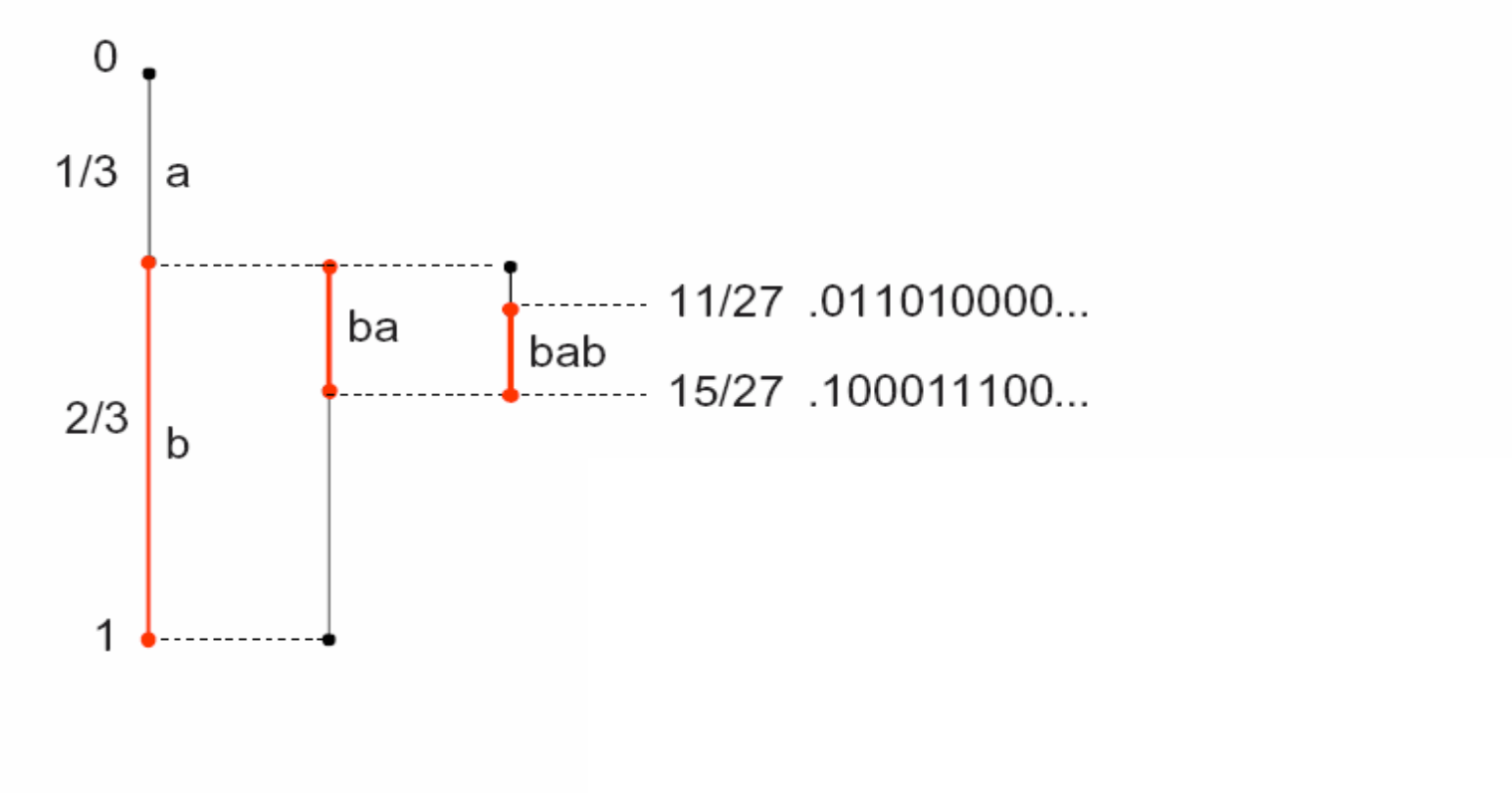
Lecture 9: Practical Arithmetic Coding

Thinh Nguyen
Oregon State University

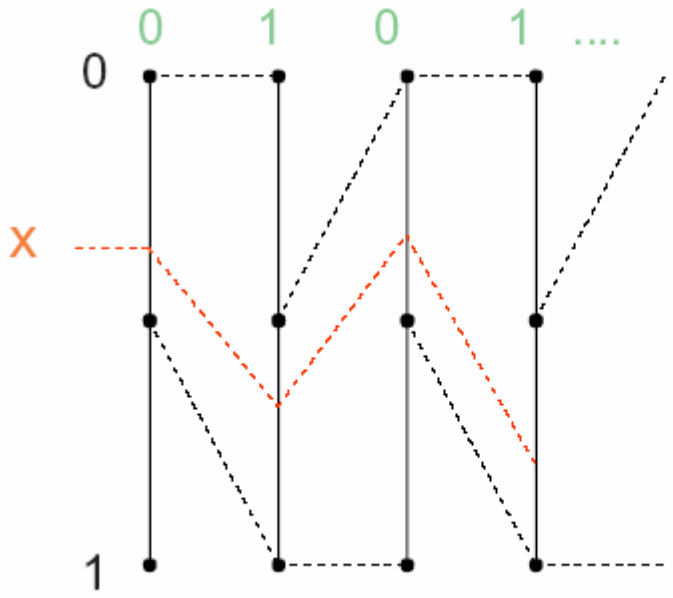
Issues with Arithmetic Coding

- The intervals are getting smaller as the sequence of symbols is getting longer.
- Arithmetics (computations) on very small numbers results in underflow!
- Need to rescale at every step!



Representation of Real Number in Binary

- Always scale the interval to unit size, but x must be changed as part of the scaling.



Binary Conversion with Scaling

```
y := x; i := 0
while y > 0 *
    i := i + 1;
    if y < 1/2 then b_i := 0; y := 2y;
    if y ≥ 1/2 then b_i := 1; y :=  2y – 1;
end{while}
b_j := 0 for all j ≥ i + 1
```

* Invariant: $x = .b_1 b_2 \ldots b_i + y/2^i$

Proof of Invariant

- Initially $x = 0 + y/2^0$
- Assume $x = .b_1 b_2 \ldots b_i + y/2^i$
  - Case 1. $y < 1/2$. $b_{i+1} = 0$ and $y' = 2y$

    $.b_1 b_2 \ldots b_i b_{i+1} + y'/2^{i+1} = .b_1 b_2 \ldots b_i 0 + 2y/2^{i+1}$

    $= .b_1 b_2 \ldots b_i + y/2^i$

    $= x$
  - Case 2. $y \geq 1/2$. $b_{i+1} = 1$ and $y' = 2y - 1$

    $.b_1 b_2 \ldots b_i b_{i+1} + y'/2^{i+1} = .b_1 b_2 \ldots b_i 1 + (2y-1)/2^{i+1}$

    $= .b_1 b_2 \ldots b_i + 1/2^{i+1} + 2y/2^{i+1} - 1/2^{i+1}$

    $= .b_1 b_2 \ldots b_i + y/2^i$

    $= x$

Exercise

x = 1/3

y	i	b
1/3	1	0
2/3	2	1
1/3	3	0
2/3	4	1
...	...	...

x = 17/27

y	i	b
17/27	1	1

Scaling

- Scaling:
  - By scaling we can keep L and R in a reasonable range of values so that W = R – L does not underflow.
  - The code can be produced progressively, not at the end.
  - Complicates decoding some.

Scaling Algorithm for Arithmetic Coding

Lower half

While [L,R) is contained in [0,.5) then

L := 2L; R := 2R

output 0, followed by C 1's

C := 0.

Upper half

While [L,R) is contained in [.5,1) then

L := 2L –1, R := 2R - 1

output 1, followed by C 0's

C := 0

Middle Half

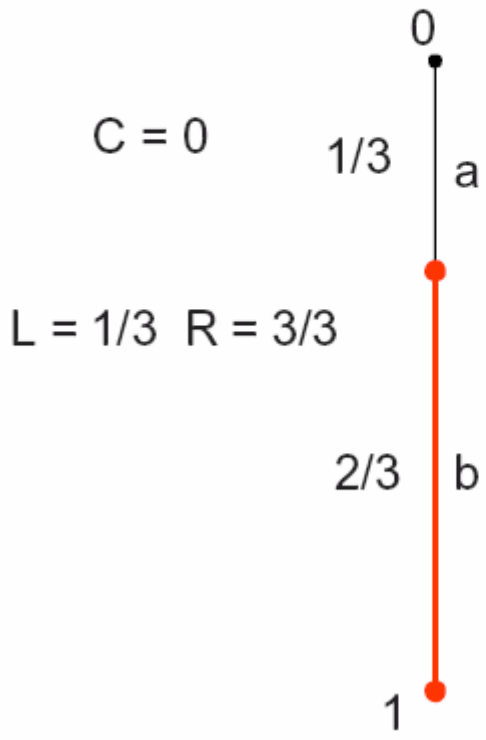
While [L,R) is contained in [.25,.75) then

L := 2L –.5, R := 2R -.5

C := C + 1.

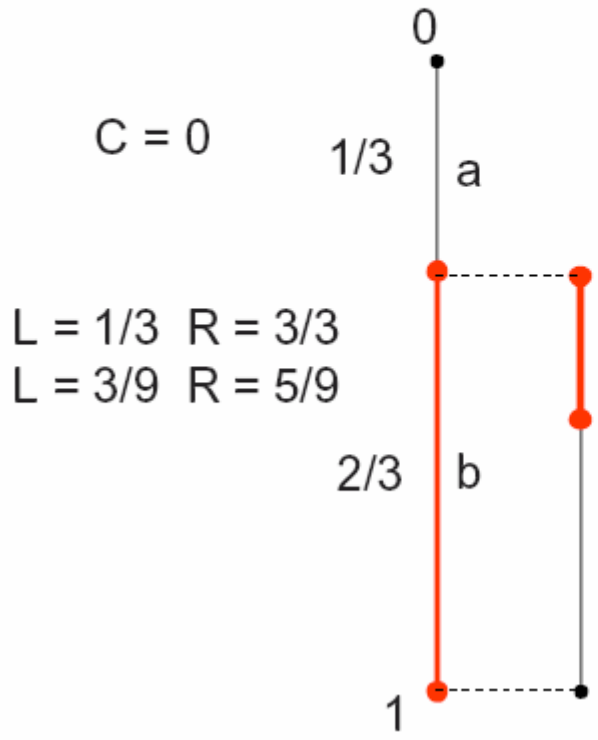
Example

- baa



Example

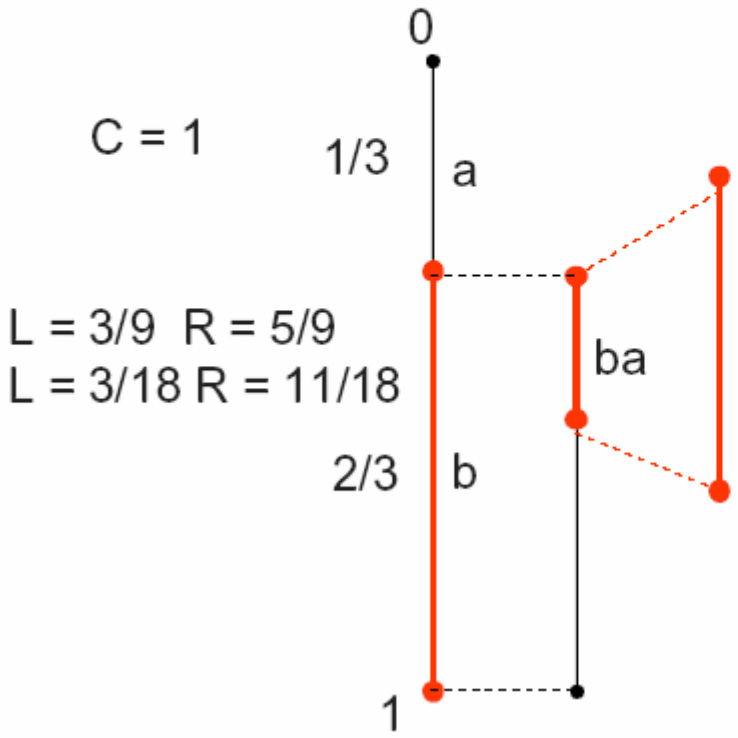
- baa



Scale middle half

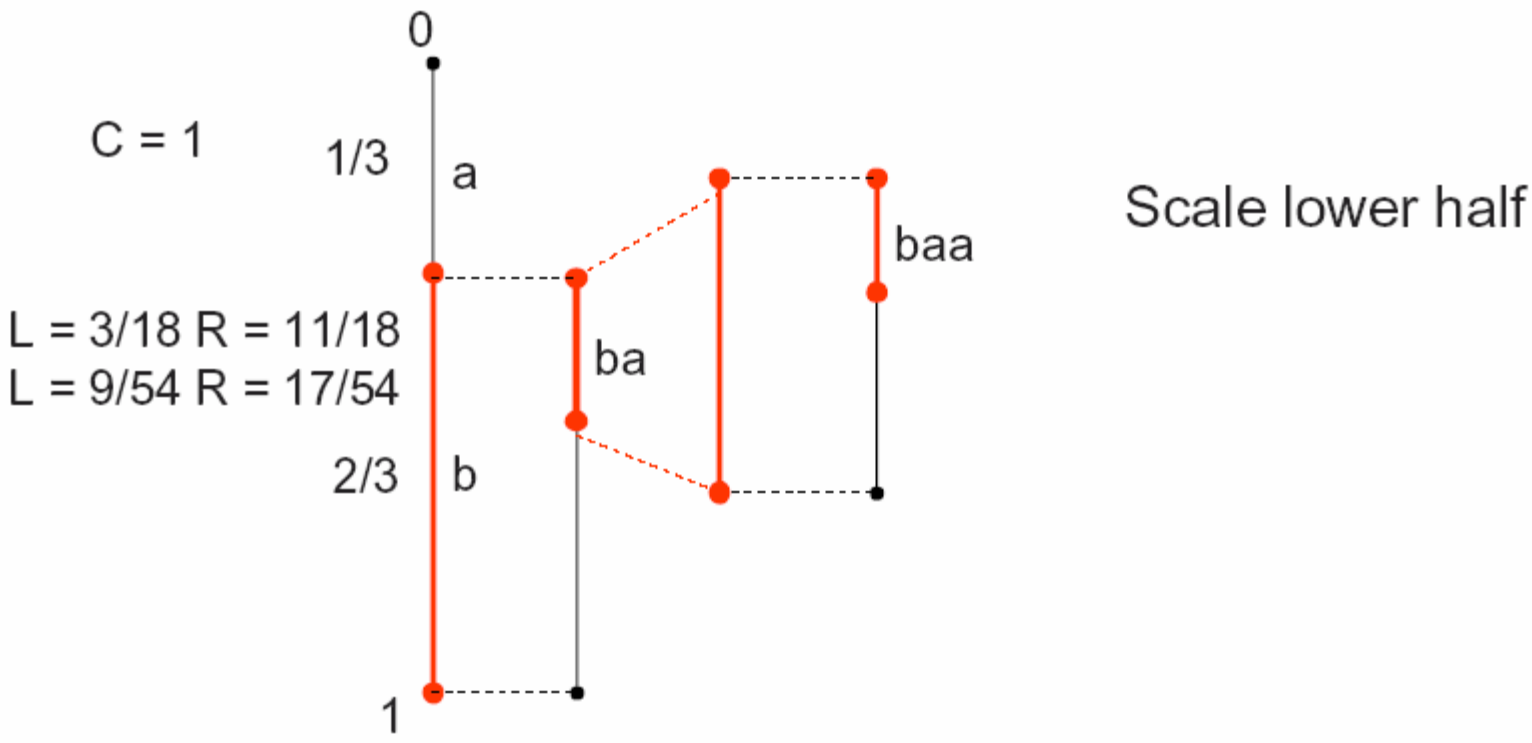
Example

- baa



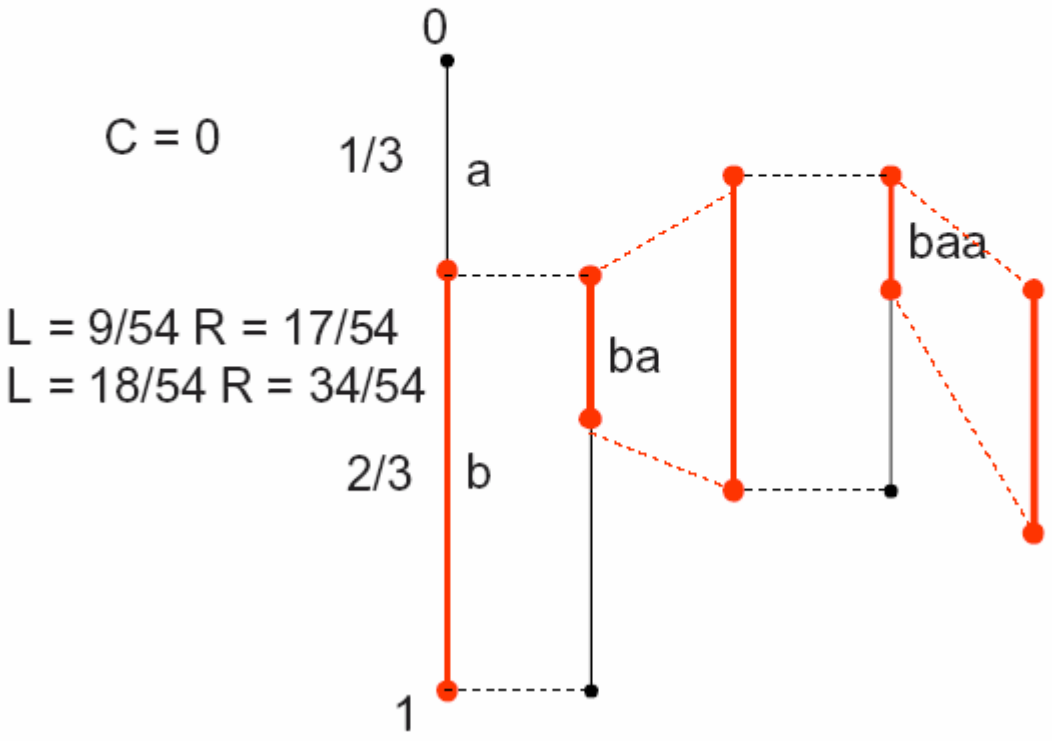
Example

- baa



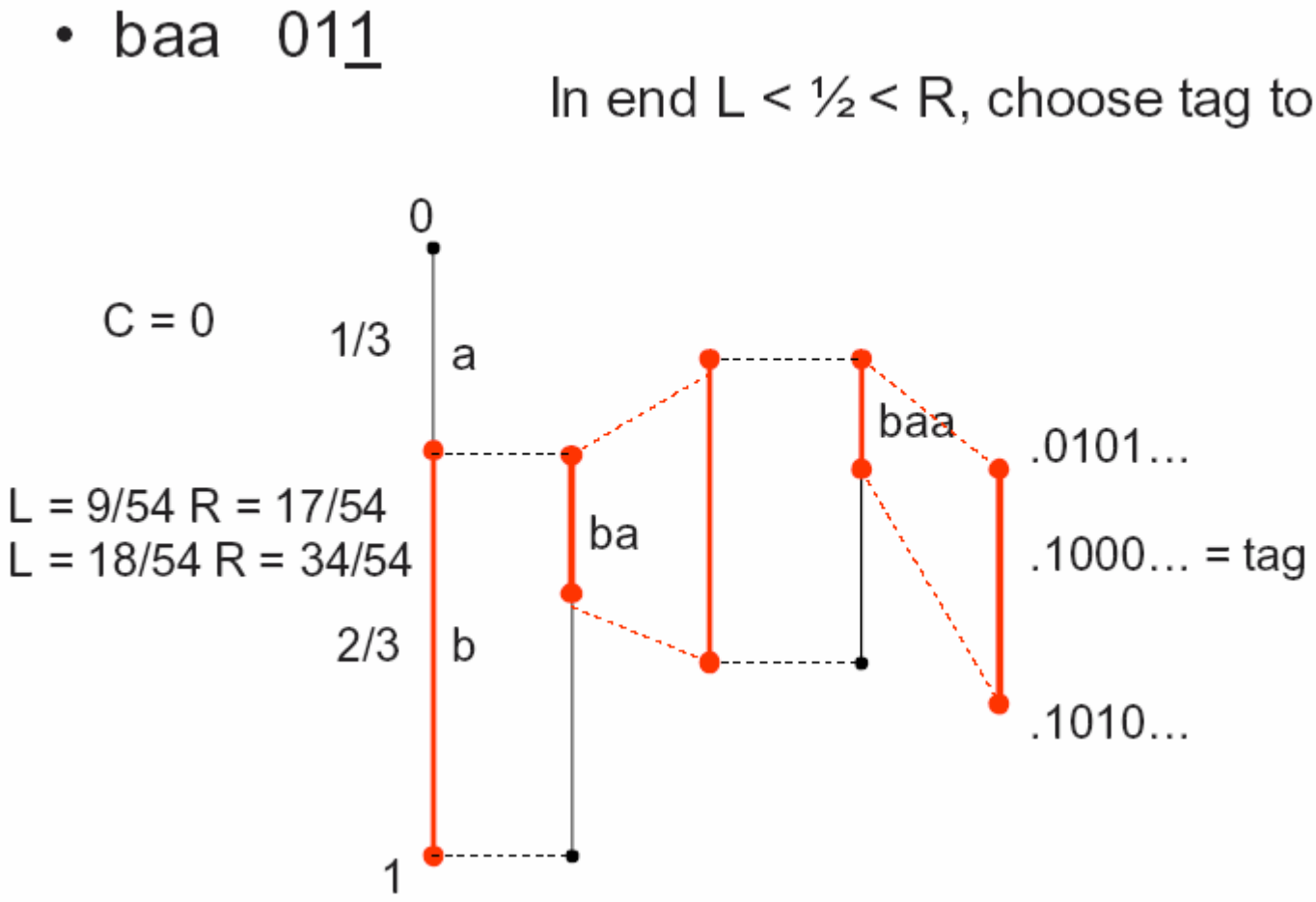
Example

- baa 01



Example

- baa 011



Integer Implementation

- m bit integers
  - Represent 0 with 000…0 (m times)
  - Represent 1 with 111…1 (m times)
- Probabilities represented by frequencies
  - n_i is the number of times that symbol a_i occurs
  - $C_i = n_1 + n_2 + \ldots + n_{i-1}$
  - $N = n_1 + n_2 + \ldots + n_m$

$$W := R - L + 1$$

$$L' := L + \left\lfloor \frac{W \cdot C_i}{N} \right\rfloor$$

$$R := L + \left\lfloor \frac{W \cdot C_{i+1}}{N} \right\rfloor - 1$$

$$L := L'$$

Coding the i-th symbol using integer calculations.
Must use scaling!

Arithmetic Coding with Context

- Maintain the probabilities for each context.
- For the first symbol use the equal probability model.
- For each successive symbol use the model for the previous symbol.

Arithmetic Coding with Context

- Simple solution – Equally Probable Model.
  - Initially all symbols have frequency 1.
  - After symbol x is coded, increment its frequency by 1.
  - Use the new model for coding the next symbol.
- Example in alphabet a,b,c,d.

		a	a	b	a	a	c
a	1	2	3	3	4	5	5
b	1	1	1	2	2	2	2
c	1	1	1	1	1	1	2
d	1	1	1	1	1	1	1

After aabaac is encoded
The probability model is

a 5/10	b 2/10
c 2/10	d 1/10

Arithmetic Coding with Context

- Both compress very well. For m symbol grouping.
  - Huffman is within 1/m of entropy.
  - Arithmetic is within 2/m of entropy.
- Context
  - Huffman needs a tree for every context.
  - Arithmetic needs a small table of frequencies for every context.
- Adaptation
  - Huffman has an elaborate adaptive algorithm
  - Arithmetic has a simple adaptive mechanism.
- Bottom Line – Arithmetic is more flexible than Huffman.