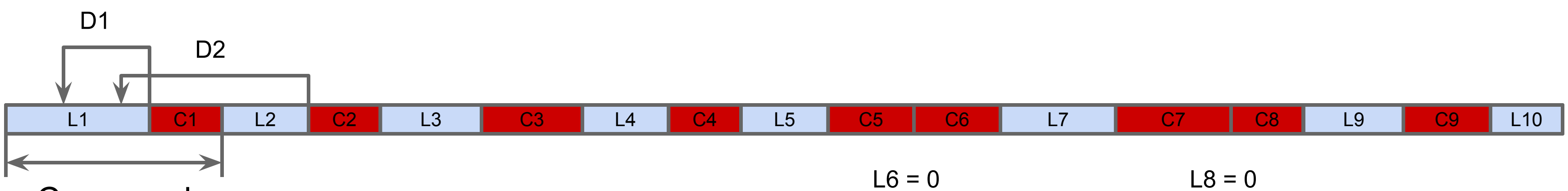
Brotli Compression Algorithm

outline of a specification

Overview

- Structure of backward reference commands
- Encoding of commands
- Encoding of distances
- Encoding of Huffman codes
- Block splitting
- Context modeling
- Meta-block structure

LZ77 decomposition



Command:
L bytes inserted (literals)
C bytes copied from D bytes distance (C>= 2, D >= 1)

Ring buffer of last 4 distances

d_4	d_3	d_2	d_1

Distance short codes 1 - 16 for

$d_1, d_2, d_3, d_4, d_1 \pm \{1, 2, 3\}, d_2 \pm \{1, 2, 3\}$

other distances are shifted by 16

Example:

distance sequence	1,	10,	10,	1,	3,	4,	7,	5
transformed to	17,	26,	1,	2,	8,	6,	10,	7

Encoding of commands

Literal insertion length alphabet:

0	1	2	3	4	5	6 - 7	8 - 9	short
10 - 13	14 - 17	18 - 25	26 - 33	34 - 49	50 - 65	66 - 97	98 - 129	medium
130 - 193	194 - 321	322 - 577	578 - 1089	1090 - 2113	2114 - 6209	6210 - 22593	22594 - 16799809	long

Copy length alphabet:

2	3	4	5	6	7	8	9	short
10 - 11	12 - 13	14 - 17	18 - 21	22 - 29	30 - 37	38 - 53	54 - 69	medium
70 - 101	102 - 133	134 - 197	198 - 325	326 - 581	582 - 1093	1094 - 2117	2118 - 16779333	long

Joint literal- and copy length histogram

	short copies	medium copies	long copies
short literal seq. last dist.	0 - 63	64 - 127	
short literal seq.	128 - 191	192 - 255	384 - 447
medium literal seq.	256 - 319	320 - 383	512 - 575
long literal seq.	448 - 511	576 - 639	640 - 703

Examples:

Command (11, 19, 100)
- medium literal seq. (bucket 0)
- medium copy (bucket 3)

→ code 323 + 4 extra bits

Command (0, 2, 1)
- short literal seq. last distance (bucket 0)
- short copy (bucket 0)

→ code 0, no extra bits

Encoding of distances

Distance alphabet consists of:

- short codes for previous distances (1 - 16)
- *M* number of “direct codes” encoded without extra bits
- buckets of length 2^n encoded with n extra bits
- optionally buckets can have k last bits fixed
- 2^{k+1} bucket for each number of extra bits

Example for M = 12 and k = 1

short codes (1 - 16)	direct codes (17 - 28)	29, 31	30, 32	33, 35	36, 38
39, 41, 43, 45	40, 42, 44, 46	47, 49, 51, 53	48, 50, 52, 54	...	

Encoding of Huffman codes

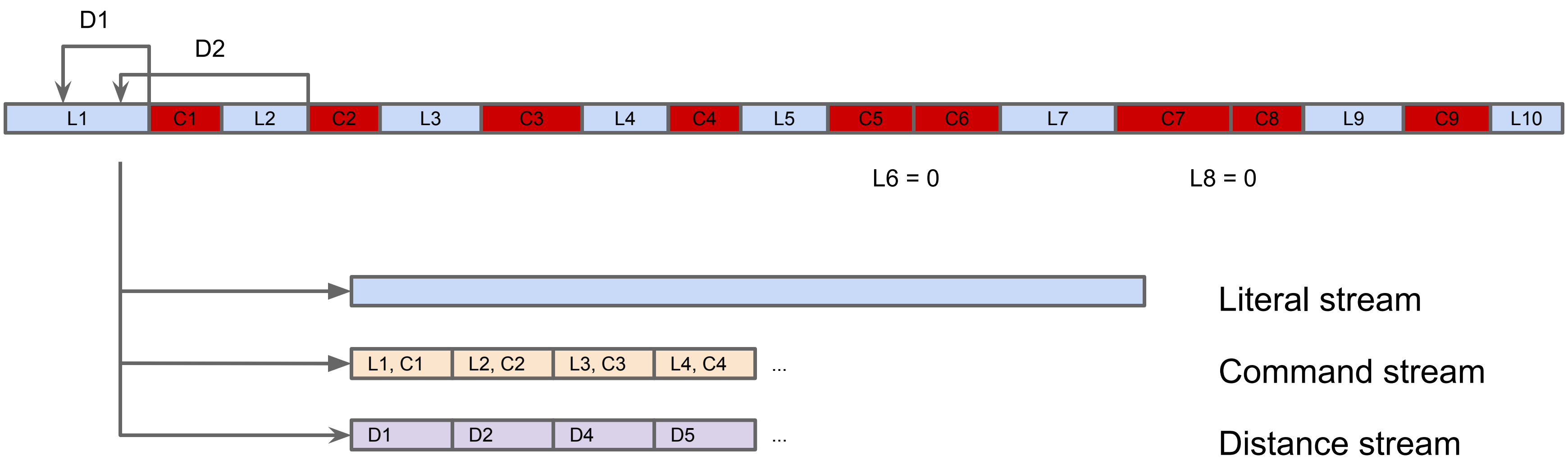
Similar to DEFLATE, with the following changes:

- Order of code length codes:

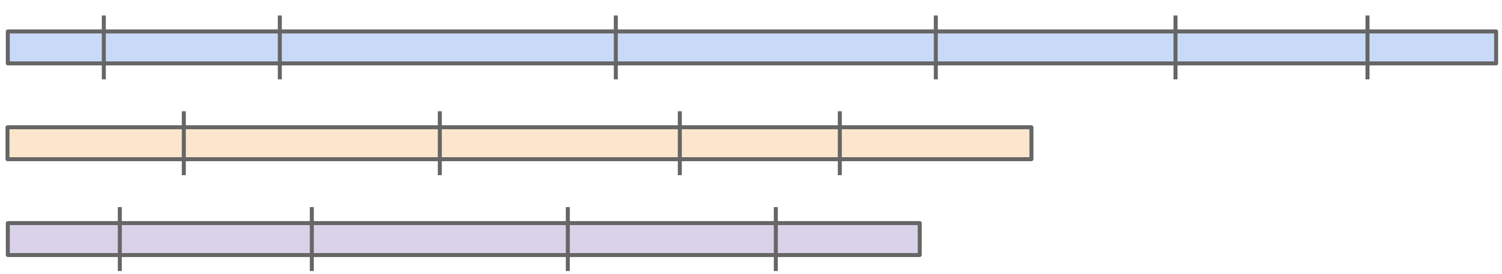
  17, 18, 0, 1, 2, 3, 4, 5, 16, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

- Special encoding of Huffman codes with <= 2 non-zero lengths
  - 1 bit for small tree marker
  - 1 bit for the number of symbols (1 or 2)
  - 1 bit to indicate if the first symbol is 0 or 1
  - 1 or 8 bits for the first symbol
  - 8 bits for the second symbol, if present
- No bits emitted if Huffman code has only one symbol

Block splitting



Independent block boundaries for literals, commands and distances:



Encoding of block switch symbols

Type1, Len1	Type2, Len2	Type3, Len3	Type4, Len4

Block length alphabet:

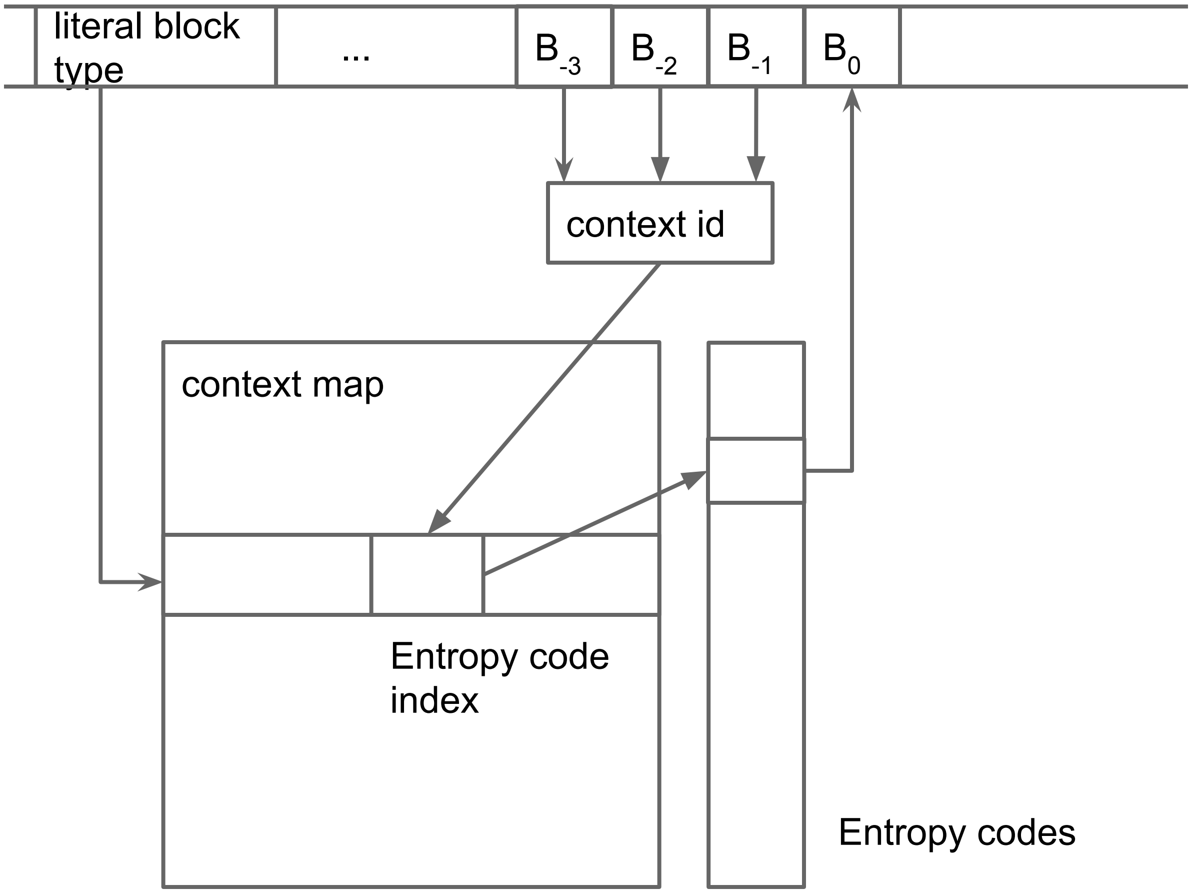
1 - 4	5 - 8	9 - 12	13 - 16	17 - 24	25 - 32	33 - 40	41 - 48
49 - 64	65 - 80	81 - 96	97 - 112	113 - 144	145 - 176	177 - 208	209 - 240
241 - 304	305 - 368	369 - 496	497 - 752	753 - 1264	1265 - 2288	2289 - 4336	4337 - 8432
8433 - 16624	16624 - 16793839						

- Max 254 block types
- Special block type codes for
  - second last block type
  - last block type + 1
- Block switch is encoded with
  - block type code
  - block length code + extra bits

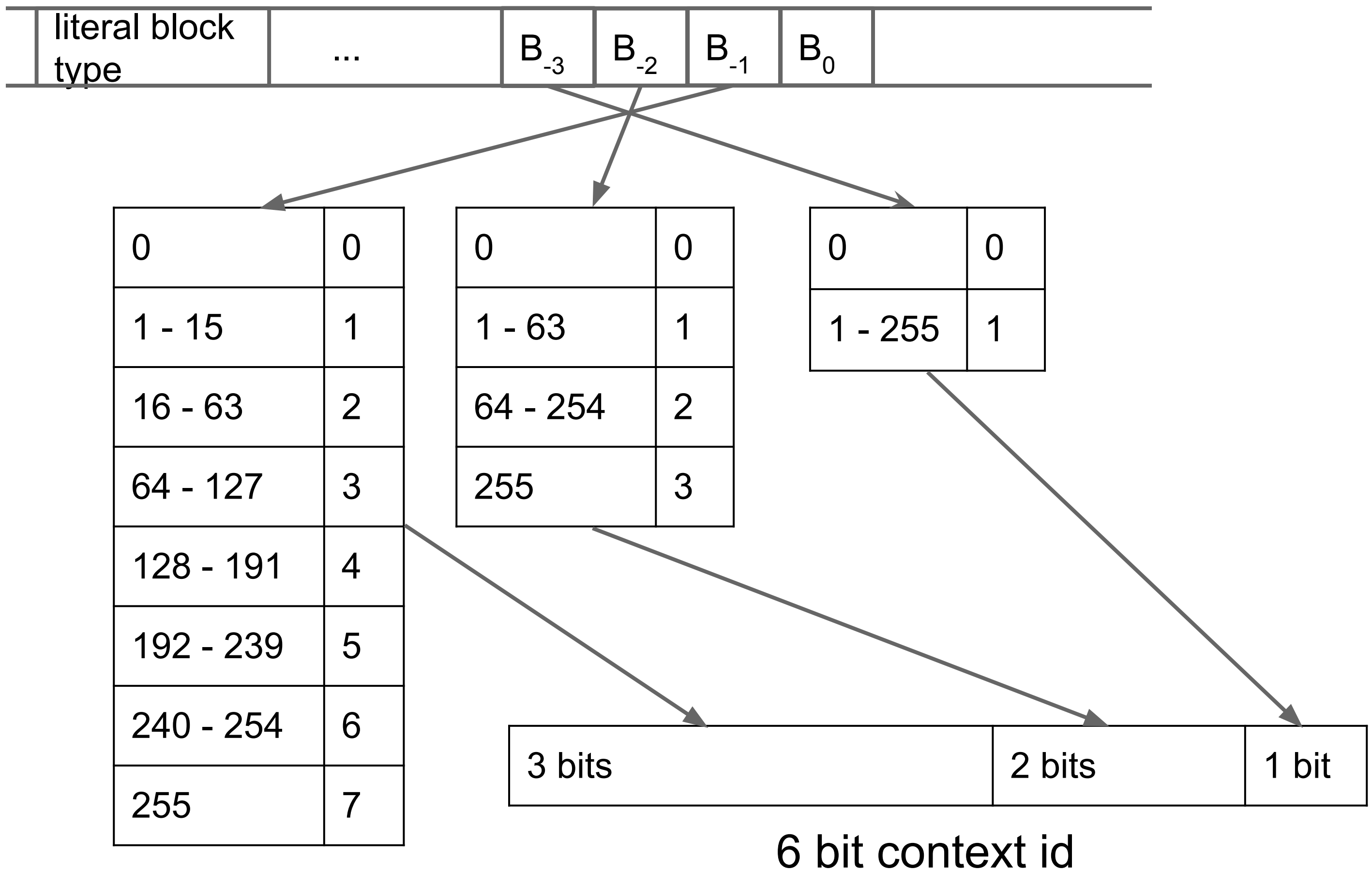
Encoding of the block split

- 1 bit to indicate if we have more than one block type
- 8 bits for number of block types - 1
- Huffman code on block type alphabet
- Huffman code on block length alphabet
- Length of first block
  - codeword based on Huffman code
  - extra bits

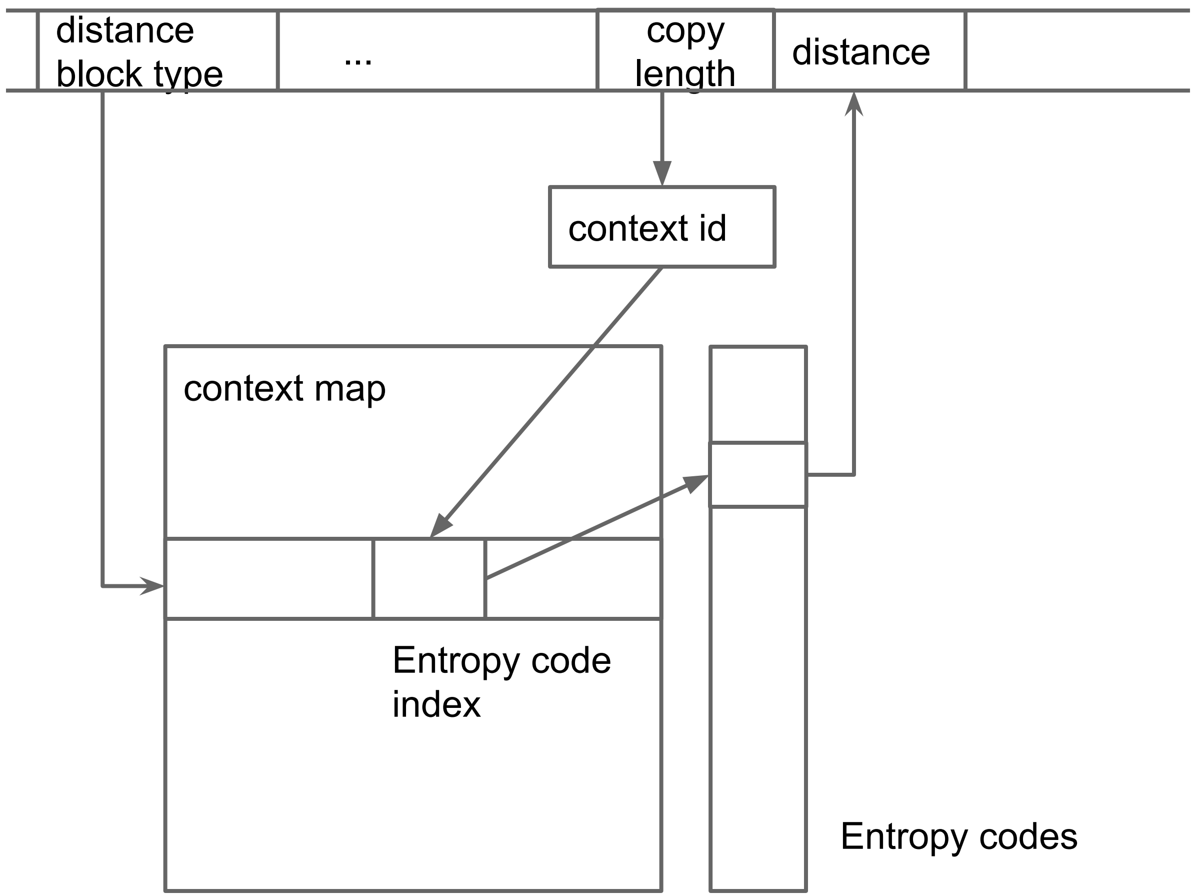
Context modeling for literals



Context modeling for literals



Context modeling for distances



Encoding of the context map

- 1 bit to enable/disable context modeling
- 4 bits for the context mode id (only in the literal context map)
- 8 bits for the number of entropy codes
- 1 bit to enable/disable run-length coding of zero sequences
- 4 bits for max zero-run-length bucket
- 1 bit to enable/disable Move-to-Front transformation
- Huffman code on entropy code ids + run length codes
- row-by-row encoding of the entropy code ids and run length codes using the above Huffman code

Format specification

```
<compressed file>:
  <uncompressed size>
  (<meta-block size><meta-block>)*

<uncompressed size>:
```
number_of_bytes 3 bits
size number_of_bytes bytes

```
<meta-block size>:
```
is_last 1 bit
size log2(uncompressed size) * is_last bits

Example:
empty file is encoded with 3 zero bits

Meta-block format

```
<meta-block>:
  <literal block split>
  <command block split>
  <distance block split>
  <distance postfix bits and num direct codes>
  <literal context map>
  <distance context map>
  <literal Huffman code>*
  <command Huffman code>+
  <distance Huffman code>*
  [<command block switch>?<command>
   (<literal block switch>?<literal>)*
   <distance block switch>?<distance>?]+
```