# Data Compression 3:
# Arithmetic Coding

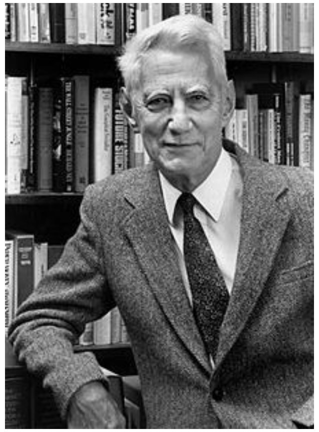
*Fang-Lue Zhang*

# The problem: encoding data succinctly

- Opportunity #1: some symbols are used more

- Claude Shannon proved (1940's) there's a way to transmit symbol strings from alphabet *X* with an average of *H(X)* bits/symbol, called the *entropy:*

$$H(X) = \sum_i P_i \log_2 \frac{1}{P_i}$$



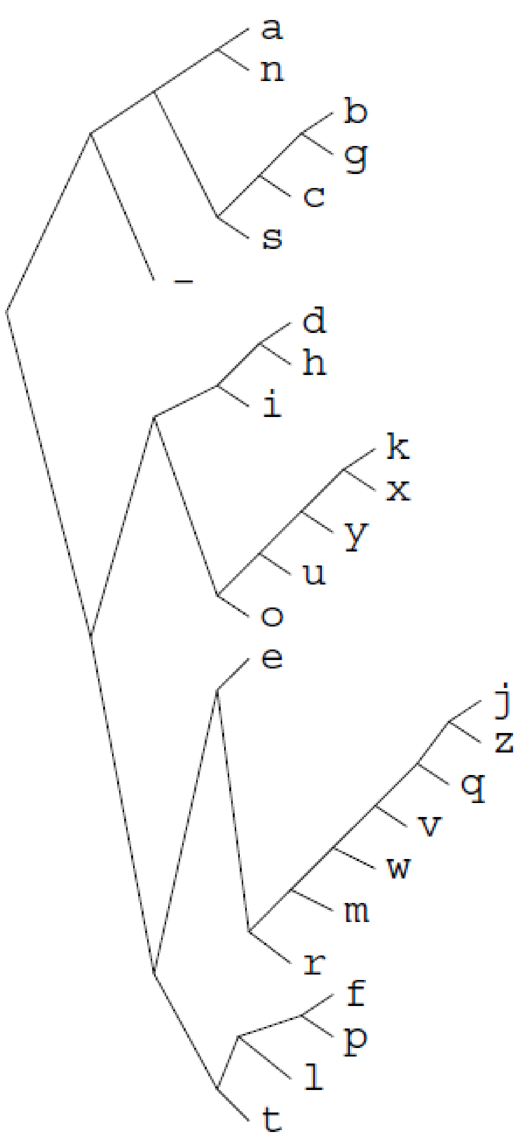
- He showed it was possible, but not how to do it!
- Huffman Coding gets quite close

| $x$ | $P(x)$ |
|---|---|
| a | 0.0575 |
| b | 0.0128 |
| c | 0.0263 |
| d | 0.0285 |
| e | 0.0913 |
| f | 0.0173 |
| g | 0.0133 |
| h | 0.0313 |
| i | 0.0599 |
| j | 0.0006 |
| k | 0.0084 |
| l | 0.0335 |
| m | 0.0235 |
| n | 0.0596 |
| o | 0.0689 |
| p | 0.0192 |
| q | 0.0008 |
| r | 0.0508 |
| s | 0.0567 |
| t | 0.0706 |
| u | 0.0334 |
| v | 0.0069 |
| w | 0.0119 |
| x | 0.0073 |
| y | 0.0164 |
| z | 0.0007 |
| – | 0.1928 |

# Huffman recap

- send each symbol as soon as it occurs (*symbol* code)
- optimal, given this restriction
- but wastes bits
- drop the restriction? (→ *stream* codes)

| $a_i$ | $p_i$ | $\log_2 \frac{1}{p_i}$ | $l_i$ | $c(a_i)$ |
|---|---|---|---|---|
| a | 0.0575 | 4.1 | 4 | 0000 |
| b | 0.0128 | 6.3 | 6 | 001000 |
| c | 0.0263 | 5.2 | 5 | 00101 |
| d | 0.0285 | 5.1 | 5 | 10000 |
| e | 0.0913 | 3.5 | 4 | 1100 |
| f | 0.0173 | 5.9 | 6 | 111000 |
| g | 0.0133 | 6.2 | 6 | 001001 |
| h | 0.0313 | 5.0 | 5 | 10001 |
| i | 0.0599 | 4.1 | 4 | 1001 |
| j | 0.0006 | 10.7 | 10 | 1101000000 |
| k | 0.0084 | 6.9 | 7 | 1010000 |
| l | 0.0335 | 4.9 | 5 | 11101 |
| m | 0.0235 | 5.4 | 6 | 110101 |
| n | 0.0596 | 4.1 | 4 | 0001 |
| o | 0.0689 | 3.9 | 4 | 1011 |
| p | 0.0192 | 5.7 | 6 | 111001 |
| q | 0.0008 | 10.3 | 9 | 110100001 |
| r | 0.0508 | 4.3 | 5 | 11011 |
| s | 0.0567 | 4.1 | 4 | 0011 |
| t | 0.0706 | 3.8 | 4 | 1111 |
| u | 0.0334 | 4.9 | 5 | 10101 |
| v | 0.0069 | 7.2 | 8 | 11010001 |
| w | 0.0119 | 6.4 | 7 | 1101001 |
| x | 0.0073 | 7.1 | 7 | 1010001 |
| y | 0.0164 | 5.9 | 6 | 101001 |
| z | 0.0007 | 10.4 | 10 | 1101000001 |
| – | 0.1928 | 2.4 | 2 | 01 |



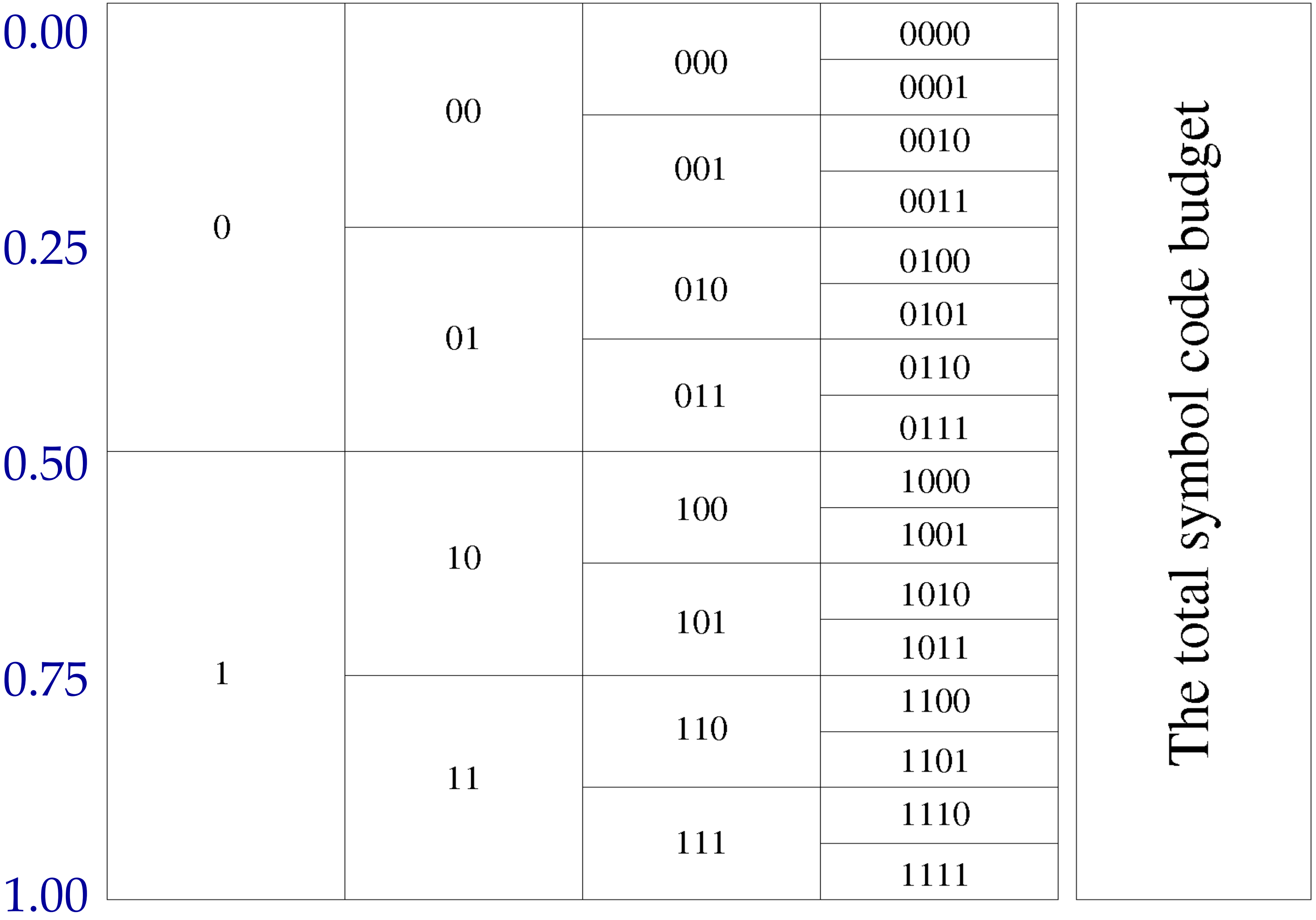
# The problem: encoding data succinctly

- Opportunity #1: some symbols are used more
- Opportunity #2: the sequence isn't random
  - → Lempel-Ziv
  - → **Arithmetic Coding,** based on rather different ideas

- *reaches* the Shannon limit, for random ordered symbols, and
- **in conjunction with a predictive language model,** it does better still

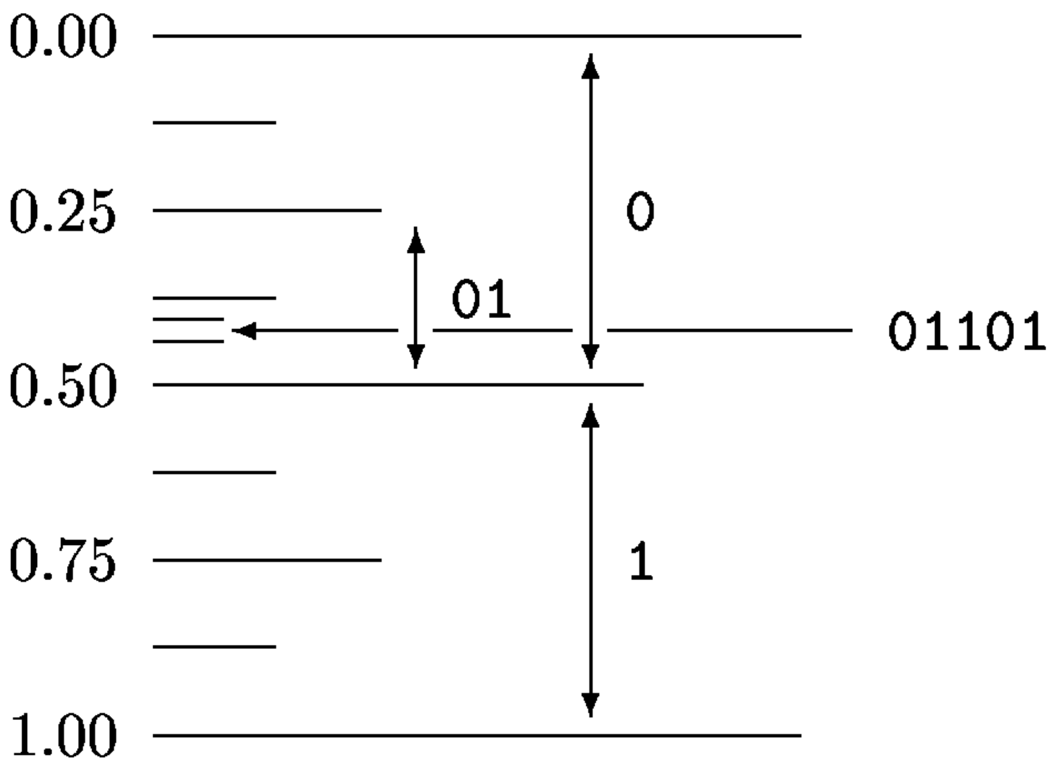
| $x$ | $P(x)$ |
|---|---|
| a | 0.0575 |
| b | 0.0128 |
| c | 0.0263 |
| d | 0.0285 |
| e | 0.0913 |
| f | 0.0173 |
| g | 0.0133 |
| h | 0.0313 |
| i | 0.0599 |
| j | 0.0006 |
| k | 0.0084 |
| l | 0.0335 |
| m | 0.0235 |
| n | 0.0596 |
| o | 0.0689 |
| p | 0.0192 |
| q | 0.0008 |
| r | 0.0508 |
| s | 0.0567 |
| t | 0.0706 |
| u | 0.0334 |
| v | 0.0069 |
| w | 0.0119 |
| x | 0.0073 |
| y | 0.0164 |
| z | 0.0007 |
| – | 0.1928 |

# The problem: encoding data succinctly



# ...and think of intervals as bit-strings

- the interval corresponding to $n$-bits has width $1/2^n$
- to specify interval of size $\alpha$, we will need about $\log_2 1/\alpha$ bits




eg: if $\alpha$=**1/8**
we need
$\log_2 1/\alpha$ = **3 bits**

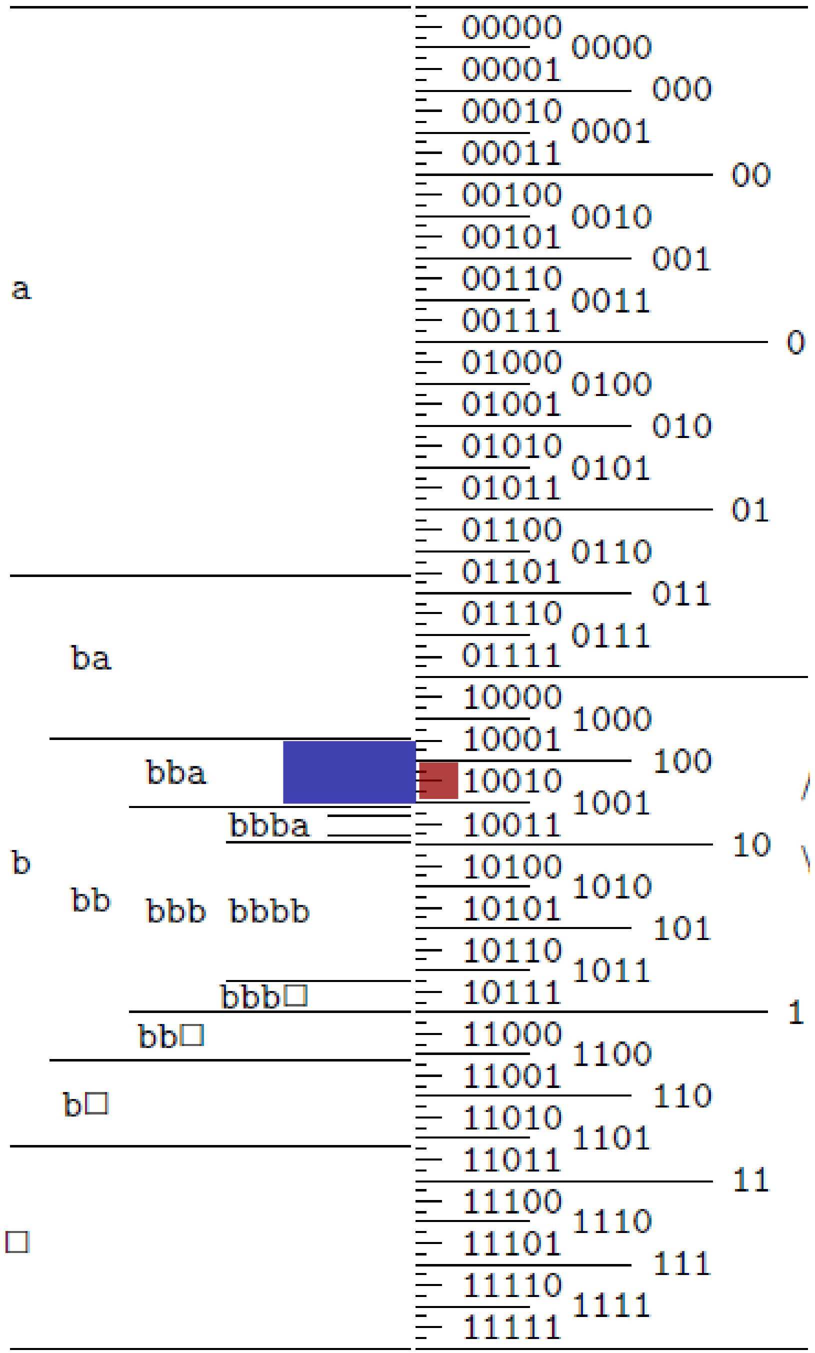
next slide considers sending symbols in a simple alphabet of just {a,b, □}

# To send symbol string, **send interval** (as bit-string)

- To send a string, I recursively partition up the interval [0,1] into segments...
  (but don't worry about the partitioning scheme just yet!)

- I send you the binary string that corresponds to the **largest interval** ***enclosed by the string I want to send***.

- You should be able to *decode* this, provided you use the same scheme for partitioning as I did!



# On-the-fly encoding: transmitting bbba□

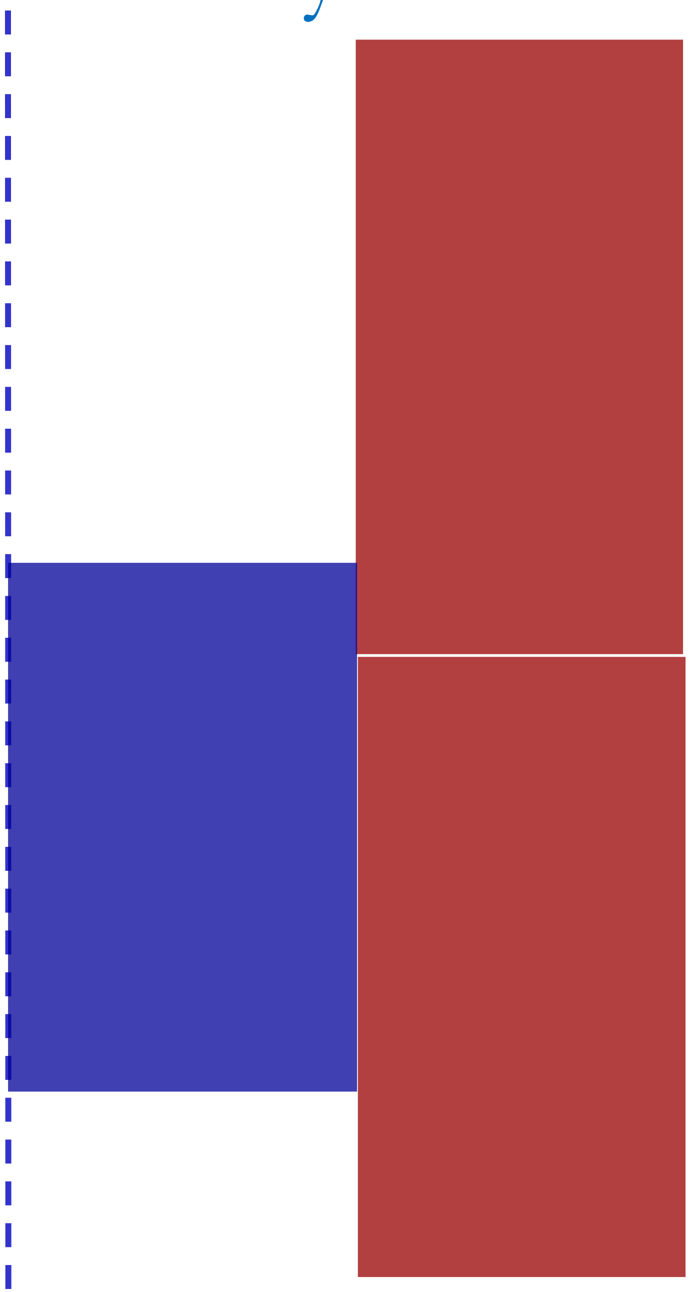
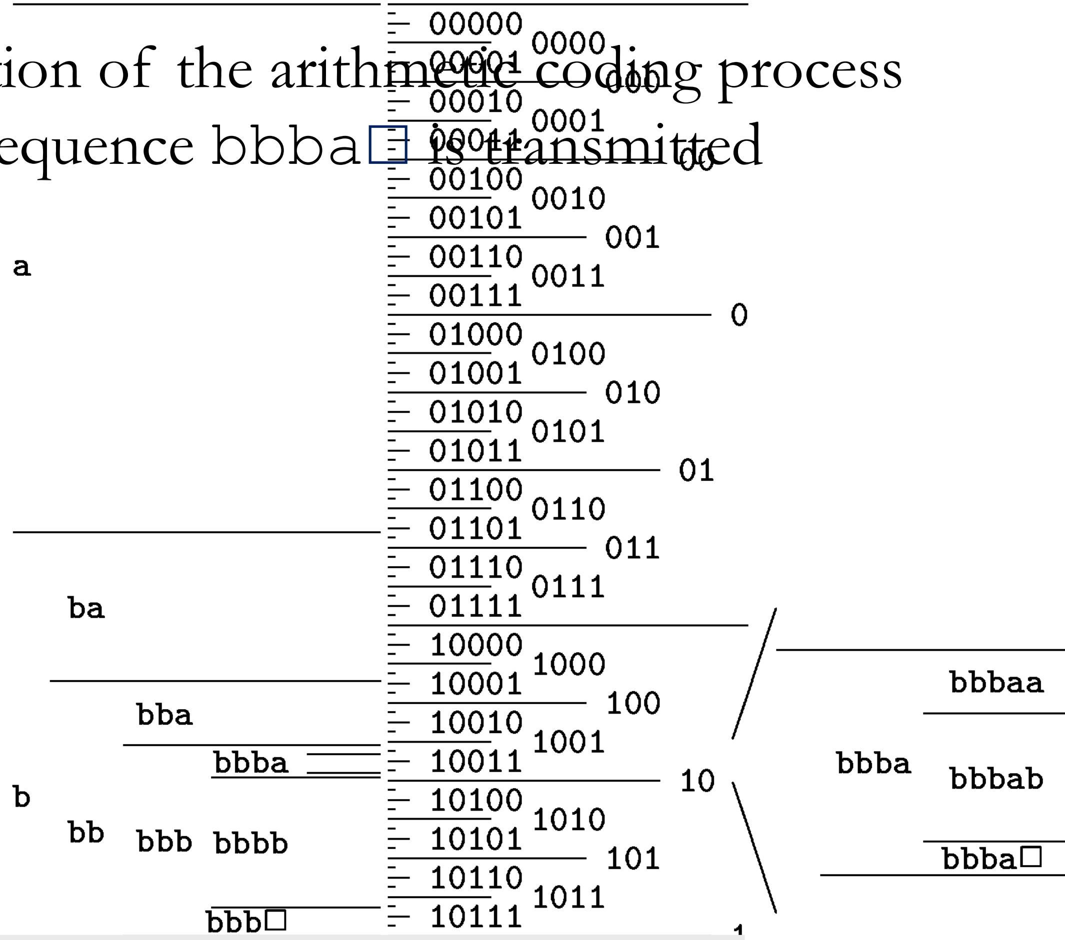


Illustration of the arithmetic coding process as the sequence bbba□ is transmitted



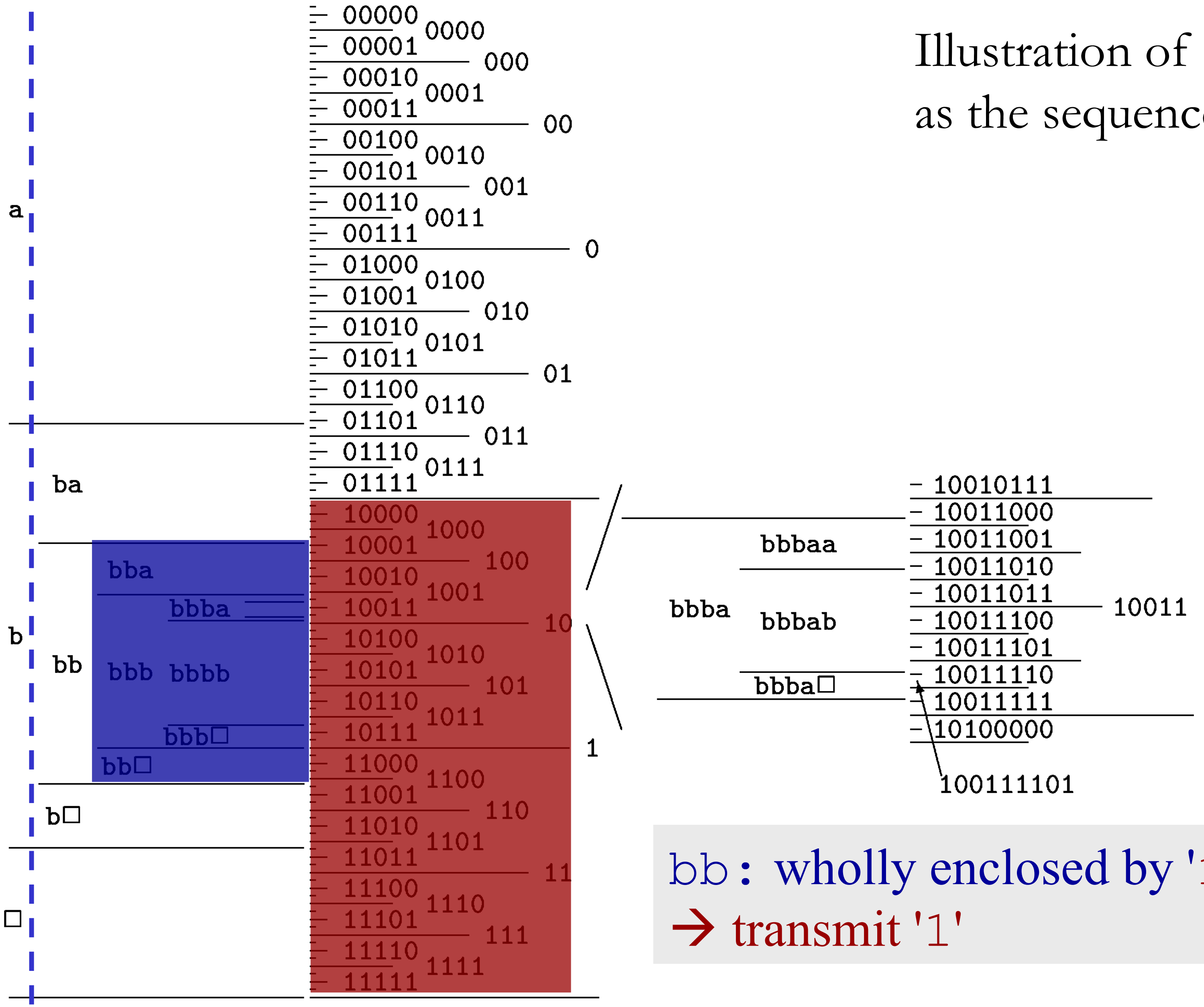

b : not wholly enclosed by 0 or 1

(i.e. could be 01, 10, or 11)

→Don't transmit anything yet

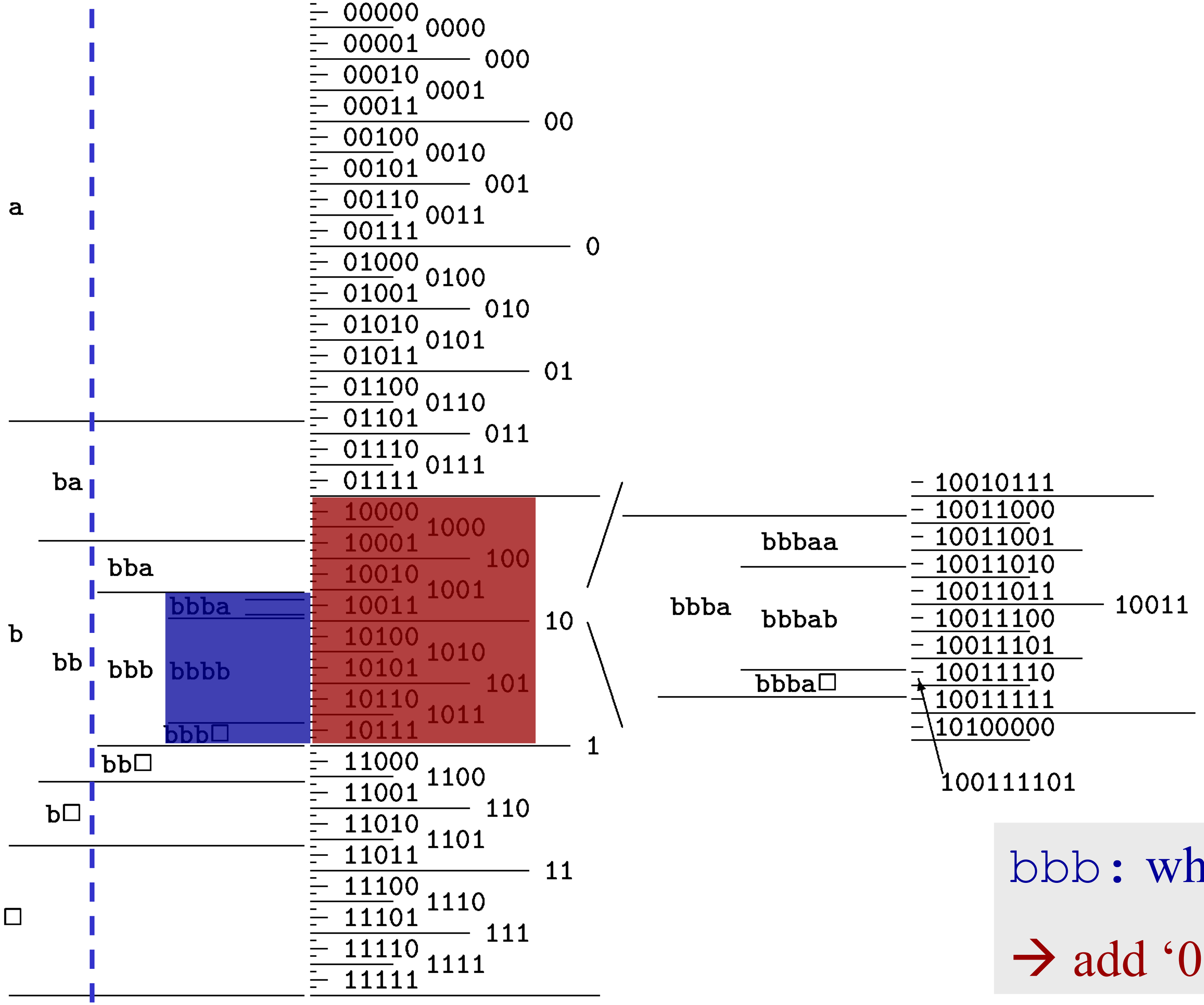
# On-the-fly encoding: transmitting bbba□

Illustration of the arithmetic coding process as the sequence bbba□ is transmitted




bb : wholly enclosed by '1' range,
→ transmit '1'

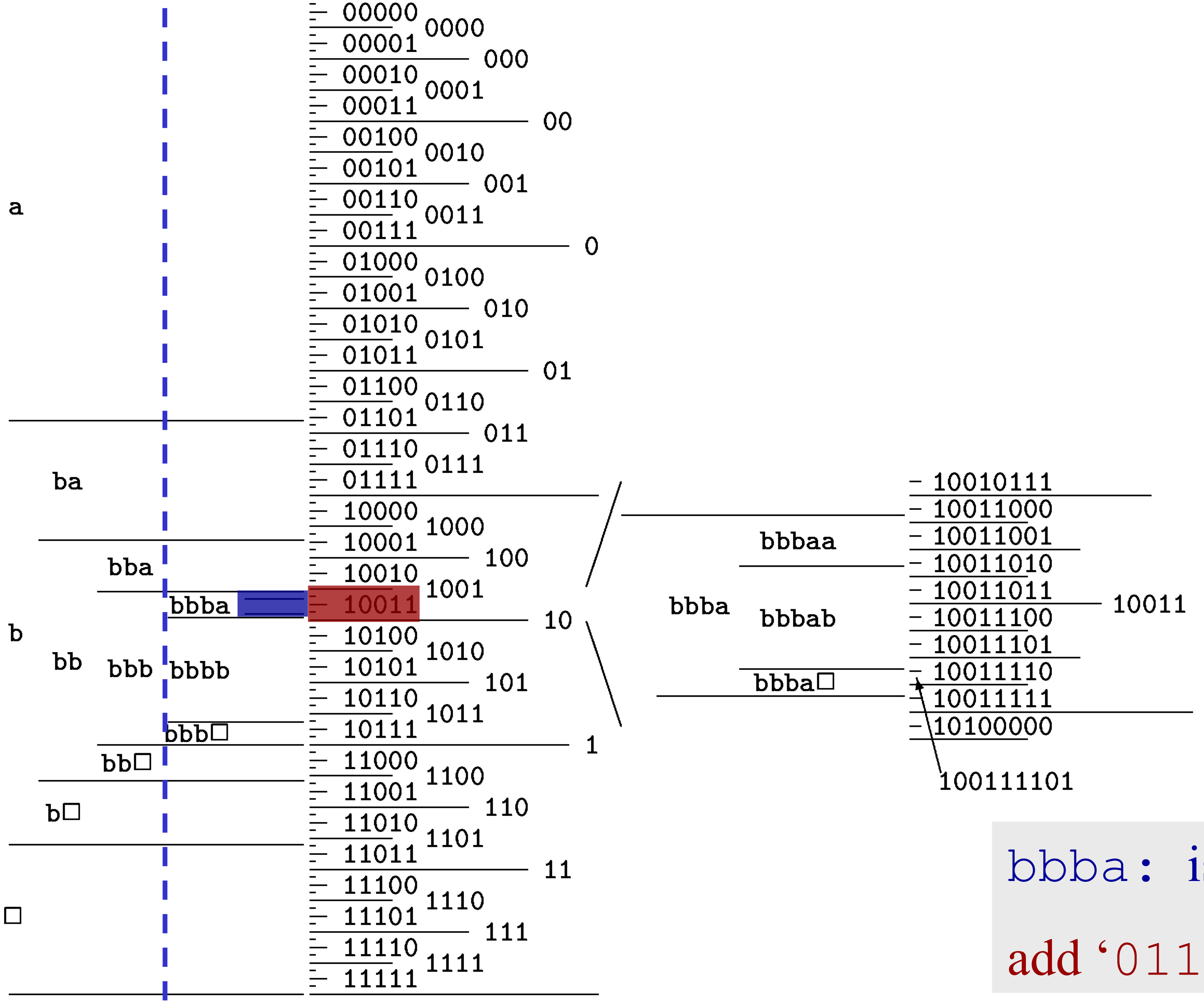
# On-the-fly encoding: transmitting bbba .




bbb : wholly within 10, so

→ add ‘0’ to the transmission

# On-the-fly encoding: transmitting bbba .




bbba : is within 10011, so

add ‘011' to the transmission

# On-the-fly decoding:

The first '1' arrives.

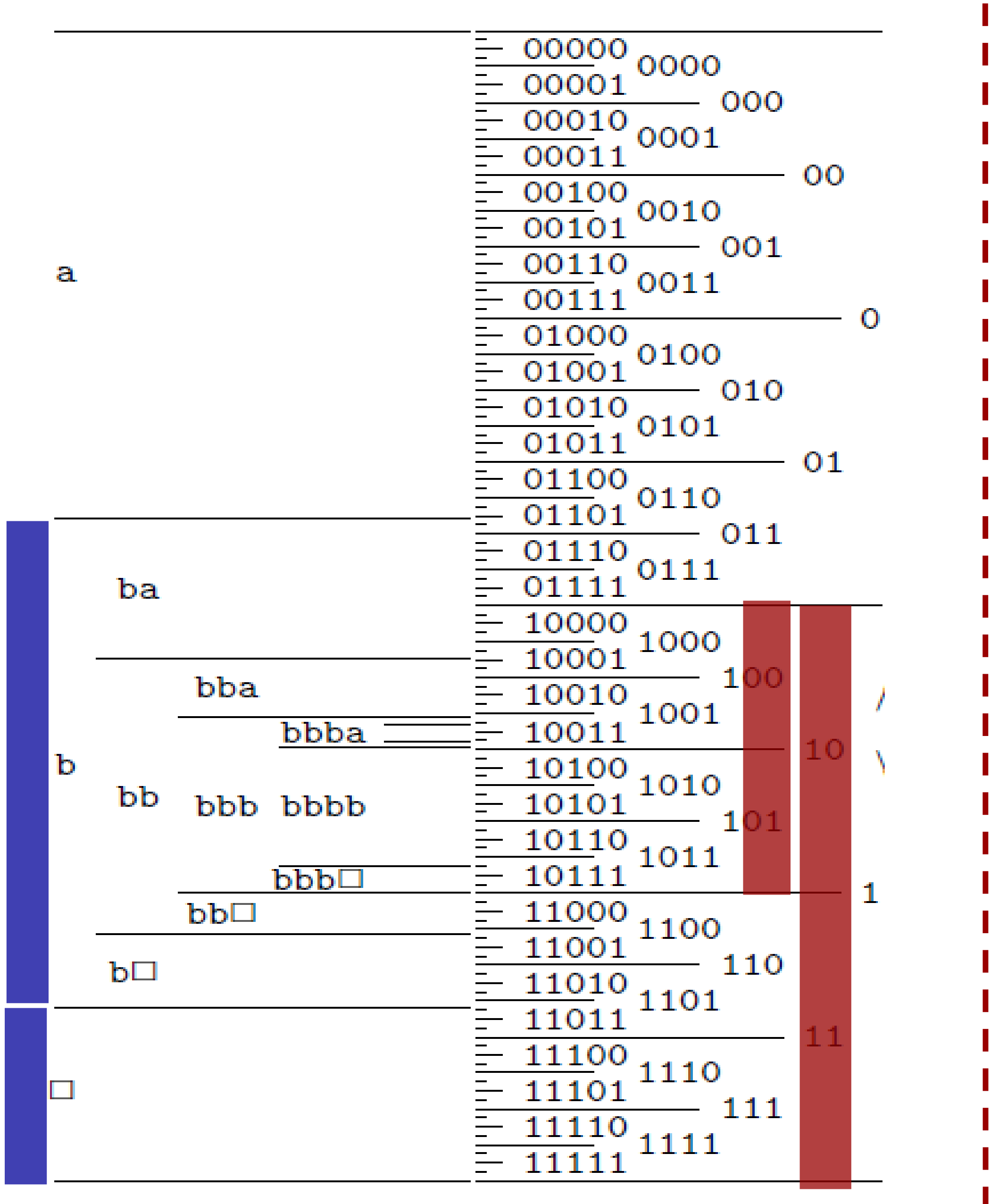
Could be b, or □.

Don't emit anything yet

'10' has arrived

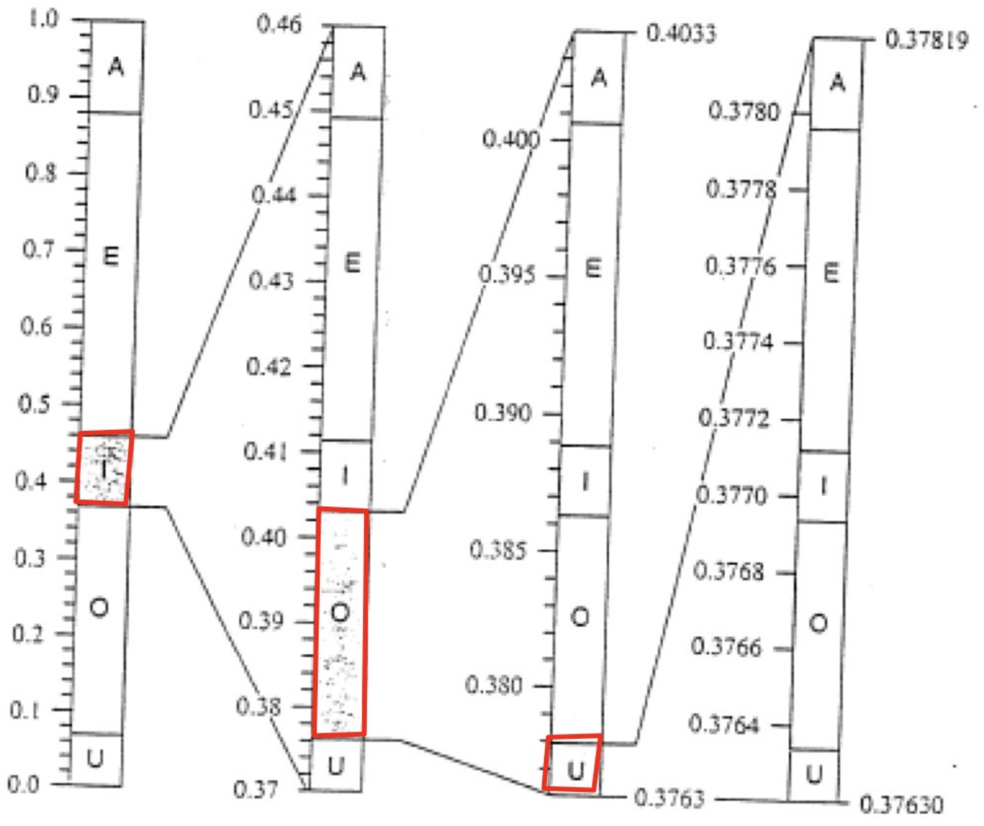
this is wholly enclosed by the 'b' interval, so now we can safely emit ‘b’

……



# A "vowellish" example

| Symbols | Probabilities | Optimal # Bits log2(1/Pi) |
|---|---|---|
| a | 0.12 | 3.06 |
| e | 0.42 | 1.25 |
| I | 0.09 | 3.47 |
| o | 0.3 | 1.74 |
| u | 0.07 | 3.84 |




To send "iou": Send any interval C within [0.37630 , 0.37819)

Using a binary fraction of 0.011000001 (9 bits)

(It would be 10 bits in Huffman coding)

**This example is from the book of Numerical Recipes**

# What's the best partitioning scheme?

- suppose our scheme gives string **S** an interval of size $\alpha_s$
- this is going to require $\log_2 1/\alpha_s$ bits
- expected message length will be $\sum_s P_s \log_2 \frac{1}{\alpha_s}$
- If we set $\alpha_s = P_s$ this matches the Shannon limit!
  (and any other scheme is worse)

So *this is the code that Shannon knew must exist*!

# What's the best partitioning for an entire string?

- thought: is there a recursive way to do the partitioning, which gives the right "real estate" to a whole **string**, not just individual symbols?
- remarkably, yes!
- based on the recursive "chain rule" of probabilities...

$$P(s_1, s_2) = P(s_1)P(s_2 \mid s_1)$$

$$P(s_1, s_2, s_3) = P(s_1)P(s_2 \mid s_1)P(s_3 \mid s_1, s_2)$$

details ***not*** examinable

$$\vdots$$

- to do it, we need to build a predictive model of the language - Machine Learning, 400 level.

# Summary

- key insight is to make a *stream code*
- with a fixed partitioning, based on fixed symbol probabilities from a look-up table, we get to the Shannon limit for "random looking" text
- with partitioning based on dynamic symbol probabilities (via a learned *predictive model*) we get close to the entropy of the *strings in the language*, ie. the theoretical limit ☺