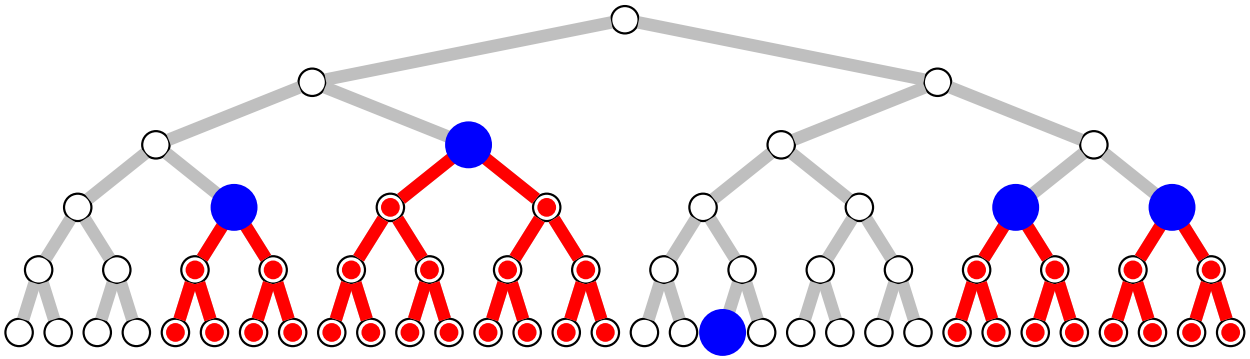
Codeword Lengths in a Prefix-Free Code

Subtrees rooted at codewords are disjoint.

Codeword Lengths in a Prefix-Free Code

Subtrees rooted at codewords are disjoint.

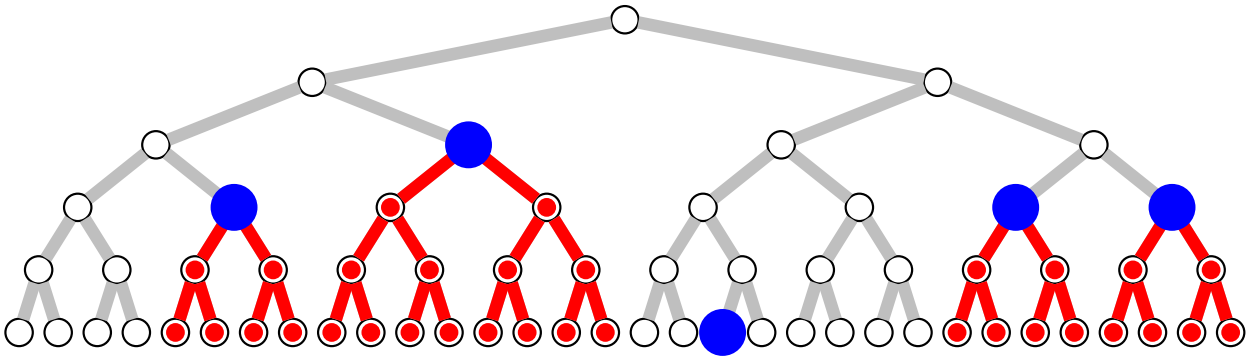
$\mathcal{C} = \{001, 01, 10010, 110, 111\}$ $l_{max} = 5$



Codeword Lengths in a Prefix-Free Code

Subtrees rooted at codewords are disjoint.

$\mathcal{C} = \{001, 01, 10010, 110, 111\}$ $l_{max} = 5$

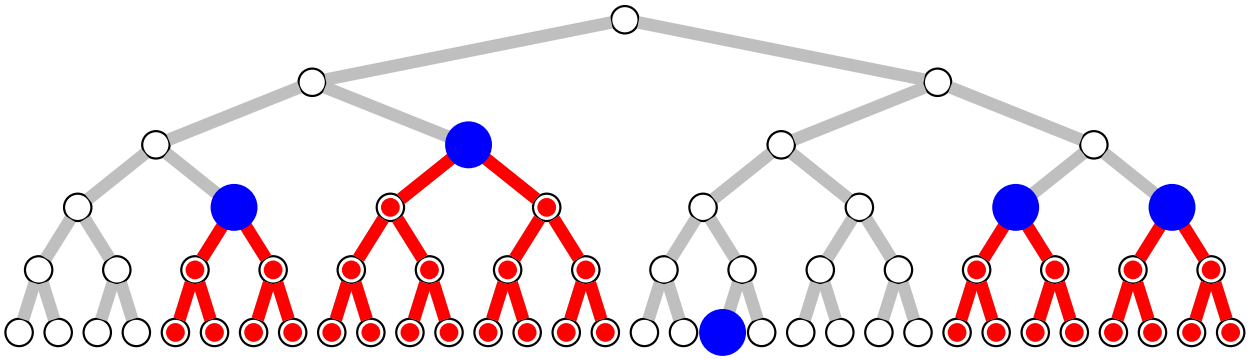


Each $w \in \mathcal{C}$ of length $l(w)$ corresponds to $r^{l_{max}-l(w)}$ leaves

Codeword Lengths in a Prefix-Free Code

Subtrees rooted at codewords are disjoint.

$\mathcal{C} = \{001, 01, 10010, 110, 111\}$ $\quad l_{max} = 5$



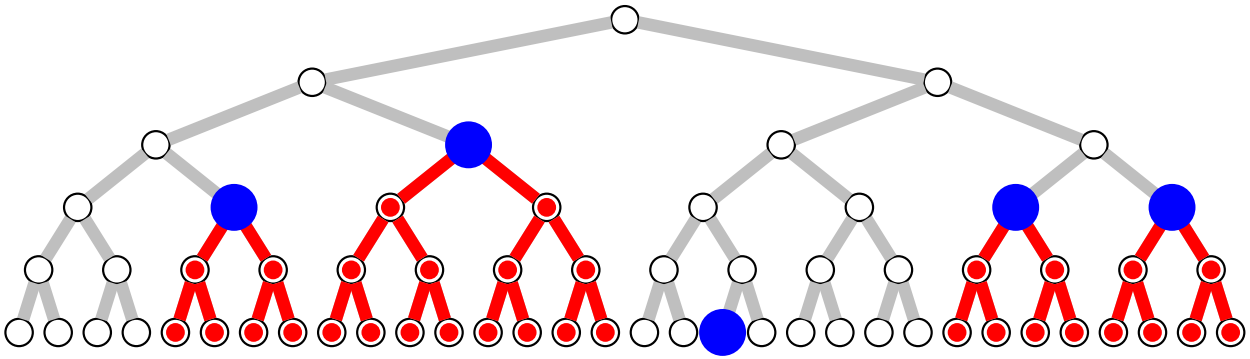
Each $w \in \mathcal{C}$ of length $l(w)$ corresponds to $r^{l_{max}-l(w)}$ leaves

$$\sum_{w \in \mathcal{C}} r^{l_{max}-l(w)} \leq r^{l_{max}}$$ where $l(w)$ is the length of w

Codeword Lengths in a Prefix-Free Code

Subtrees rooted at codewords are disjoint.

$\mathcal{C} = \{001, 01, 10010, 110, 111\}$ $\quad l_{max} = 5$



Each $w \in \mathcal{C}$ of length $l(w)$ corresponds to $r^{l_{max}-l(w)}$ leaves

$$\sum_{w \in \mathcal{C}} r^{l_{max}-l(w)} \leq r^{l_{max}}$$ where $l(w)$ is the length of w

or $$\sum_{w \in \mathcal{C}} r^{-l(w)} \leq 1$$

Kraft's Inequality (I)

Theorem (Kraft's Inequality [1949])

There exists a prefix-free r-ary code $\mathcal{C}$ with codewords of respective lengths $l_1, \ldots, l_n$ if and only if

$$\sum_{i=1}^{n} r^{-l_i} \leq 1$$

Kraft's Inequality (I)

Theorem (Kraft's Inequality [1949])

There exists a prefix-free r-ary code $\mathcal{C}$ with codewords of respective lengths $l_1, \ldots, l_n$ if and only if

$$\sum_{i=1}^{n} r^{-l_i} \leq 1$$

- the "only if" part comes from the ideas in last slide
- the proof of the "if" part is constructive (next slide)

Kraft's Inequality (II)

Given $l_1 \leq \ldots \leq l_n$ such that $S = \sum_{i=1}^{n} r^{-l_i} \leq 1$ define the increasing sequence $x_k = \sum_{i=1}^{k-1} r^{-l_i}$.

Observe that $x_k r^{l_k}$ is an integer and $x_k r^{l_k} < S r^{l_k} \leq r^{l_k}$.

Kraft's Inequality (II)

Given $l_1 \leq \ldots \leq l_n$ such that $S = \sum_{i=1}^{n} r^{-l_i} \leq 1$ define the increasing sequence $x_k = \sum_{i=1}^{k-1} r^{-l_i}$.

Observe that $x_k r^{l_k}$ is an integer and $x_k r^{l_k} < S r^{l_k} \leq r^{l_k}$.

Now, assign to each l_k the codeword $w_k \in \{0, \ldots, r-1\}^{l_k}$ obtained as the r-ary representation of $x_k r^{l_k}$.

Kraft's Inequality (II)

Given $l_1 \leq \ldots \leq l_n$ such that $S = \sum_{i=1}^{n} r^{-l_i} \leq 1$ define the increasing sequence $x_k = \sum_{i=1}^{k-1} r^{-l_i}$.

Observe that $x_k r^{l_k}$ is an integer and $x_k r^{l_k} < S r^{l_k} \leq r^{l_k}$.

Now, assign to each l_k the codeword $w_k \in \{0, \ldots, r-1\}^{l_k}$ obtained as the r-ary representation of $x_k r^{l_k}$.

The resulting code is prefix-free. Indeed, to be w_k a prefix of w_m, necessarily $k < m \leq n$ and $(x_m - x_k) r^{l_m} < r^{l_m - l_k}$, which is not possible since $x_m - x_k \geq x_{k+1} - x_k = r^{-l_k}$.

McMillan's Inequality (I)

Theorem (McMillan's Inequality [1956])

There exists a uniquely-decodable r-ary code $\mathcal{C}$ with codewords of respective lengths $l_1, \ldots, l_n$ if and only if

$$\sum_{i=1}^{n} r^{-l_i} \leq 1$$

McMillan's Inequality (I)

Theorem (McMillan's Inequality [1956])

There exists a uniquely-decodable r-ary code $\mathcal{C}$ with codewords of respective lengths $l_1, \ldots, l_n$ if and only if

$$\sum_{i=1}^{n} r^{-l_i} \leq 1$$

- the "if" part is a direct consequence of Kraft's inequality
- the proof of the "only if" is more elaborated (see next slide)

McMillan's Inequality (II)

Let's assume the existence of a uniquely-decodable r-ary code $\mathcal{C}$ with codeword lengths $l_1 \leq \ldots \leq l_n$ and define $S = \sum_{i=1}^{n} r^{-l_i}$.

McMillan's Inequality (II)

Let's assume the existence of a uniquely-decodable r-ary code $\mathcal{C}$ with codeword lengths $l_1 \leq \ldots \leq l_n$ and define $S = \sum_{i=1}^{n} r^{-l_i}$.

As $\mathcal{C}$ is uniquely-decodable then all concatenations of k codewords are different. Now, in the code $\mathcal{C}^k$ for some $k > 1$

$$S_k = \sum_{i_i=1}^{n} \cdots \sum_{i_k=1}^{n} r^{-(l_{i_1}+\ldots+l_{i_k})} = \left(\sum_{i_1=1}^{n} r^{-l_{i_1}}\right) \cdots \left(\sum_{i_k=1}^{n} r^{-l_{i_k}}\right) = S^k$$

McMillan's Inequality (II)

Let's assume the existence of a uniquely-decodable r-ary code $\mathcal{C}$ with codeword lengths $l_1 \leq \ldots \leq l_n$ and define $S = \sum_{i=1}^{n} r^{-l_i}$.

As $\mathcal{C}$ is uniquely-decodable then all concatenations of k codewords are different. Now, in the code $\mathcal{C}^k$ for some $k > 1$

$$S_k = \sum_{i_i=1}^{n} \cdots \sum_{i_k=1}^{n} r^{-(l_{i_1}+\ldots+l_{i_k})} = \left(\sum_{i_1=1}^{n} r^{-l_{i_1}}\right) \cdots \left(\sum_{i_k=1}^{n} r^{-l_{i_k}}\right) = S^k$$

Let $n_j^{(k)}$ be the number of codewords of length j in $\mathcal{C}^k$, which fulfils $n_j^{(k)} \leq r^j$. Then, $S^k = \sum_{j=kl_1}^{kl_n} n_j^{(k)} r^{-j} \leq kl_n - kl_1 + 1$.

McMillan's Inequality (II)

Let's assume the existence of a uniquely-decodable r-ary code $\mathcal{C}$ with codeword lengths $l_1 \leq \ldots \leq l_n$ and define $S = \sum_{i=1}^{n} r^{-l_i}$.

As $\mathcal{C}$ is uniquely-decodable then all concatenations of k codewords are different. Now, in the code $\mathcal{C}^k$ for some $k > 1$

$$S_k = \sum_{i_i=1}^{n} \cdots \sum_{i_k=1}^{n} r^{-(l_{i_1}+\ldots+l_{i_k})} = \left(\sum_{i_1=1}^{n} r^{-l_{i_1}}\right) \cdots \left(\sum_{i_k=1}^{n} r^{-l_{i_k}}\right) = S^k$$

Let $n_j^{(k)}$ be the number of codewords of length j in $\mathcal{C}^k$, which fulfils $n_j^{(k)} \leq r^j$. Then, $S^k = \sum_{j=kl_1}^{kl_n} n_j^{(k)} r^{-j} \leq kl_n - kl_1 + 1$.

Thus, $\ln S \leq \dfrac{\ln(kl_n - kl_1 + 1)}{k}$ and in the limit $k \to \infty$ we obtain $S \leq 1$.

Final Remarks

- Prefix-free codes perform as well as uniquely-decodable ones . . .
- The larger the arity is, the easier it is to accommodate a given codeword length sequence to obtain a prefix-free code

Final Remarks

- Prefix-free codes perform as well as uniquely-decodable ones but decoding prefix-free codes is simpler
- The larger the arity is, the easier it is to accommodate a given codeword length sequence to obtain a prefix-free code

Final Remarks

- Prefix-free codes perform as well as uniquely-decodable ones but decoding prefix-free codes is simpler
- The larger the arity is, the easier it is to accommodate a given codeword length sequence to obtain a prefix-free code

Final Remarks

- Prefix-free codes perform as well as uniquely-decodable ones but decoding prefix-free codes is simpler
- The larger the arity is, the easier it is to accommodate a given codeword length sequence to obtain a prefix-free code

Some questions about optimization:

- Which is the smallest arity of the code for a given length sequence?
- Which is the shortest code for given input alphabet and arity?
- **How can we optimize the code length if the input symbols occur with different probabilities?**

Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

END OF PART II

Sardinas-Patterson Theorem: Details

Assume $\mathcal{C}$ is not uniquely decodable. Then there exists a minimal $S_0 \in \mathcal{C}^{\geq 1}$ such that $S_0 = x_1\, x_2 \cdots x_k = y_1\, y_2 \cdots y_l$

Sardinas-Patterson Theorem: Details

Assume $\mathcal{C}$ is not uniquely decodable. Then there exists a minimal $S_0 \in \mathcal{C}^{\geq 1}$ such that $S_0 = x_1\, x_2 \cdots x_k = y_1\, y_2 \cdots y_l$ which implies that $y_1 = x_1 s_1$ or $x_1 = y_1 s_1$ for some $s_1 \in \mathcal{C}_1$.

Sardinas-Patterson Theorem: Details

Assume $\mathcal{C}$ is not uniquely decodable. Then there exists a minimal $S_0 \in \mathcal{C}^{\geq 1}$ such that $S_0 = x_1\, x_2 \cdots x_k = y_1\, y_2 \cdots y_l$ which implies that $y_1 = x_1 s_1$ or $x_1 = y_1 s_1$ for some $s_1 \in \mathcal{C}_1$.

Remove the shortest initial codeword (say x_1) and define S_1 such that $S_0 = x_1 S_1$. Then $S_1 = x_2 \cdots x_k = s_1\, y_2 \cdots y_l$

Sardinas-Patterson Theorem: Details

Assume $\mathcal{C}$ is not uniquely decodable. Then there exists a minimal $S_0 \in \mathcal{C}^{\geq 1}$ such that $S_0 = x_1\, x_2 \cdots x_k = y_1\, y_2 \cdots y_l$ which implies that $y_1 = x_1 s_1$ or $x_1 = y_1 s_1$ for some $s_1 \in \mathcal{C}_1$.

Remove the shortest initial codeword (say x_1) and define S_1 such that $S_0 = x_1 S_1$. Then $S_1 = x_2 \cdots x_k = s_1\, y_2 \cdots y_l$

Similarly, $s_1 = x_2 s_2$ or $x_2 = s_1 s_2$ for some $s_2 \in \mathcal{C}_2$. Then, remove the shortest of s_1 and x_2. The remaining sequence S_2 is either $S_2 = x_3 \cdots x_k = s_2\, y_2 \cdots y_l$ or $S_2 = s_2\, x_3 \cdots x_k = y_2 \cdots y_l$.

Sardinas-Patterson Theorem: Details

Assume $\mathcal{C}$ is not uniquely decodable. Then there exists a minimal $S_0 \in \mathcal{C}^{\geq 1}$ such that $S_0 = x_1\, x_2 \cdots x_k = y_1\, y_2 \cdots y_l$ which implies that $y_1 = x_1 s_1$ or $x_1 = y_1 s_1$ for some $s_1 \in \mathcal{C}_1$.

Remove the shortest initial codeword (say x_1) and define S_1 such that $S_0 = x_1 S_1$. Then $S_1 = x_2 \cdots x_k = s_1\, y_2 \cdots y_l$

Similarly, $s_1 = x_2 s_2$ or $x_2 = s_1 s_2$ for some $s_2 \in \mathcal{C}_2$. Then, remove the shortest of s_1 and x_2. The remaining sequence S_2 is either $S_2 = x_3 \cdots x_k = s_2\, y_2 \cdots y_l$ or $S_2 = s_2\, x_3 \cdots x_k = y_2 \cdots y_l$.

Continue until at some point $S_i = x_j = s_i$ or $S_i = s_i = y_j$. Then, $\mathcal{C} \cap \mathcal{C}_i \neq \emptyset$.

go back...