Encoding One Symbol at a Time

- To begin with, let's think about encoding one symbol X_i at a time, using a fixed code that defines a mapping of each source symbol into a finite sequence of code symbols called a *codeword*.
(Later on we will consider encoding blocks of symbols together.)
- We will encode a sequence of source symbols X by concatenating the codewords of each.
- This is called a *symbol code*.
- E.g. source alphabet is $\mathcal{A}_X = \{C, G, T, A\}$. One possible code:
$C \rightarrow 0; \quad G \rightarrow 10; \quad T \rightarrow 110; \quad A \rightarrow 1110$
So we would have $CCAT \rightarrow 001110110$.
- We require that the mapping be such that we can *decode* this sequence, no matter what the original symbols were.

Notation for Sequences & Codes

- $\mathcal{A}_X$ and $\mathcal{A}_Z$ are the source and code alphabets.
- $\mathcal{A}_X^+$ and $\mathcal{A}_Z^+$ denote sequences of one or more symbols from the source or code alphabets.
- A symbol code, C, is a mapping $\mathcal{A}_X \to \mathcal{A}_Z^+$. We use $c(x)$ to denote the codeword to which C maps x.
- We use concatenation to extend this to a mapping for the *extended code*, $C^+ : \mathcal{A}_X^+ \to \mathcal{A}_Y^+$:

$$c^+(x_1 x_2 \cdots x_N) = c(x_1)c(x_2)\cdots c(x_N)$$

  i.e., we code a string of symbols by just stringing together the codes for each symbol.
- I'll sometimes also use C to denote the set of all legal codewords: $\{w \mid w = C(a) \text{ for some } a \in \mathcal{A}_X\}$.

Uniquely Decodable & Instantaneous Codes

- A code is *uniquely decodable* if the mapping $C^+ : \mathcal{A}_X^+ \rightarrow \mathcal{A}_Z^+$ is one-to-one, i.e. $\forall\ \mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{A}_X^+$, $\mathbf{x} \neq \mathbf{x}' \Rightarrow c^+(\mathbf{x}) \neq c^+(\mathbf{x}')$
- A code is obviously not uniquely decodable if two symbols have the same codeword — ie, if $c(a_i) = c(a_j)$ for some $i \neq j$ — so we'll usually assume that this isn't the case.
- A code is *instantaneously decodable* if any source sequences $\mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{A}^+$ for which $\mathbf{x}$ is not a prefix of $\mathbf{x}'$ have encodings $\mathbf{z} = C(\mathbf{x})$ and $\mathbf{z}' = C(\mathbf{x}')$ for which $\mathbf{z}$ is not a prefix of $\mathbf{z}'$.
  Otherwise, after receiving $\mathbf{z}$, we wouldn't yet know whether the message starts with $\mathbf{z}$ or with $\mathbf{z}'$.
- Instantaneous codes are also called *prefix-free codes* or just *prefix codes*.

What Codes are Decodable?

- We only want to consider codes that can be successfully decoded.
- To define what that means, we need to set some rules of the game:
  1. How does the channel terminate the transmission?
     (e.g. it could explicitly mark the end, it could send only 0s after the end, it could send random garbadge after the end,...)
  2. How soon do we require a decoded symbol to be known?
     (e.g. "instantaneously" – as soon as the codeword for the symbol is received, within a fixed delay of when its codeword is received, not until the entire message has been received,...)
- Easiest case: assume the end of the transmission is explicitly marked, and don't require any symbols to be decoded until the entire transmission has been received.
- Hardest case: require instantaneous decoding, and thus it doesn't matter what happens at the end of the transmission.

	Code A	Code B	Code C	Code D
a	10	0	0	0
b	11	10	01	01
c	111	110	011	11

Code A: Not uniquely decodable
Both bbb and cc encode as 111111

Code B: Instantaneously decodable
End of each codeword marked by 0

Code C: Decodable with one-symbol delay
End of codeword marked by *following* 0

Code D: Uniquely decodable, but with unbounded delay:
011111111111111 decodes as $accccccc$
01111111111111 decodes as $bcccccc$

Existence of Codes

- Since we hope to compress data, we would like codes that are uniquely decodable and whose codewords are short.
- Also, we'd like to use instantaneous codes where possible since they are easiest and most efficient to decode.
- If we could make all the codewords really short, life would be really easy. Too easy. Why?
  Because there are only a few possible short codewords and we can't reuse them or else our code wouldn't be decodable.
- Instead, making some codewords short will require that other codewords be long, if the code is to be uniquely decodable.
- **Question 1:** What sets of codeword lengths are possible?
- **Question 2:** Can we always manage to use instantaneous codes?

McMillan's Inequality

- There is a uniquely decodable binary code with codewords having lengths $l_1, \ldots, l_I$ if and only if

$$\sum_{i=1}^{I} \frac{1}{2^{l_i}} \leq 1$$

- E.g. there is a uniquely decodable binary code with lengths $1, 2, 3, 3$, since

$$1/2 + 1/4 + 1/8 + 1/8 = 1$$

- An example of such a code is $\{0, 01, 011, 111\}$.
- There is *no* uniquely decodable binary code with lengths $2, 2, 2, 2, 2$, since

$$1/4 + 1/4 + 1/4 + 1/4 + 1/4 > 1$$

Kraft's Inequality

- There is an instantaneous binary code with codewords having lengths $l_1, \ldots, l_I$ if and only if

$$\sum_{i=1}^{I} \frac{1}{2^{l_i}} \leq 1$$

- This is exactly the same condition as McMillan's inequality!
- E.g. there is an instantaneous binary code with lengths $1, 2, 3, 3$, since

$$1/2 + 1/4 + 1/8 + 1/8 = 1$$

- An example of such a code is $\{0,\ 10,\ 110,\ 111\}$.
- There is an instantaneous binary code with lengths $2, 2, 2$, since

$$1/4 + 1/4 + 1/4 < 1$$

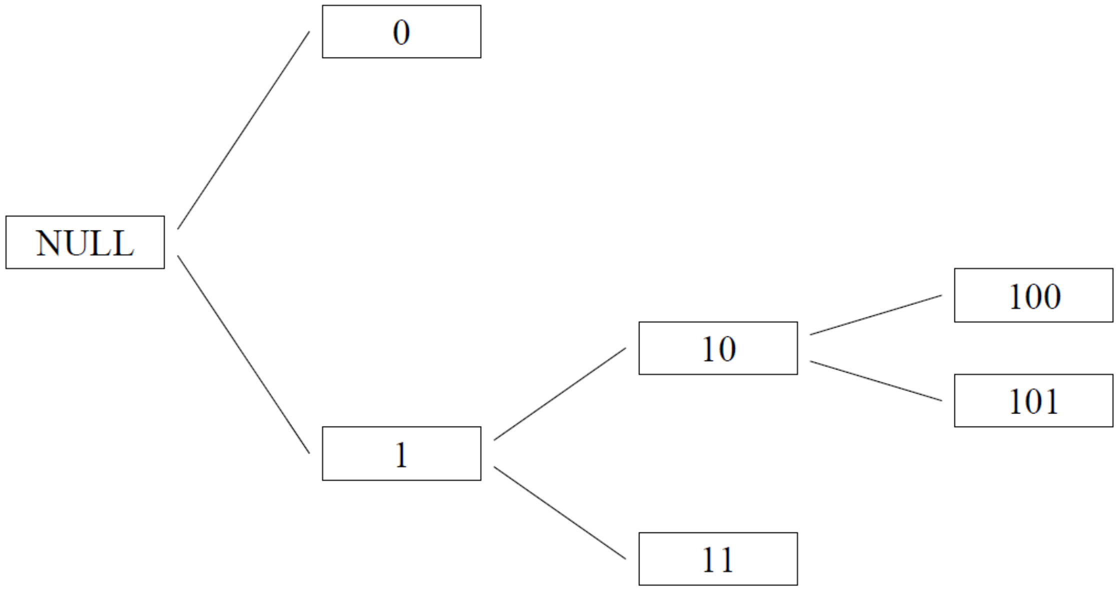
- An example of such a code is $\{00,\ 10,\ 01\}$.

We Can Always Use Instantaneous Codes

- Since instantaneous codes are a proper subset of uniquely decodable codes, we might have expected that the condition for existence of a u.d. code to be less stringent than that for instantaneous codes.
- But combining Kraft's and McMillan's inequalities, we conclude that there is an instantaneous binary code with lengths $l_1, \ldots, l_I$ if and only if there is a uniquely decodable code with these lengths.
- **Implication:** There is probably no practical benefit to using uniquely decodable codes that aren't instantaneous.
- **Happy consequence:** We don't have to worry about how the encoding is terminated (if at all) or about decoding delays (at least for symbol codes; for block codes this will change).

Visualizing Prefix Codes as Trees

- We can view codewords of an instantaneous (prefix) code as leaves of a tree.
- The root represents the null string; each level corresponds to adding another code symbol.
- Here is the tree for a code with codewords 0, 11, 100, 101:



Building an Instantaneous Code

- Let the lengths of the codewords be {1,2,3,3}.
- First check: $2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} \leq 1.$
- Our final code can be read from the leaf nodes: {1,00,010,011}.