Daala Entropy Coder: AOM Hardware Subgroup

Timothy B. Terriberry
June 27, 2016

Entropy Coder Use in AOM: High-Level Overview

Current EC Use (1)

- Probabilities constant for each frame
  - Initialized to defaults
  - (Optional) Feed-forward updates from prior frames
  - Explicitly coded updates to current frame
- Encoder
  - Initial RDO search for all tiles using defaults/feed-forward probabilities, buffer all coding decisions
  - Collect statistics from all tiles, then decide which updates to code (RDO to see if they save enough)
  - Write out symbols from buffered decisions

Current EC Use (2)

- Benefits: very simple decoder
- Drawbacks:
  - Suboptimal for interactive video
    - Must disable feed-forward updates
    - Must reset/update all probabilities every frame
  - Encoder must buffer decisions for whole frame
  - Can’t generate bitstream for any tile until all tiles have finished encoding (higher latency)
  - Can’t model non-stationary contexts
  - RDO done with wrong probabilities

The Case for Adaptive Probabilities

- Drawbacks: more work in the decoder
  - Most already being done for feed-forward updates
- Benefits
  - Don’t need to disable anything for interactive
  - Need to buffer at most one MTU (~1200 bytes)
  - Can output bitstreams for all tiles while encoding (one-pass, parallel operation)
  - Can model non-stationary contexts
  - Can do RDO with real probabilities

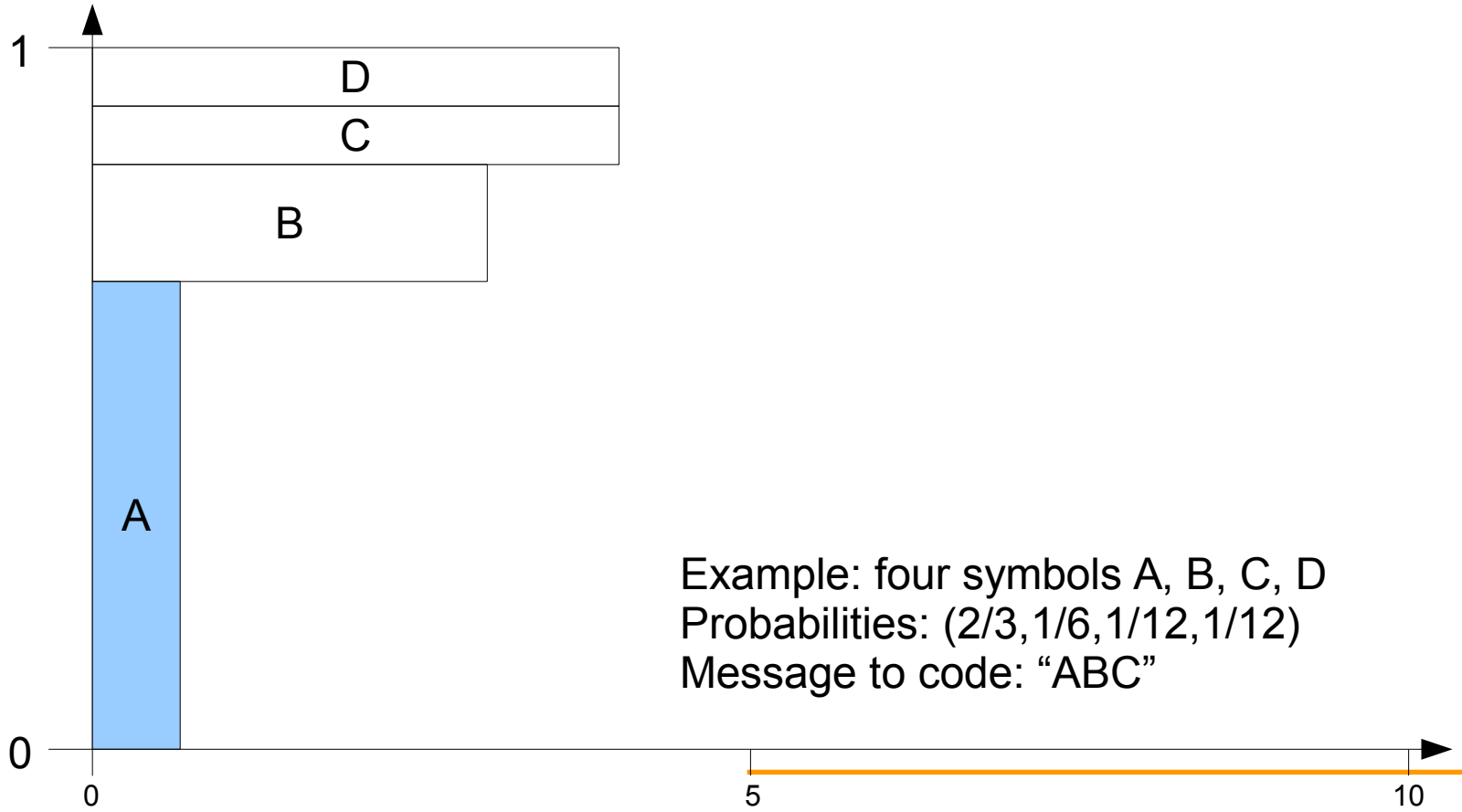
The Case for Non-Binary

- Performance
  - Fewer symbols means higher throughput
- IPR (details omitted)

Arithmetic Coding: Low-Level Overview

Arithmetic Coding: Intro (1)

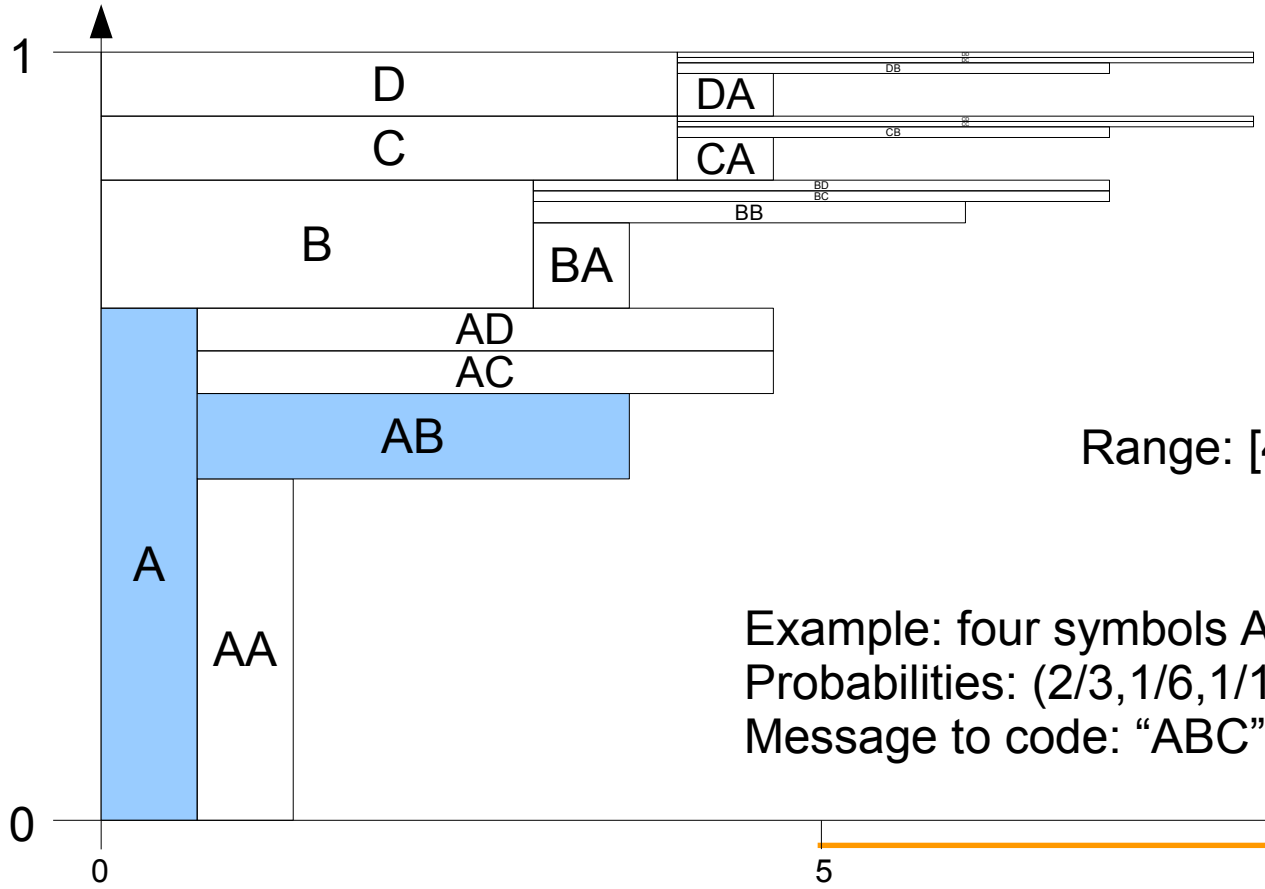
- Basic idea: split an interval proportional to the probability of each symbol



Example: four symbols A, B, C, D
Probabilities: (2/3,1/6,1/12,1/12)
Message to code: “ABC”

Arithmetic Coding: Intro (2)

- Choose a sub-interval, recurse

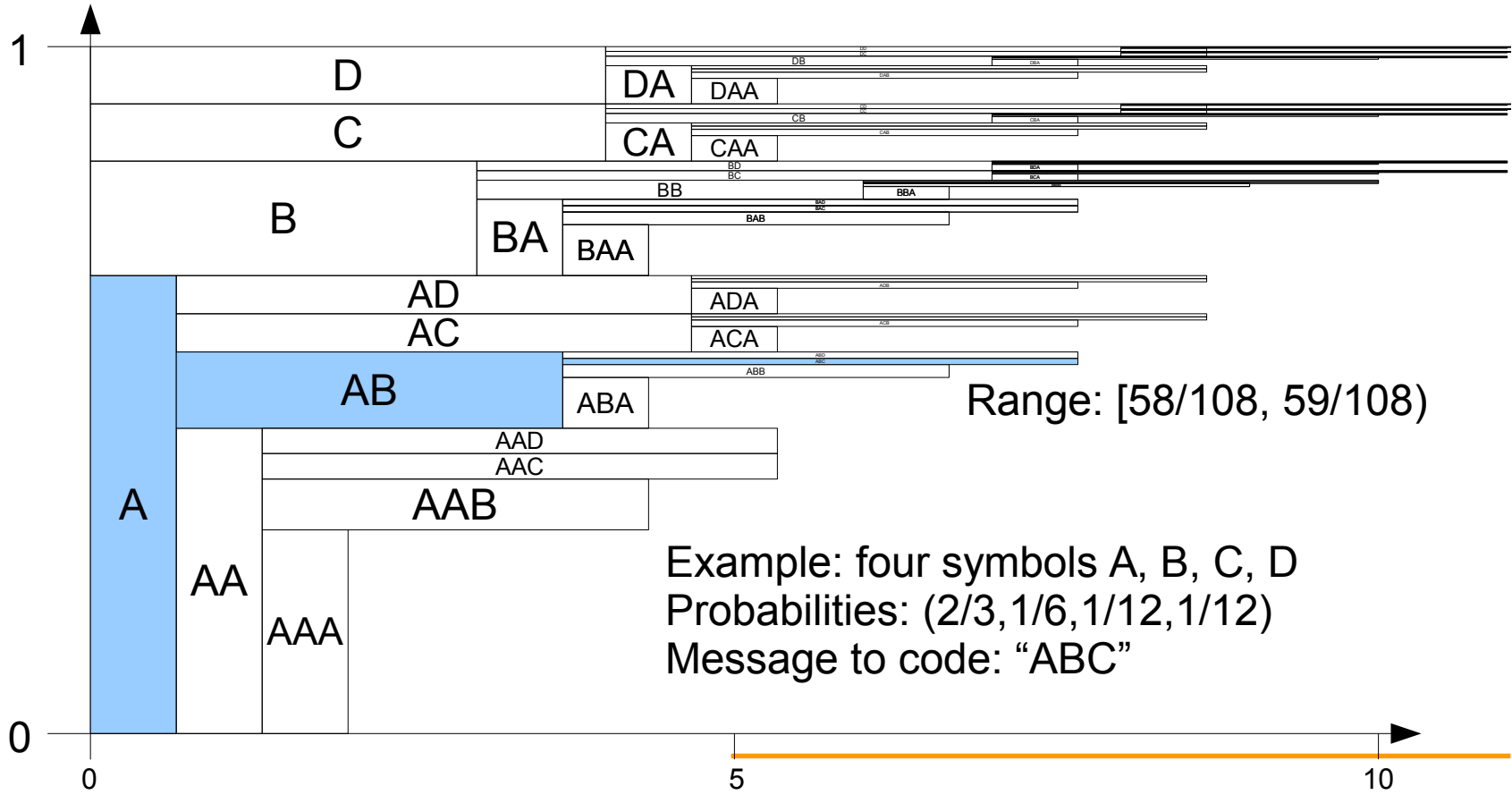


Range: [4/9, 5/9)

Example: four symbols A, B, C, D
Probabilities: (2/3,1/6,1/12,1/12)
Message to code: “ABC”

Arithmetic Coding: Intro (3)

- Encoded message: any number in final interval
- Number of bits ≤ $\lceil -\log_2(1/108) \rceil = \lceil 6.75 \rceil = 7$



Mozilla

Practicalities

- State required
  - *L*: lower bound of interval
  - *R*: size of interval
- Finite precision
  - Size of *R* fixed (e.g., 16 bits)
  - Rescale *L* to keep *R* between ½ and 1
  - Flush high-order bits of *L* to an external buffer

Main State Update

- p: probability of actual symbol
- f: cumulative probability of lower symbols
- $u(f,R)$: Partition function
  - Maps cumulative probability to the range $[0,R]$ (details in a few slides)
- $L \rightarrow L + u(f,R)$
- $R \rightarrow u(f + p,R) - u(f,R)$

Boring Stuff

- Size of *R*
  - 8 bits in VPx,16 bits in Daala
- When to rescale?
  - After every symbol
- Carry propagation
  - If range straddles a bit boundary (*L* + *R* overflows a register), high order bits of *L* not yet known
  - Various well-known strategies to cope

Interesting Stuff

- How do we partition the interval *R*?
- Where do we get the symbol probabilities?

Partition Functions

VPx Partition Function

$$u(R, f) = 1 + \lfloor (R-1) * f/256 \rfloor$$

- *f*: 8-bit “probability”
  - Note: *f* = 0 and *f* = 1 are the same!
  - Actual probability slightly off from *f*/256.0
- Requires one 8x8 → 16 bit multiply
- Within 0.1% of theoretically optimal
- Math generalizes easily to non-binary coding
  - 8 bits is not enough for non-binary probabilities

Daala Partition Function: Dyadic Rational Probabilities

$$u(R, f) = \lfloor f * R / 32768 \rfloor$$

- *f*: 15-bit cumulative probability
  - Actual probability is *f*/32768.0
- Requires one 15x16 → 31 bit multiply
  - Can replace *R* with ⌊*R*/256⌋ for 15x8 → 23 without much accuracy loss (no worse than VPx)
    - One extra operation in software, but happy to do it if it helps hardware

Daala Partition Function: Non-Dyadic Probabilities

$$u(R, f) = f + \min(f, R - T)$$

- *f*: 15-bit cumulative probability
- *T*: total probability
  - Actual probability is *f*/T
- Requires $T \le R < 2*T$ (rescale if not)
- Requires no multiplies, only 15-bit add/min
- Introduces small (1...2%) approximation error
  - Comparable to CABAC

Non-Binary Coding

- Encoding almost the same
  - Requires two evaluations of $u(f,R)$ instead of one
- Decoding requires search
  - Evaluate $u(f,R)$ for all symbols (up to 16)
  - Find which sub-interval contains coded value
  - Alternative: split alphabet recursively
    - Implementation detail how much to do in parallel

Raw Bits

- VPx has no explicit “bypass mode” (just $f = 128$)
- Can easily do the same for Daala (Dyadic case, $f = 16384$)
- Instead, currently write raw bits at the end of a packet (working backwards towards start)
  - Allows software to code arbitrary # of bits at once
  - Implementation details:
    - Encoder uses two buffers
    - Combined in a final pass that also does all carry propagation

Probability Modeling

Current AOM Modeling

- When using Daala EC in AOM, binary trees converted to static Cumulative Distribution Functions (CDFs)
  - Max CDF size of 16 (most trees can be converted to one CDF)
  - Total probability T = 32768 (uses dyadic case)
- All other probability modeling unchanged
  - RDO still counts binary statistics
  - Feed-forward updates in decoder do, too
  - Frame header still codes binary probability updates

Current Daala Modeling

- Most based on simple frequency counts
  - Initialize flat, e.g., *F*={32,64,96,128,...}
  - Add a constant for each coded symbol, e.g., to code a 1, update *F'*={32,192,224,256,...}
    - One 15-bit vector add
  - When counts get too large, rescale
    - One 15-bit vector shift/add
- Requires non-dyadic partition function

Non-Binary Example: PVQ Coefficient Coding (1)

- Basic idea: sum of absolute values, *K*, known (true for PVQ, must be encoded for scalar)
- Split coefficient vector in half
- Code how much of *K* goes in the second half
  - If *K* is larger than 7, code top 3 bits with arithmetic coder, code rest with raw bits
  - Context chosen from the vector dimension (14 possibilities, from 2...128) and top bits of *K* (1...7)
  - One CDF of size 2...8 per context
    - 490 15-bit words

Non-Binary Example: PVQ Coefficient Coding (2)

- Special case: $K = 1$ and vector dimension ≤ 16
  - Code exact location of the 1 with one symbol
  - 12 contexts based on the vector dimension:
    - 4 for vectors that start out with dimension ≤ 16
    - 8 for vectors that get split down to dimension ≤ 16
  - One CDF of size 2...16 per context
    - 113 15-bit words
- Sign bits coded with raw bits
- Worst case: (N – 1) entropy-coded symbols plus $2N$ – 1 calls to read/write raw bits

Dyadic Probability Modeling (Experimental)

- Fix total, T, at 32768
- Probability updates maintain this total
  - Symbol $i <$ coded value
    - $f_i \rightarrow f_i - \lfloor (f_i + 2^{rate} - i - 2)/2^{rate} \rfloor$
  - Symbol $i \geq$ coded value
    - $f_i \rightarrow f_i - \lfloor (f_i + M - i - 32769)/2^{rate} \rfloor$
    - M = alphabet size
- Additional rules for first few symbols in a given context to speed up adaptation

Questions?