Toward A Unified Abstract Content API

Composing content at a higher level

Romain Beauxis, FOSDEM 2025

Context: Liquidsoap

- Created ca. 2004 by Samuel Mimram and David Baelde while at ENS
- Scripting language for media streaming
- Functional, statically typed with inferred types
- Expanded to a fully featured language with tight integration with FFmpeg
- Support for audio and video muxing and streaming

Context: Liquidsoap

```
# The main playlist
s = playlist("/path/to/playlist")

# Add a fallback
security = single("/path/to/message.mp3")

s = fallback([s, security])

let {audio = a, metadata = m, track_marks = tm} = source.tracks(s)

# Some background
v = single("/path/to/video.mp4")

let {video = v} = source.tracks(v)

# Mux video background with playlist.
radio = source({audio=a, video=v, metadata=m, track_marks=tm})
```

Context: Liquidsoap

```
# The main playlist
s = playlist(id="audio", "~/sources/test-stream/jeff")

# Add a fallback
security = single("~/sources/test-stream/jeff/whatever.mp3")

s = fallback(track_sensitive=false, [s, security])

let {audio = a, metadata = m, track_marks = tm} = source.tracks(s)

# Some background
v = playlist(id="video", "~/sources/test-stream/fosdem/background")

security = single("~/sources/test-stream/fosdem/background/1.mp4")

v = fallback(track_sensitive=false, [v, security])

let {video = v} = source.tracks(v)

# Mux video background with playlist.
radio = source({audio=a, video=v, metadata=m, track_marks=tm})
```

```
# Add radio name
radio =
  video.add_text(
    color=0x000000,
    x=100 @ px,
    y=100 @ px,
    size=200 @ rem,
    "My Radio",
    radio
  )

# Output to HLS
enc =
  %ffmpeg(
    format = "mpegts",
    %audio(codec = "aac", b = "192k"),
    %video(codec = "libx264", preset = "ultrafast", g = 50)
  )

streams = [("radio", enc)]

output.file.hls("/path/to/hls", streams, radio)
```

Context: Liquidsoap

```
# The main playlist
s = playlist(id="audio", "~/sources/test-stream/jeff")

# Add a fallback
security = single("~/sources/test-stream/jeff/whatever.mp3")

s = fallback(track_sensitive=false, [s, security])

let {audio = a, metadata = m, track_marks = tm} = source.tracks(s)

# Some background
v = playlist(id="video", "~/sources/test-stream/fosdem/background")

security = single("~/sources/test-stream/fosdem/background/1.mp4")

v = fallback(track_sensitive=false, [v, security])

let {video = v} = source.tracks(v)

# Mux video background with playlist.
radio = source({audio=a, video=v, metadata=m, track_marks=tm})
```

```
# Output to HLS
enc =
  %ffmpeg(
    format = "mpegts",
    %audio.copy,
    %video.copy
  )

streams = [("radio", enc)]

output.file.hls("/path/to/hls", streams, radio)
```

Context: Liquidsoap

```
# The main playlist
s = playlist(id="audio", "~/sources/test-stream/jeff")

# Add a fallback
security = single("~/sources/test-stream/jeff/whatever.mp3")

s = fallback(track_sensitive=false, [s, security])

let {audio = a, metadata = m, track_marks = tm} = source.tracks(s)

# Some background
v = playlist(id="video", "~/sources/test-stream/fosdem/background")

security = single("~/sources/test-stream/fosdem/background/1.mp4")

v = fallback(track_sensitive=false, [v, security])

let {video = v} = source.tracks(v)

# Mux video background with playlist.
radio = source({audio=a, video=v, metadata=m, track_marks=tm})
```

```
# Output to HLS
enc =
  %ffmpeg(
    format = "mpegts",
    %audio.copy,
    %video.copy
  )

streams = [("radio", enc)]

output.file.hls("/path/to/hls", streams, radio)
```

Context: Liquidsoap

- Scheduled switch for hourly playlists
- Switch to connected live source client (DJs)
- Multiple outputs: Icecast, HLS (also multiple encodings/encapsulations), RTMP, etc..
- Dynamic encoded content handling
- Related to other needs e.g. encoder restart in FFmpeg

Context: Liquidsoap

Demo! 

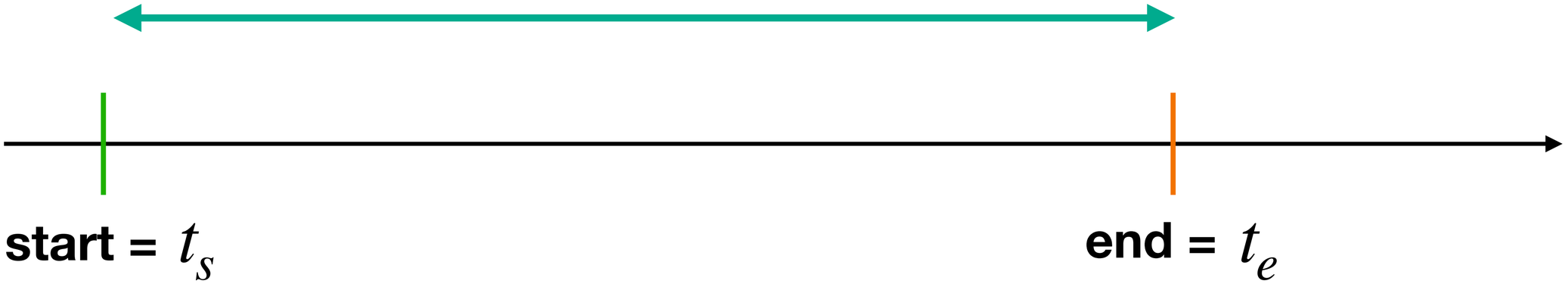
Context: Liquidsoap

Streaming 101

frame duration = 0.02s

⏰: CPU

Generate 0.02s of content, send it to all outputs

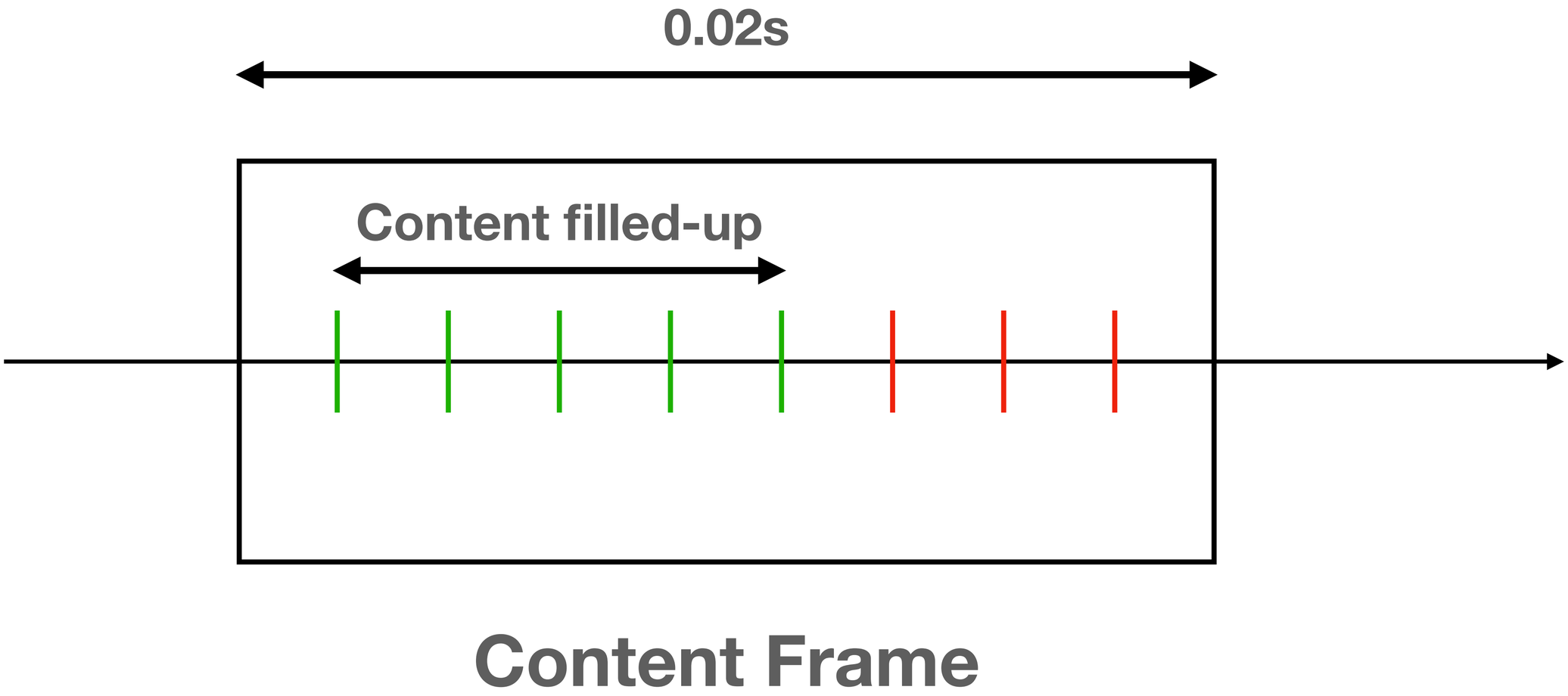


$t_e - t_s < 0.02s$: **sleep** 😴

$t_e - t_s \geq 0.02s$: **we're late!** 😰

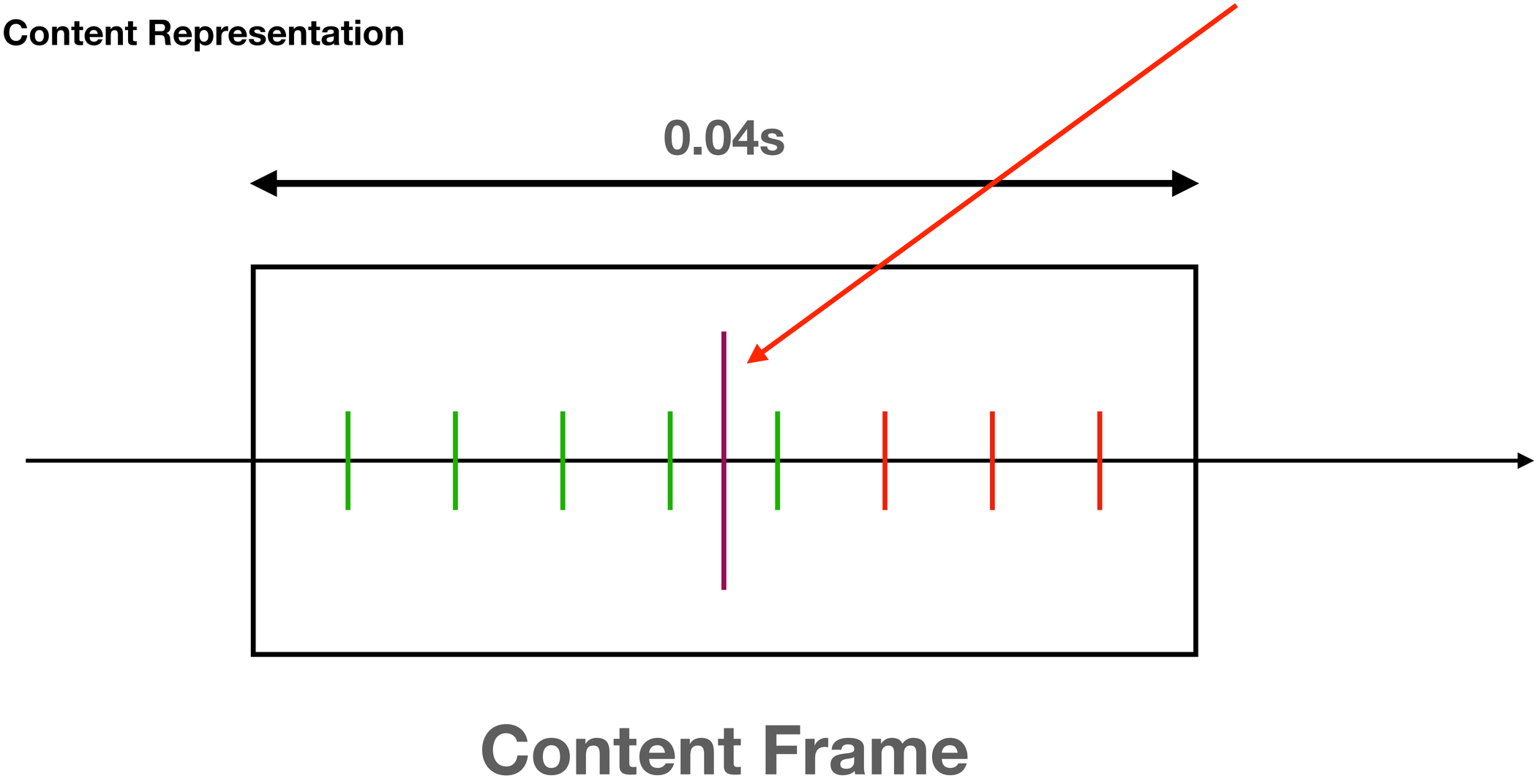
Context: Liquidsoap

Old Content Representation



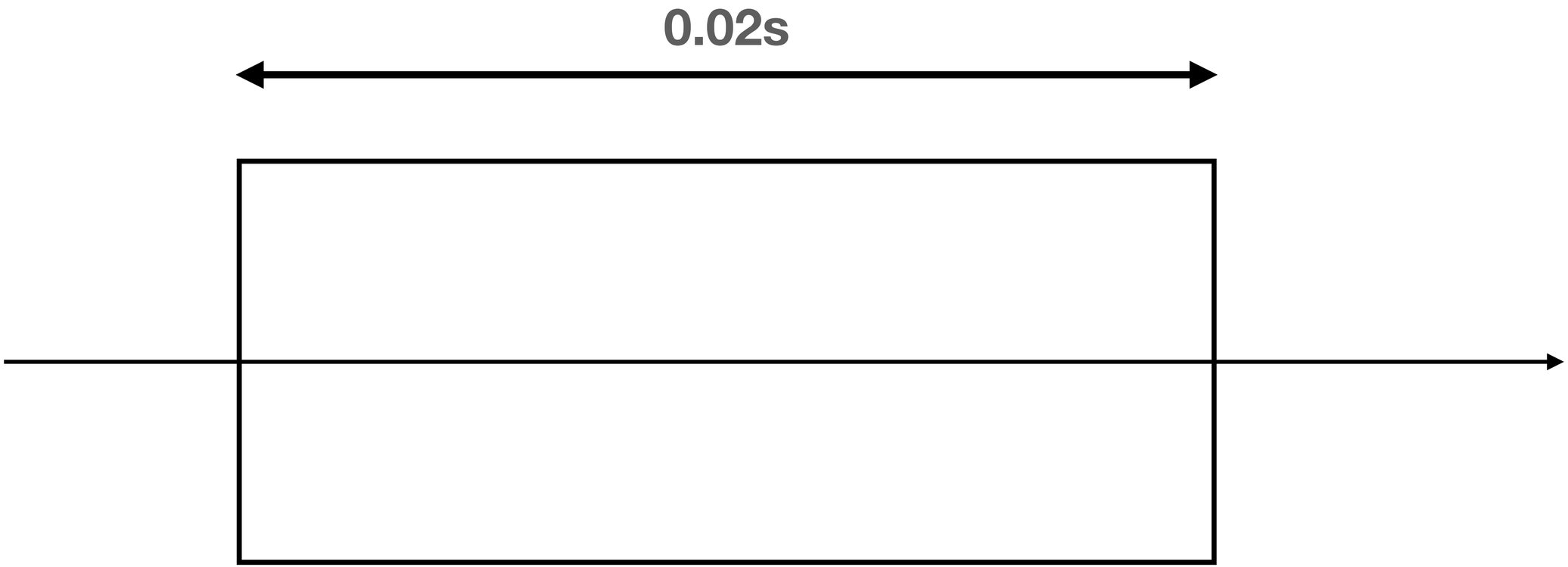
Context: Liquidsoap

Old Content Representation



Context: Liquidsoap

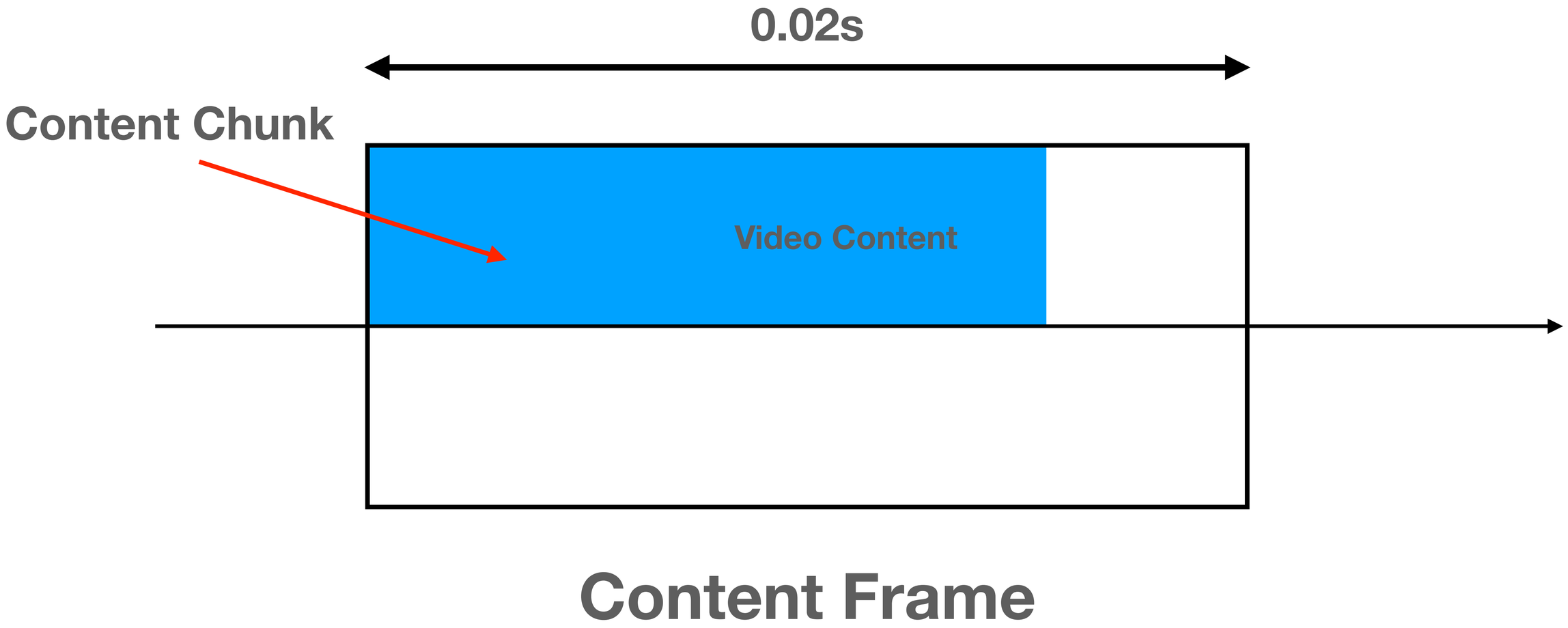
New Content Representation



Content Frame

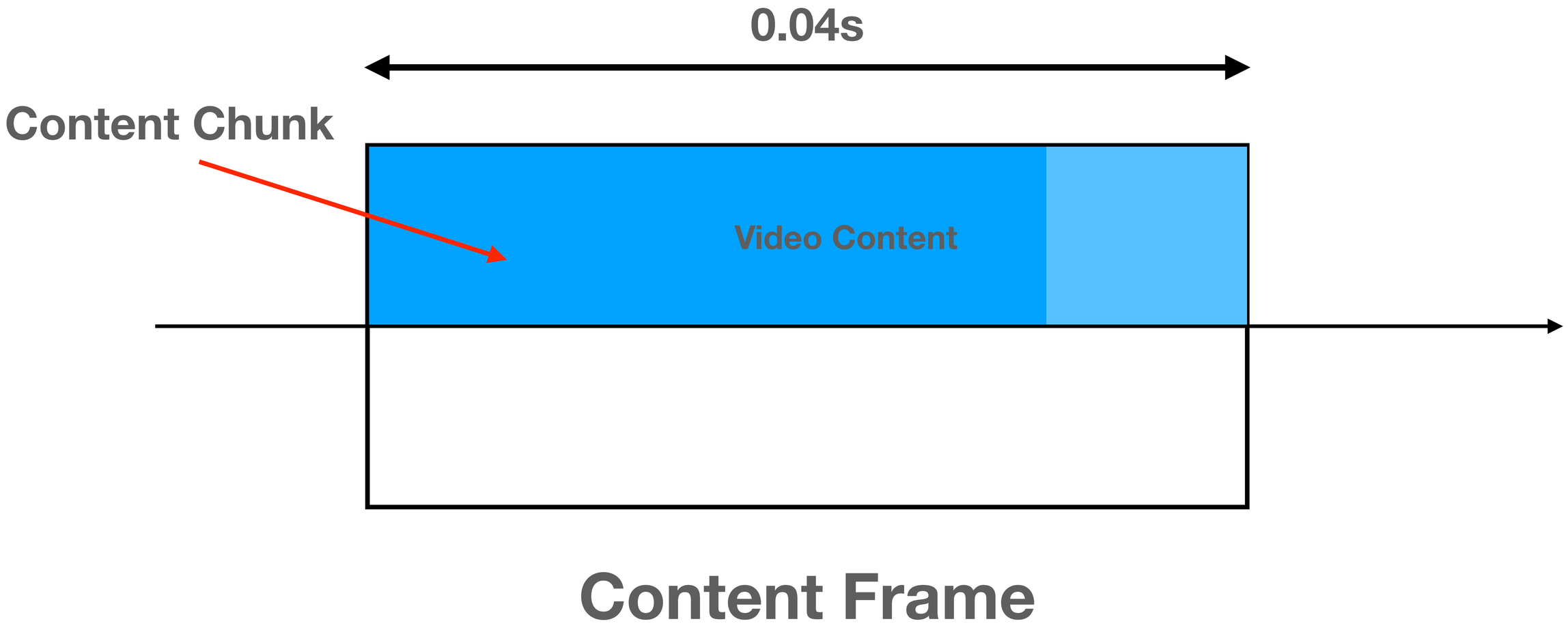
Context: Liquidsoap

New Content Representation



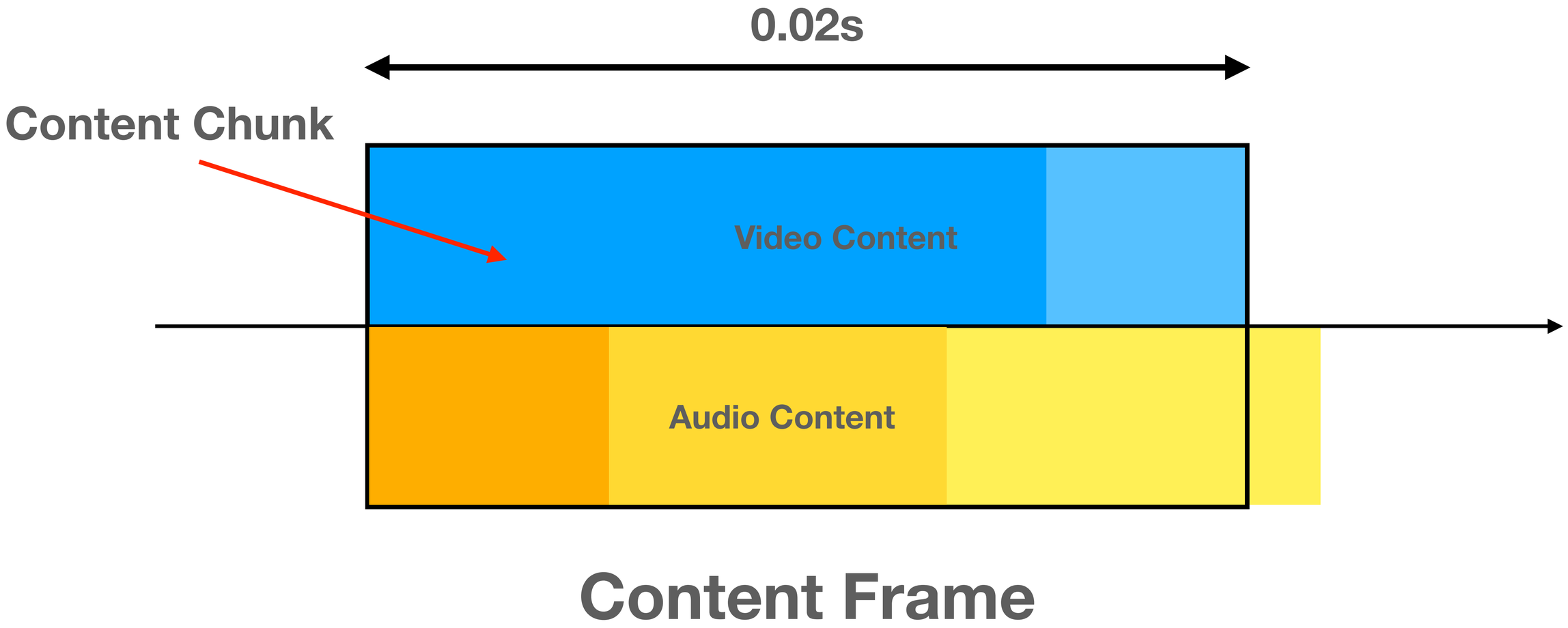
Context: Liquidsoap

New Content Representation



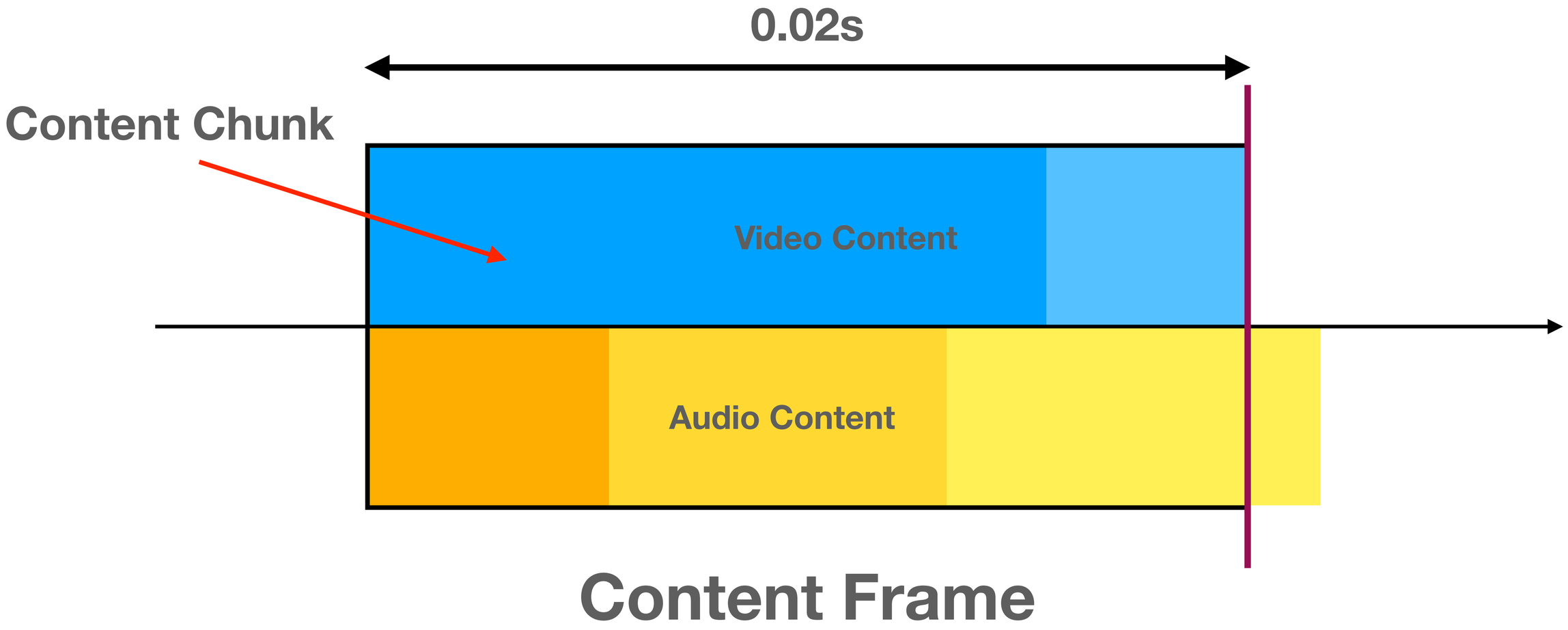
Context: Liquidsoap

New Content Representation



Context: Liquidsoap

New Content Representation



A Unified Content Composition API?

- Abstract API to compose content
- Working with both decoded and encoded content
- Encapsulating the right limitations (not everything will work!)
- Reusable by many different projects

A Content Algebra

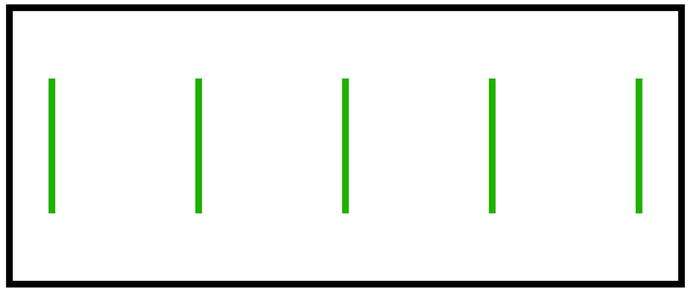
Content Elements

A Content Algebra

Content Elements

Frame: Decoded Content

- Audio: PCM array



- Video: Single Video Image

A Content Algebra

Content Elements

Packet: Encoded Content

A Content Algebra

Content Elements

Packet: Encoded Content

- MP3 Frame
- Ogg packet
- Video I-Frame, P-Frame, etc..

A Content Algebra

Content Elements

Packet: Encoded Content

- MP3 Frame
- Ogg packet
- Video I-Frame, P-Frame, etc..

Opaque! 📦

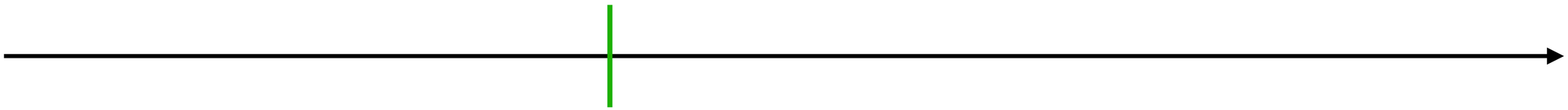
Unbreakable! 🔗

A Content Algebra

Content Chunk

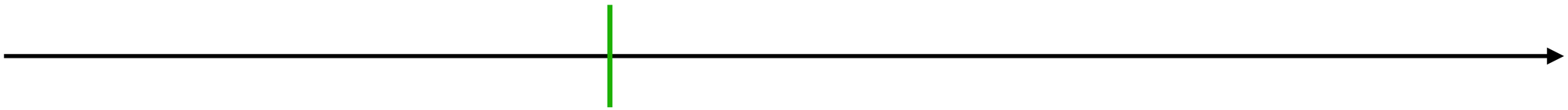
A Content Algebra

Content Chunk



A Content Algebra

Content Chunk

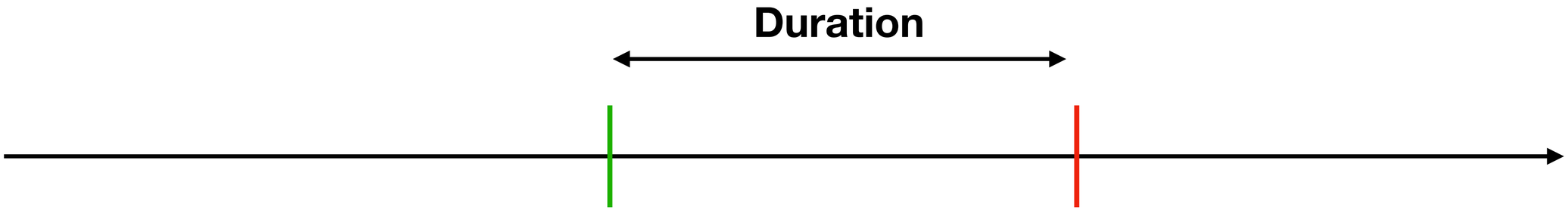


Frame: Presentation Time Stamp (PTS)

Packet: Decoder Time Stamp (DTS)

A Content Algebra

Content Chunk

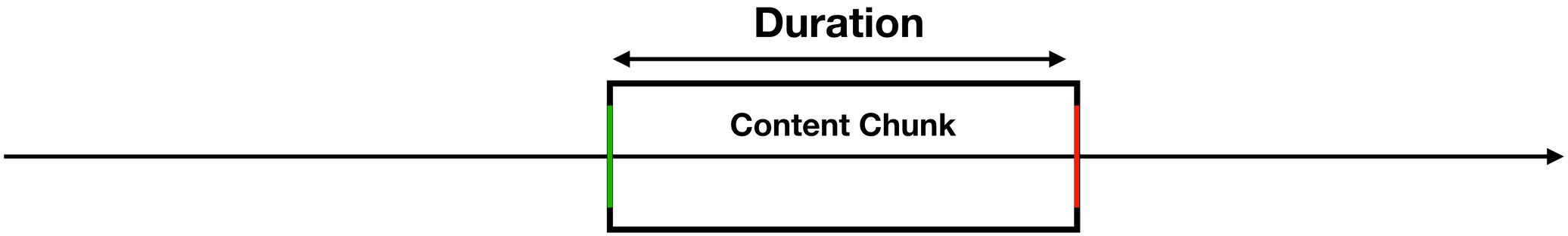


Frame: Presentation Time Stamp (PTS)

Packet: Decoder Time Stamp (DTS)

A Content Algebra

Content Chunk

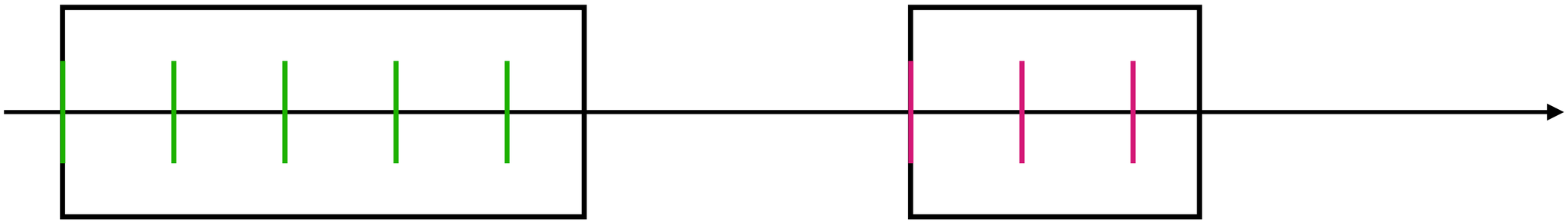


Frame: Presentation Time Stamp (PTS)

Packet: Decoder Time Stamp (DTS)

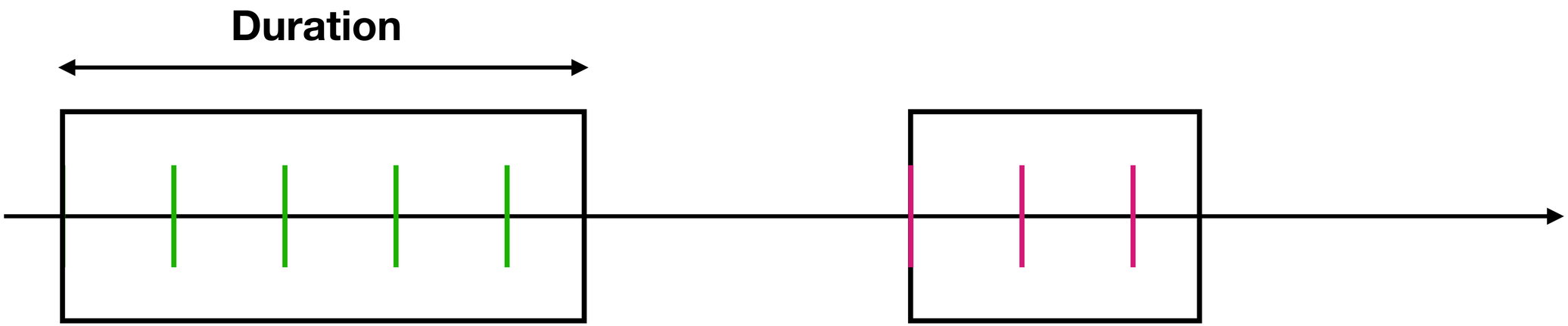
A Content Algebra

Content Composition: concatenation



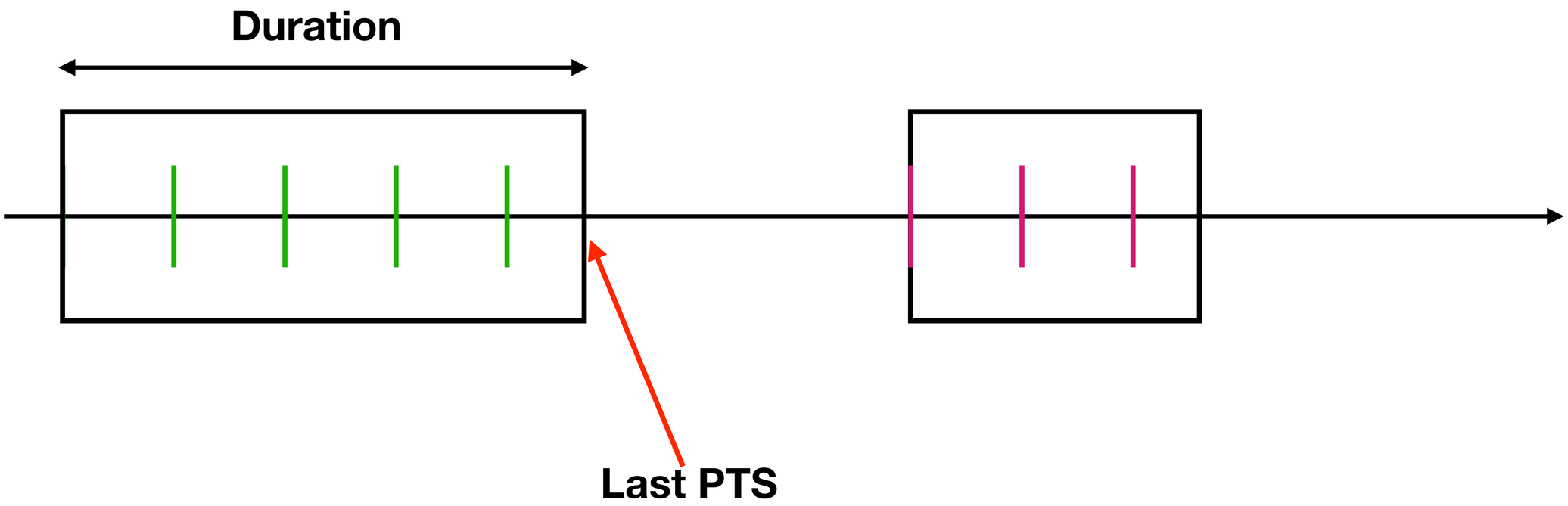
A Content Algebra

Content Composition: concatenation



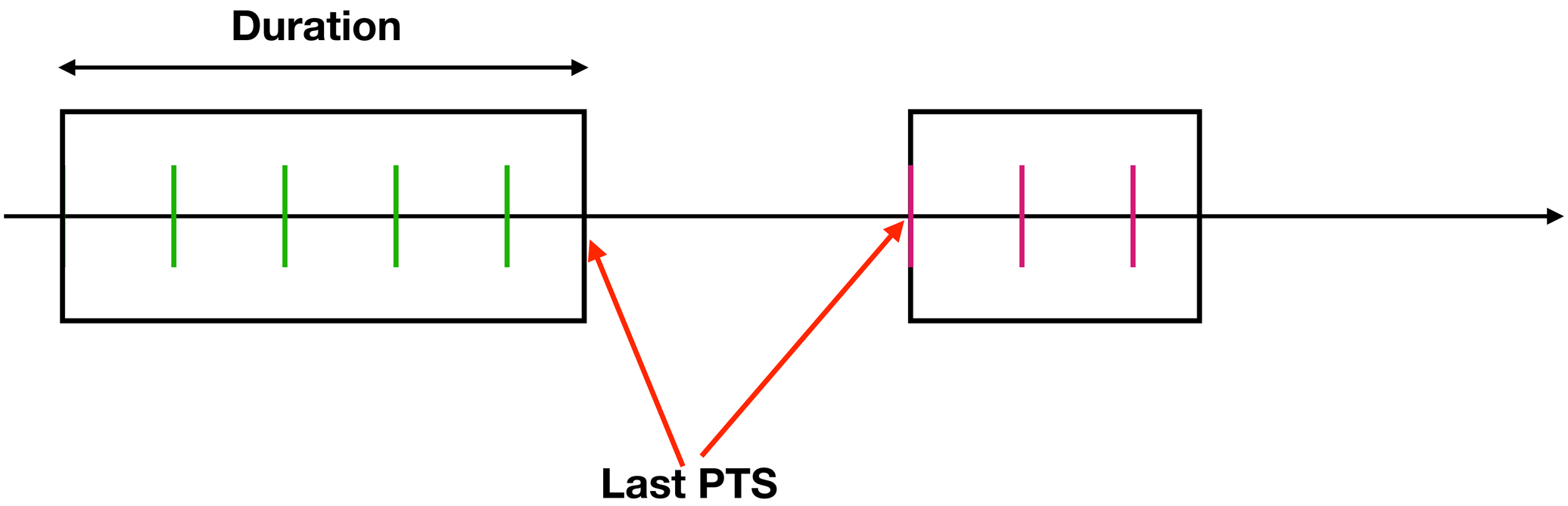
A Content Algebra

Content Composition: concatenation



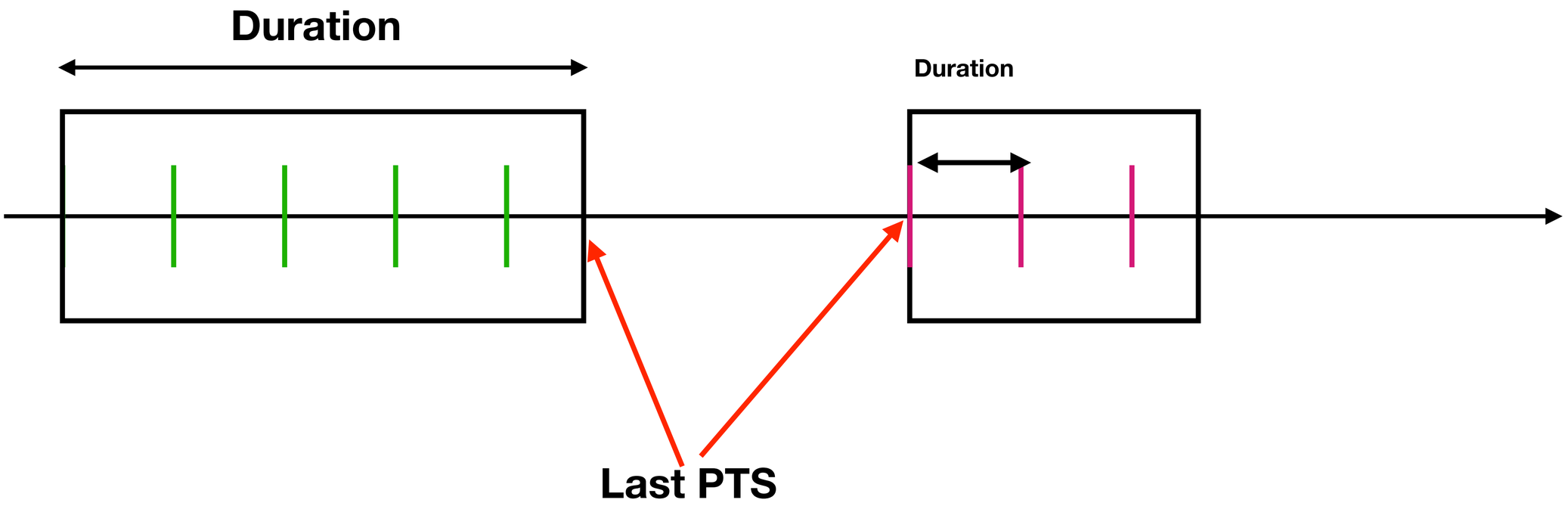
A Content Algebra

Content Composition: concatenation



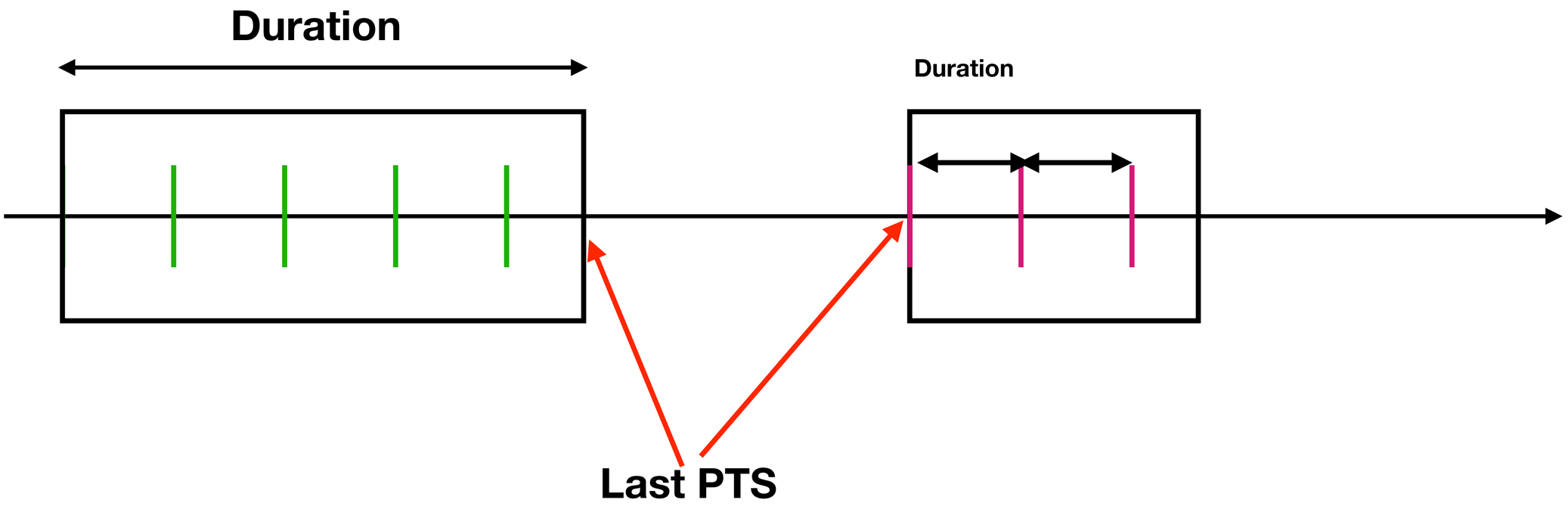
A Content Algebra

Content Composition: concatenation



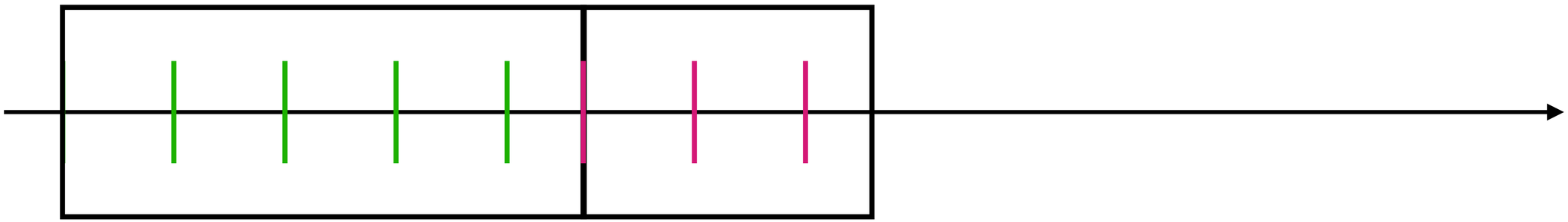
A Content Algebra

Content Composition: concatenation



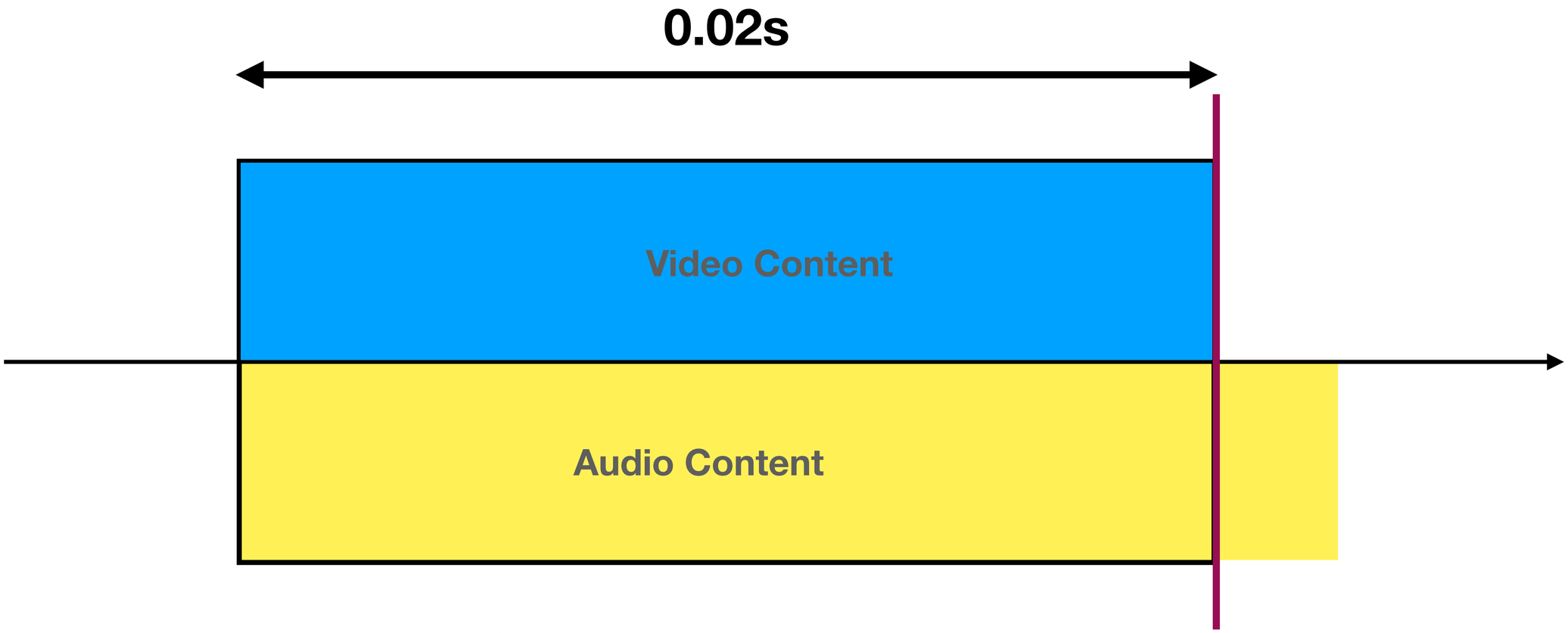
A Content Algebra

Content Composition: concatenation



A Content Algebra

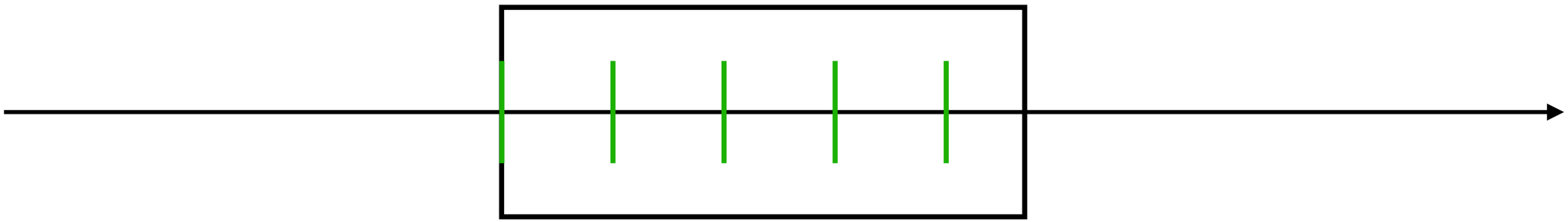
Content Composition: slicing



Content Frame

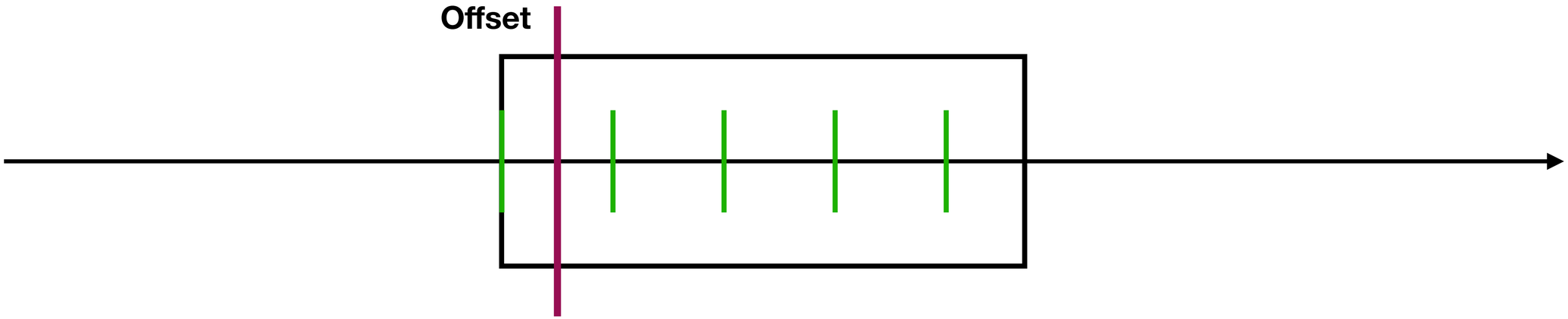
A Content Algebra

Content Composition: slicing



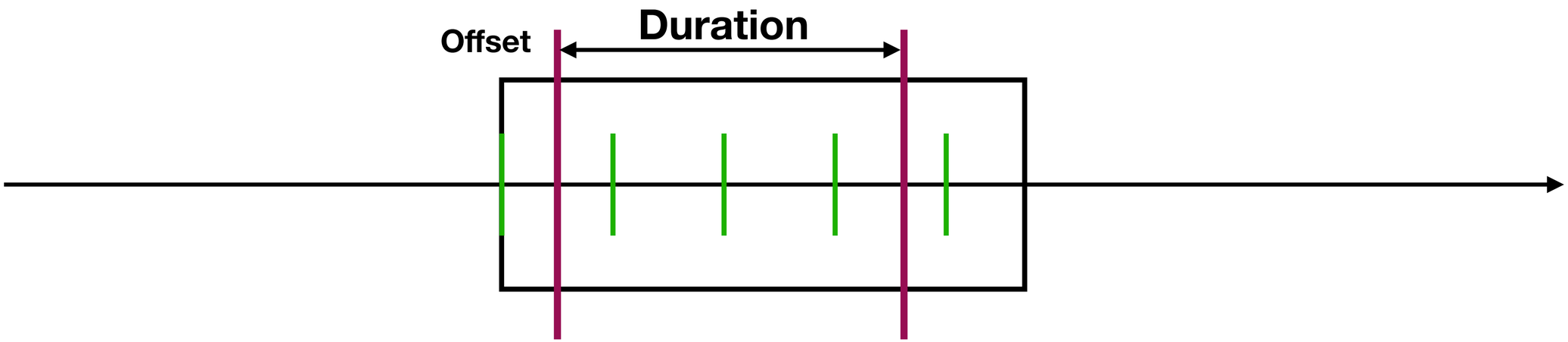
A Content Algebra

Content Composition: slicing



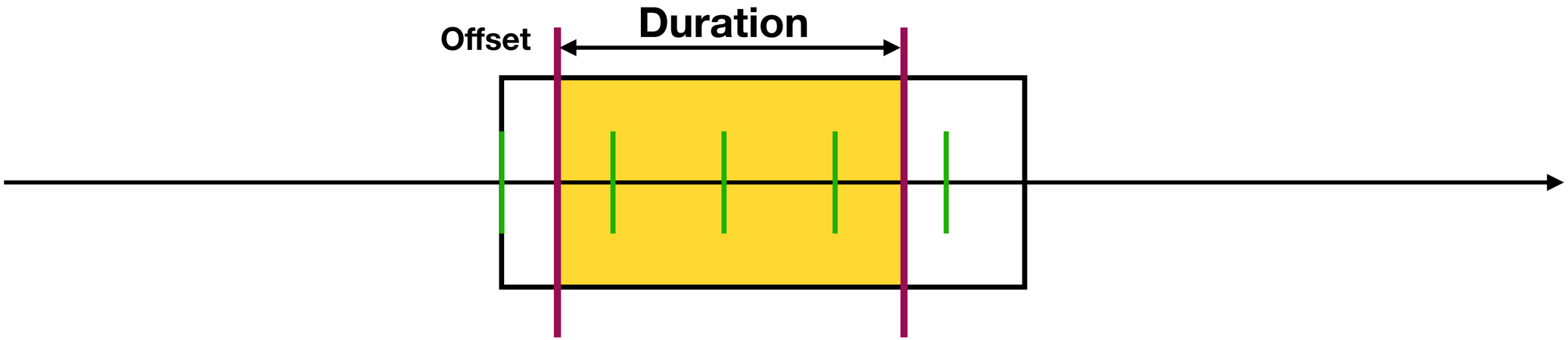
A Content Algebra

Content Composition: slicing



A Content Algebra

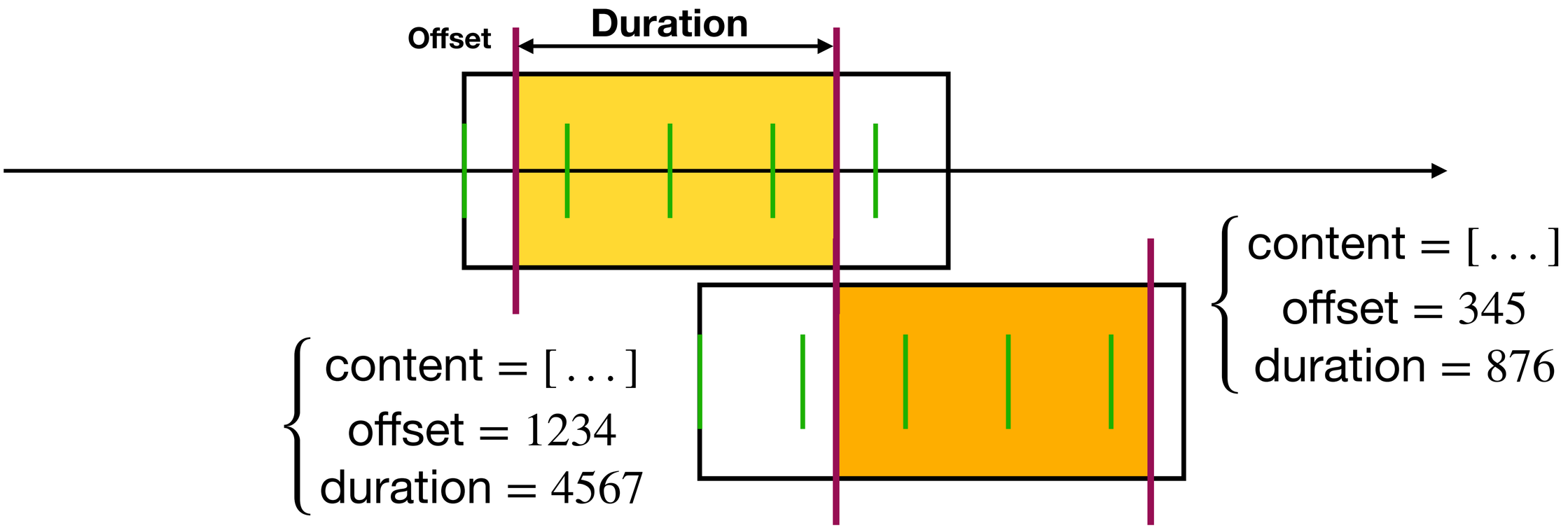
Content Composition: slicing



$$\begin{cases} \text{content} = [\ldots] \\ \text{offset} = 1234 \\ \text{duration} = 4567 \end{cases}$$

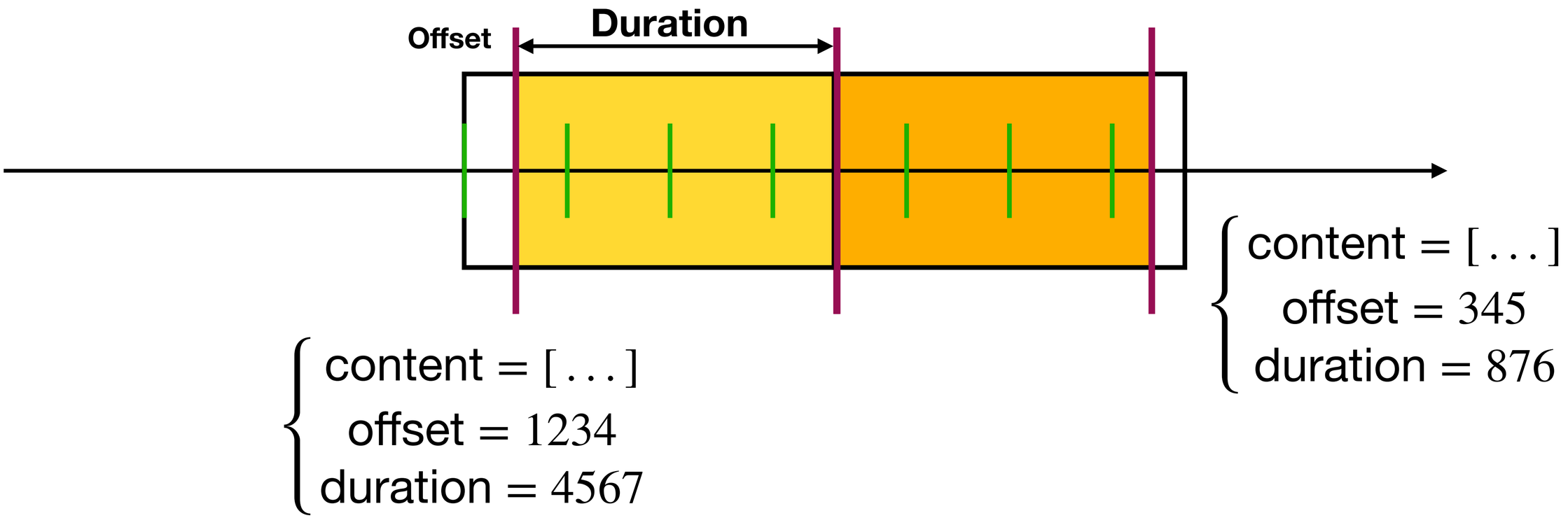
A Content Algebra

Content Composition: slicing



A Content Algebra

Content Composition: slicing



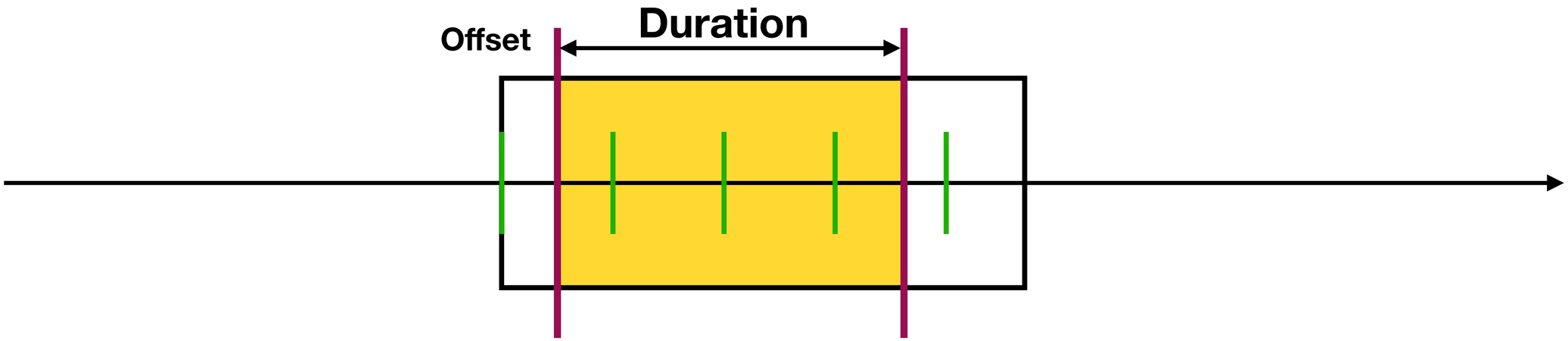
A Content Algebra

Content Composition: the hard bits

A Content Algebra

Content Composition: the hard bits

Minimal chunk size:

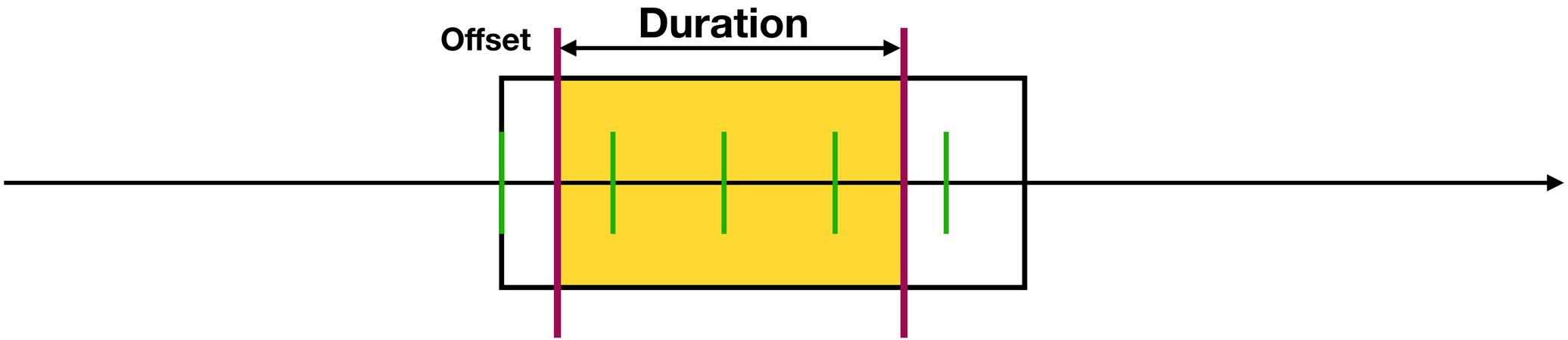


A Content Algebra

Content Composition: the hard bits

Minimal chunk size:

- **PCM Audio: 1 / sample_rate**

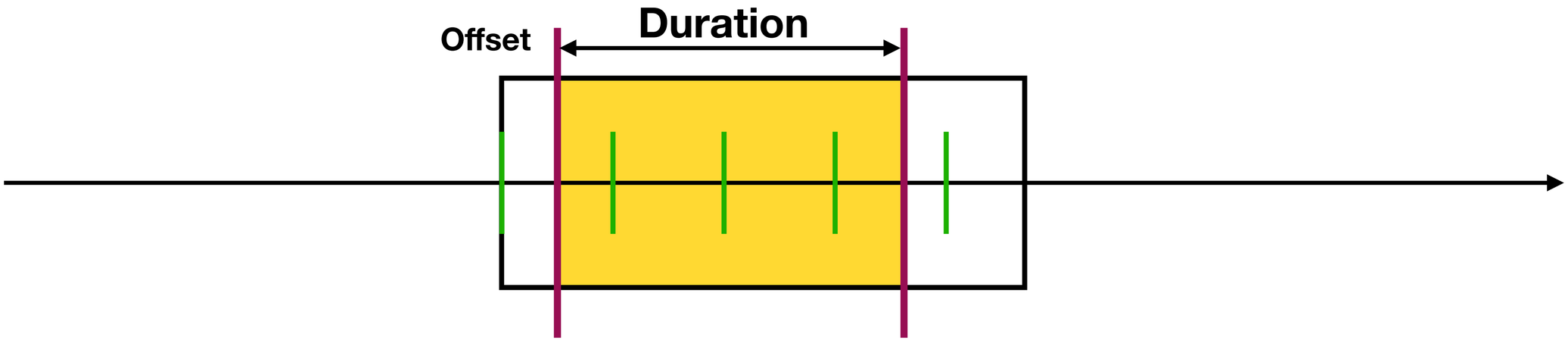


A Content Algebra

Content Composition: the hard bits

Minimal chunk size:

- **PCM Audio: 1 / sample_rate**
- **Raw video: 1 / frame_rate**

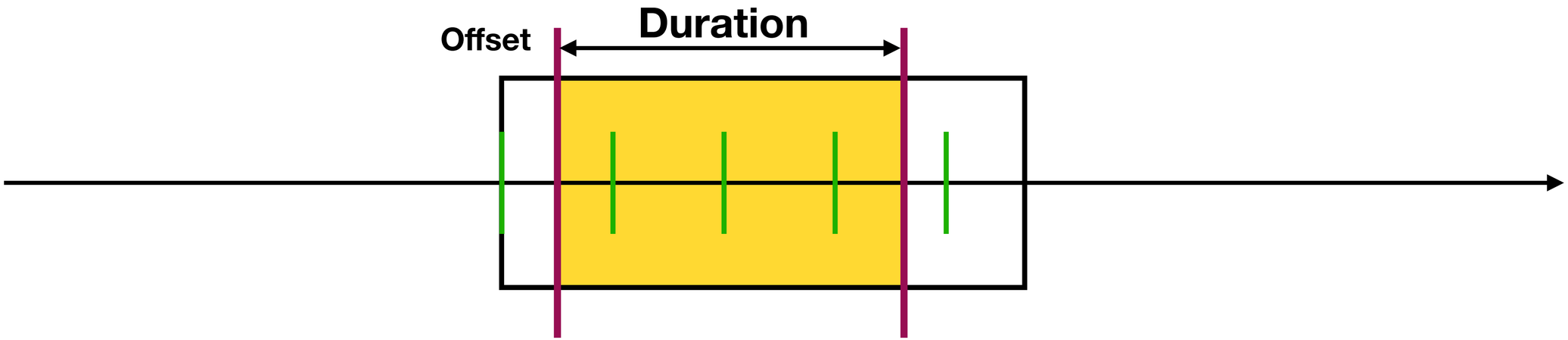


A Content Algebra

Content Composition: the hard bits

Minimal chunk size:

- PCM Audio: 1 / sample_rate
- Raw video: 1 / frame_rate
- Encoded packets ??



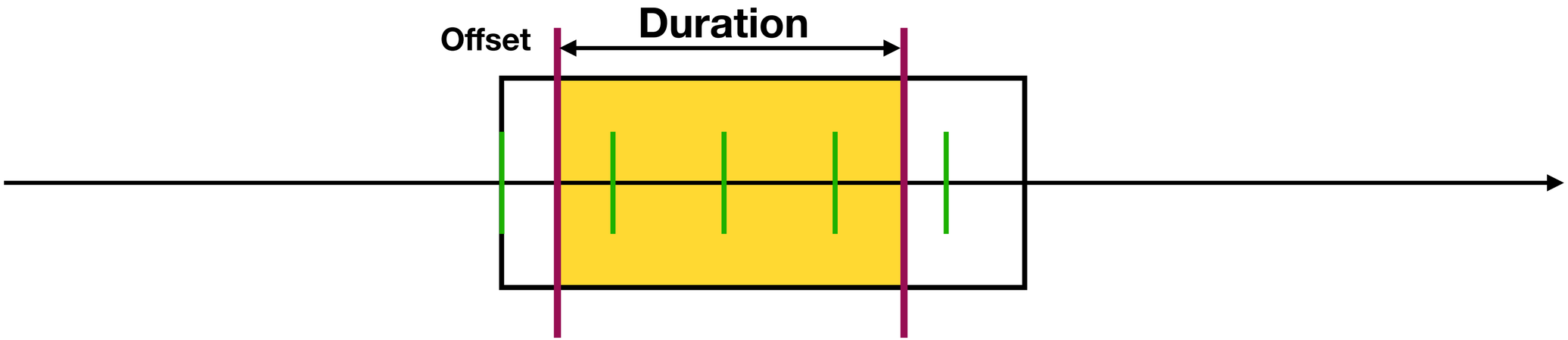
A Content Algebra

Content Composition: the hard bits

main rate = sample rate

Minimal chunk size:

- PCM Audio: 1 / sample_rate
- Raw video: 1 / frame_rate
- Encoded packets ??



A Content Algebra

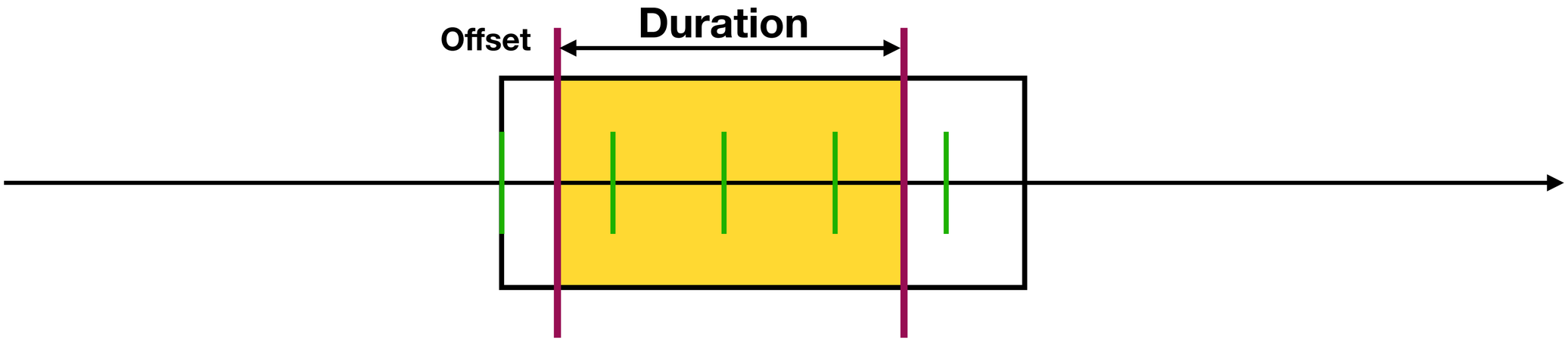
Content Composition: the hard bits

main rate = sample rate

Minimal chunk size:

- **PCM Audio: 1 / sample_rate**
- **Raw video: 1 / frame_rate**
- **Encoded packets ??**

nearest image resampling



A Content Algebra

Content Composition: the hard bits

Minimal chunk size:

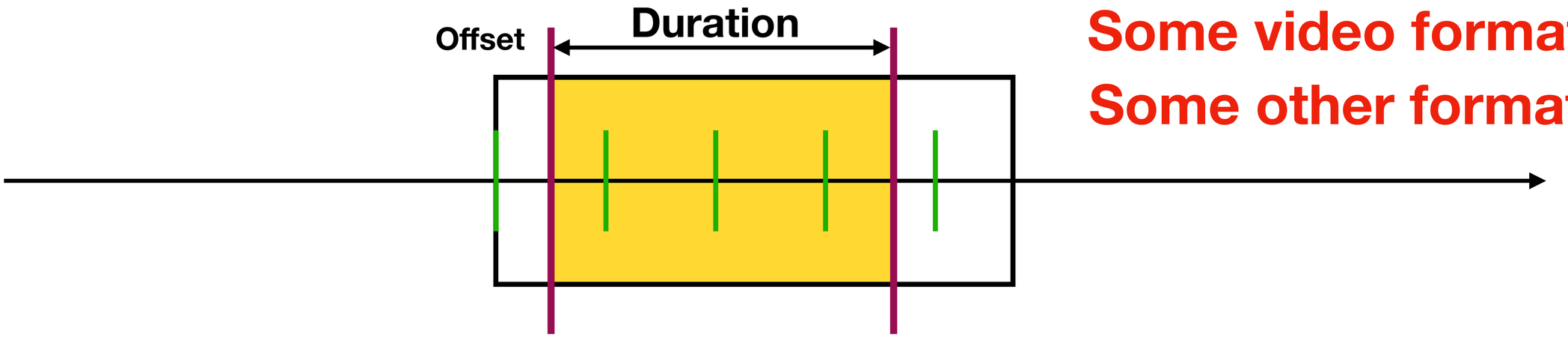
- PCM Audio: 1 / sample_rate
- Raw video: 1 / frame_rate
- Encoded packets ??

main rate = sample rate

nearest image resampling

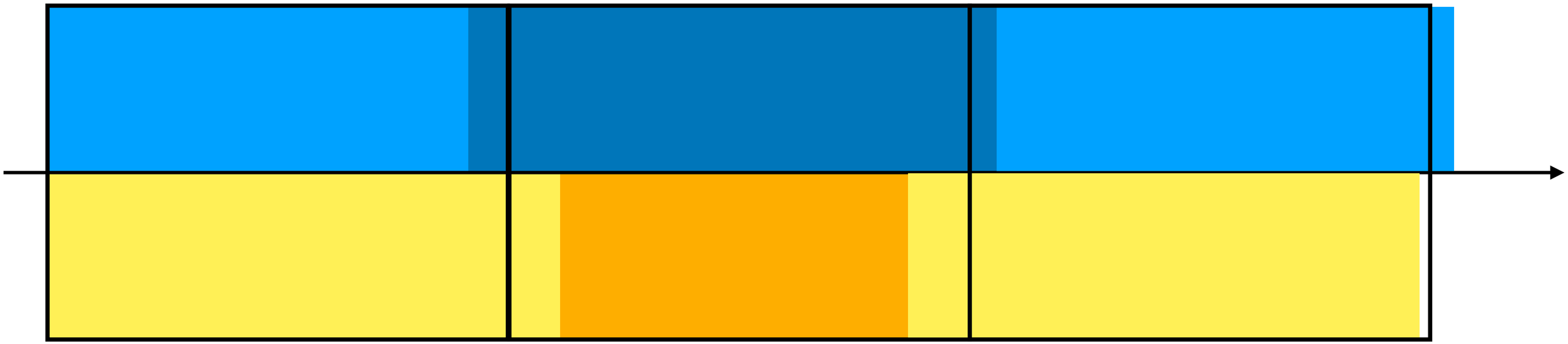
Some video formats: ✅

Some other formats: 🚫



A Content Algebra

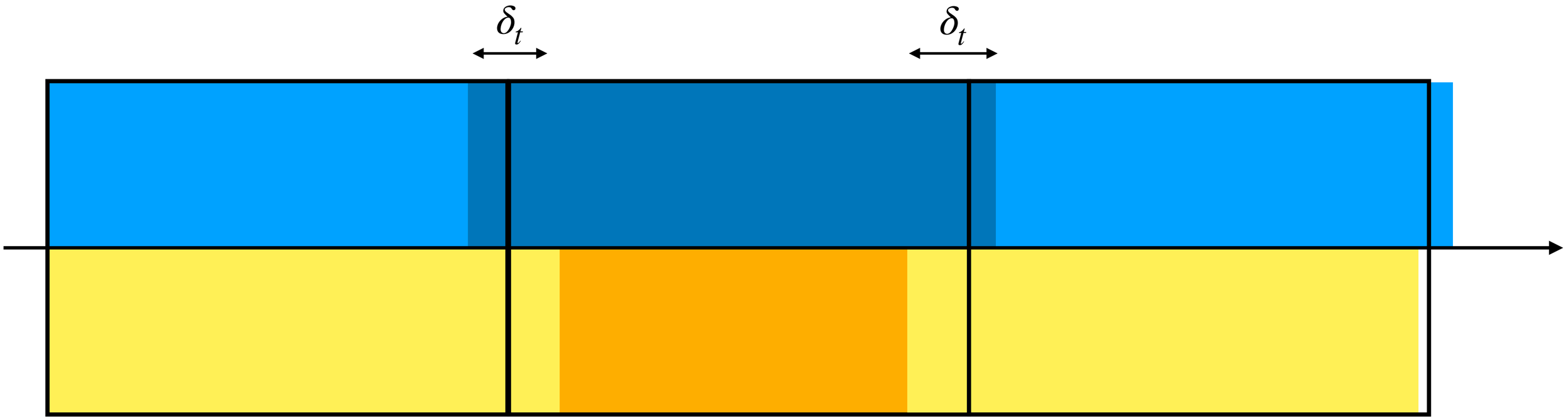
Content Composition: the hard bits



Content Frame

A Content Algebra

Content Composition: the hard bits

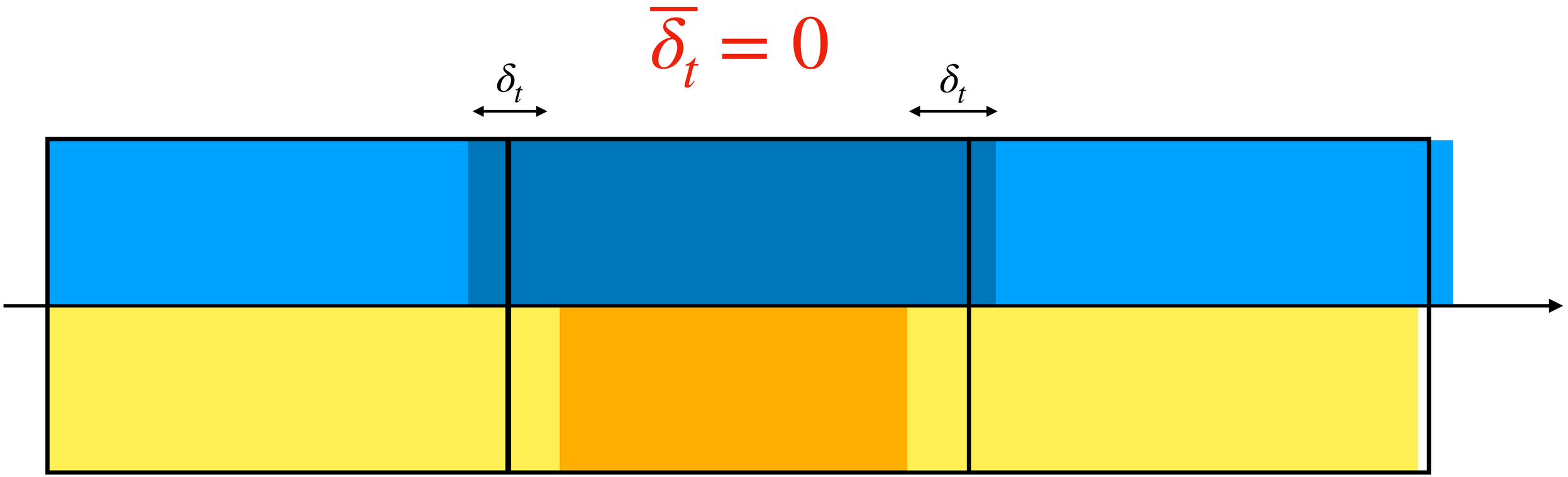


Content Frame

A Content Algebra

Content Composition: the hard bits

$$\overline{\delta_t} = 0$$



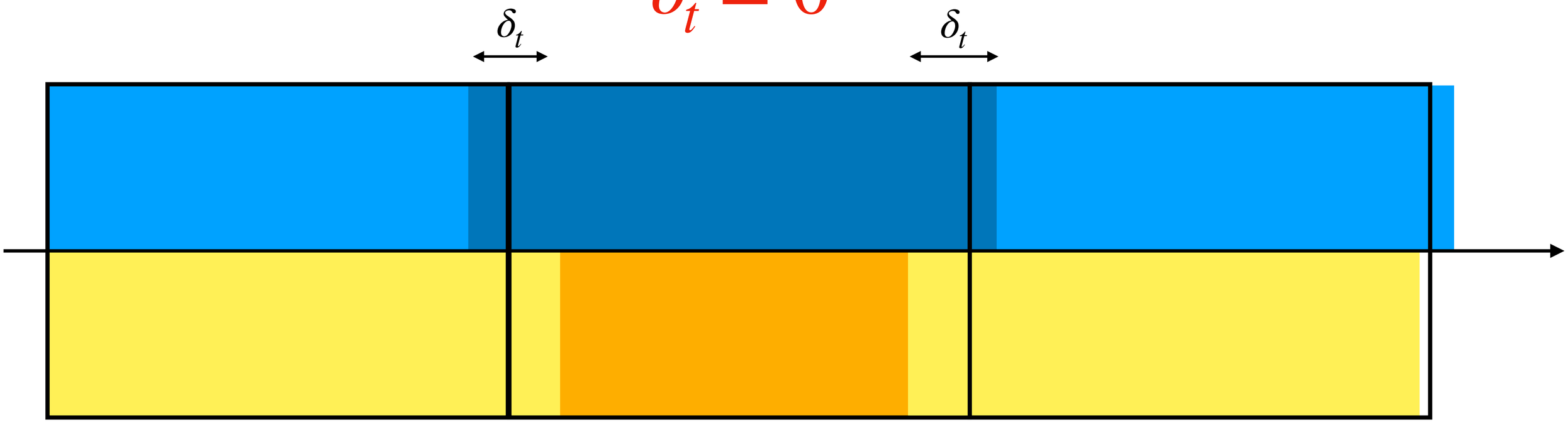
Content Frame

A Content Algebra

Content Composition: the hard bits

- **ATSC IS-191: -45ms to 15ms**
- **EBU R37-2007: -60ms to 40ms**
- **ITU BT.1359-1: -125ms to 45ms**
- **ITU BR.265-9: -22ms to 22ms**

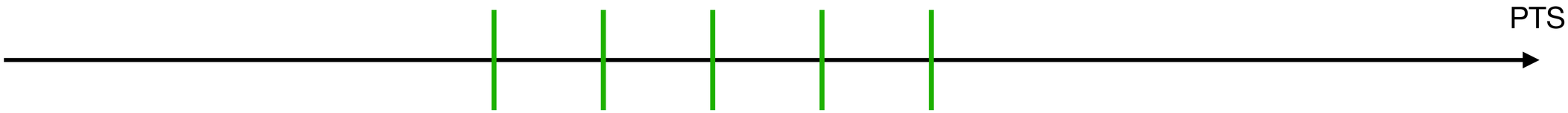
$$\overline{\delta_t} = 0$$



Content Frame

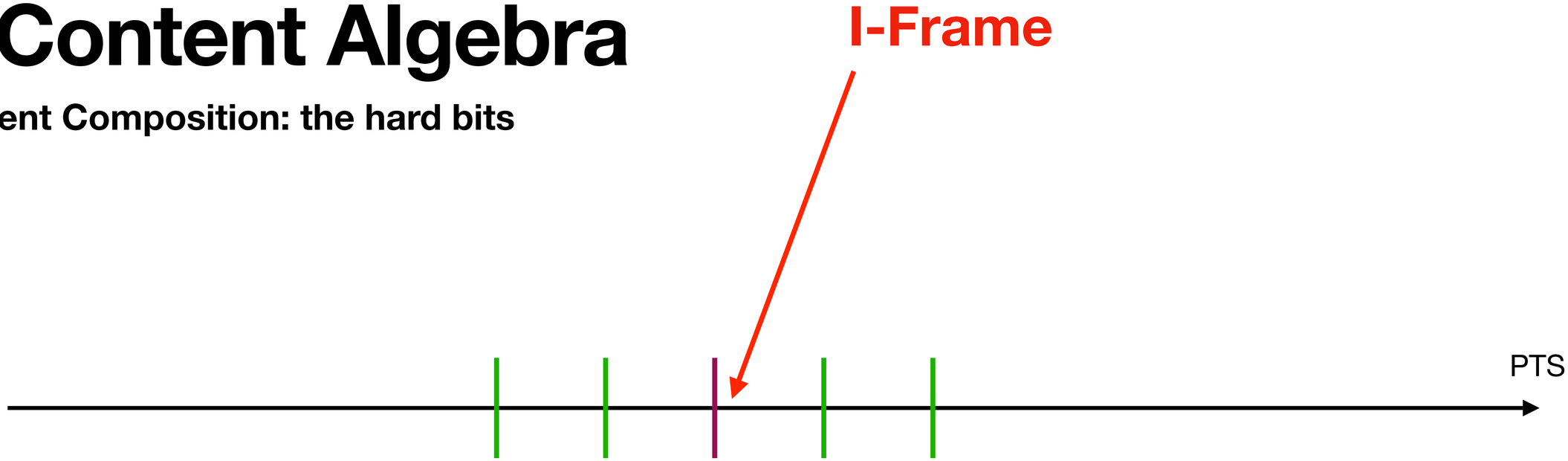
A Content Algebra

Content Composition: the hard bits



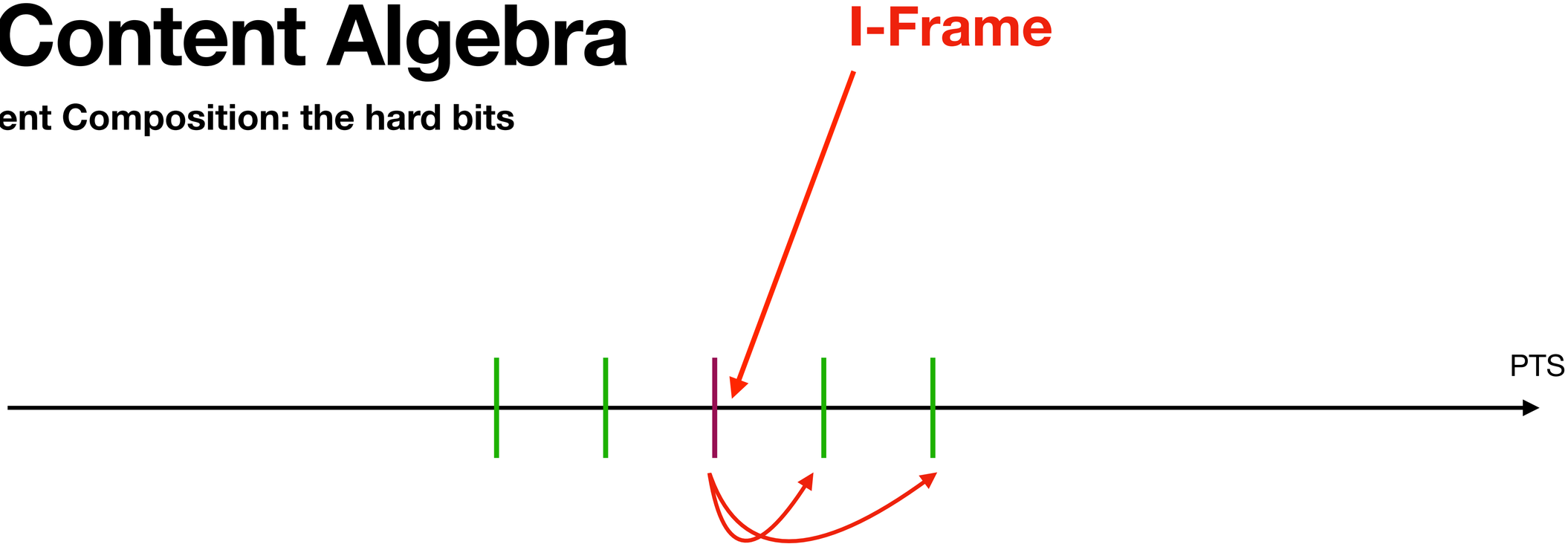
A Content Algebra

Content Composition: the hard bits



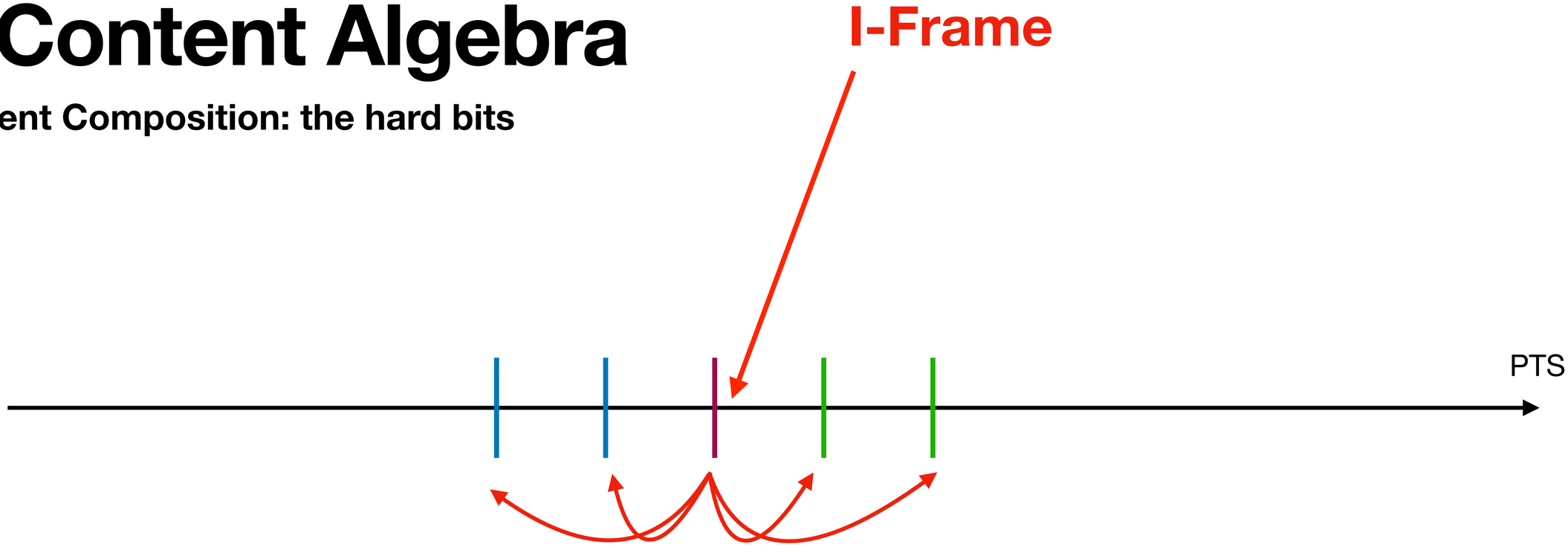
A Content Algebra

Content Composition: the hard bits



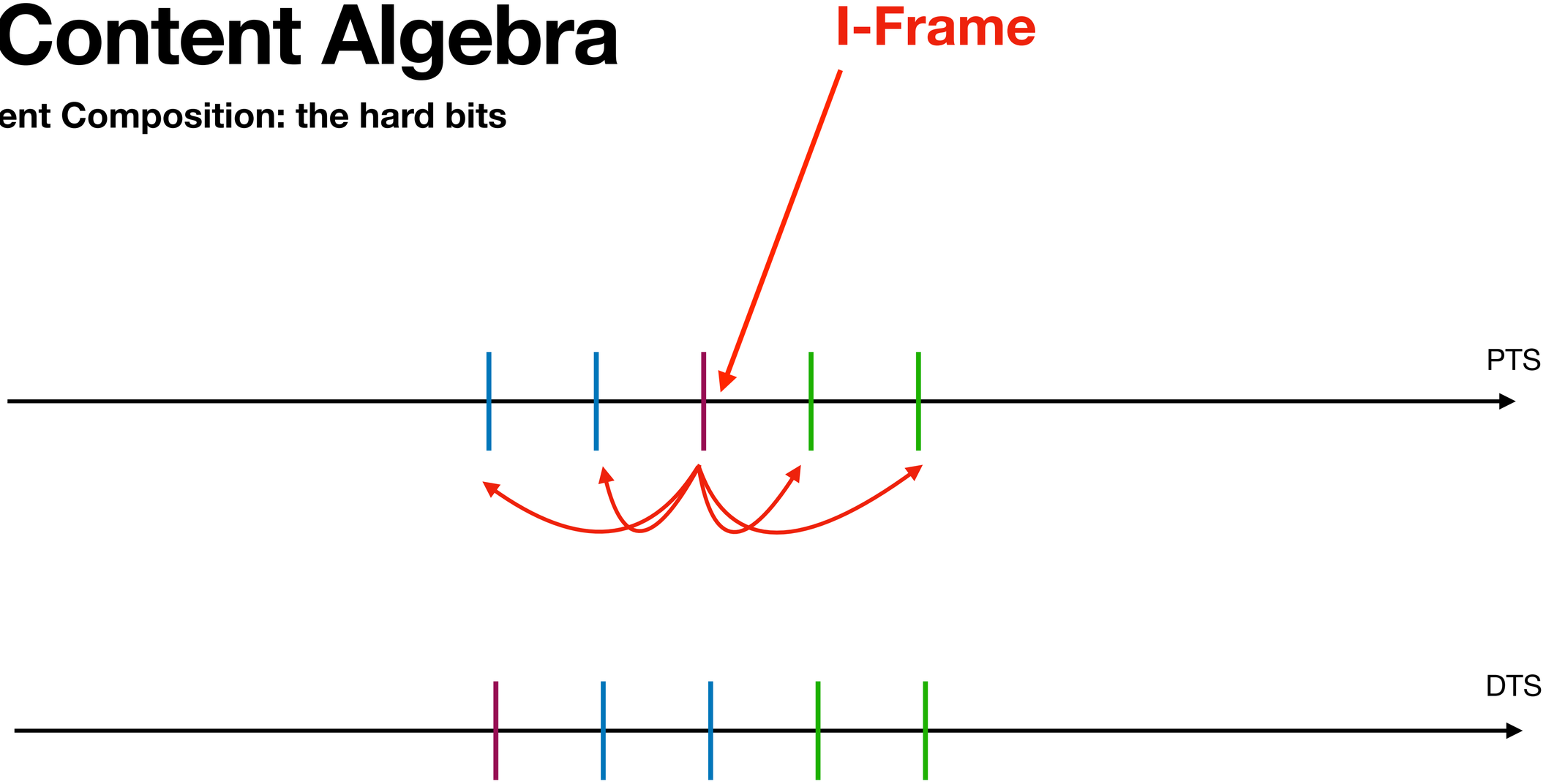
A Content Algebra

Content Composition: the hard bits



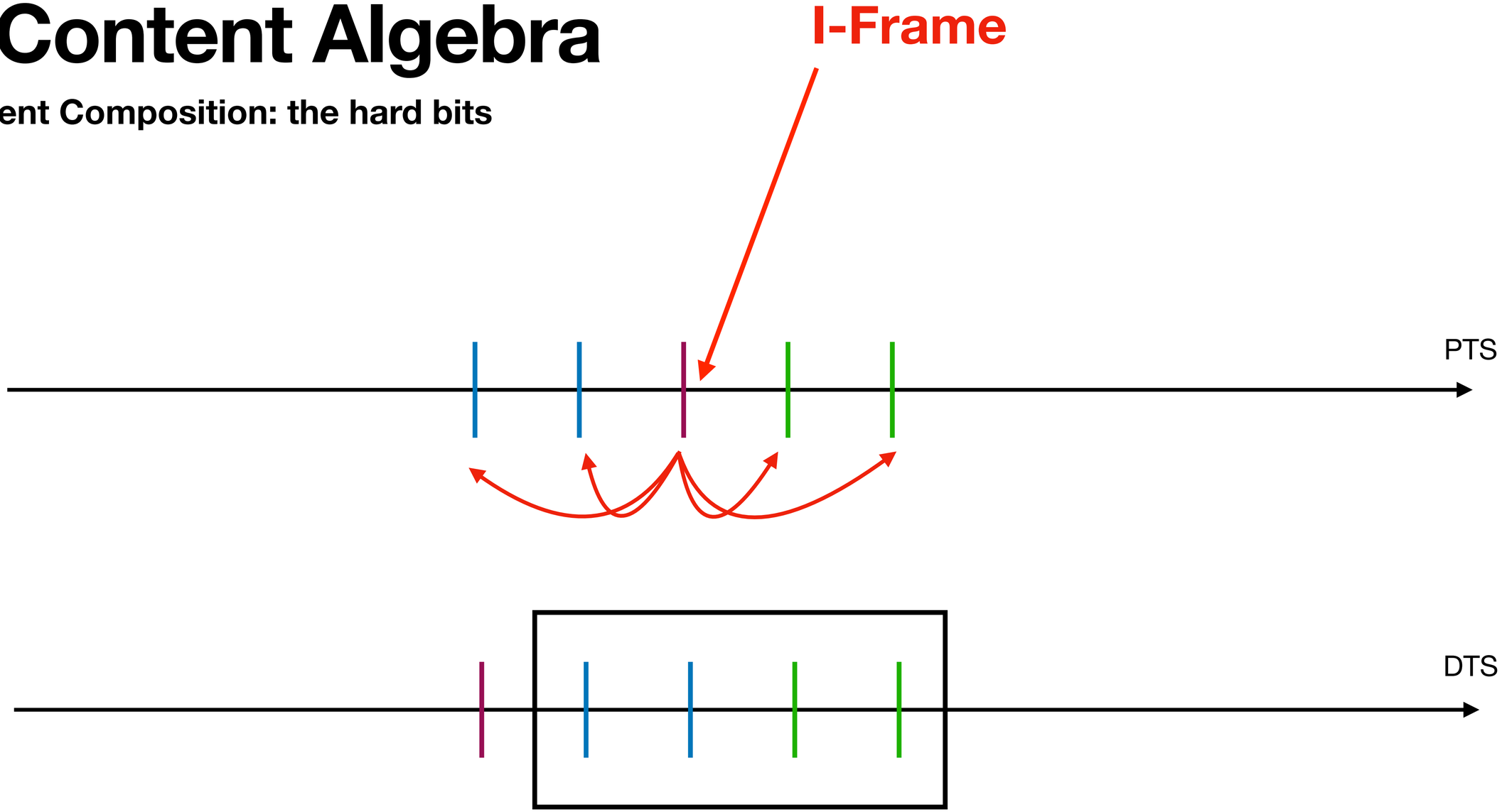
A Content Algebra

Content Composition: the hard bits



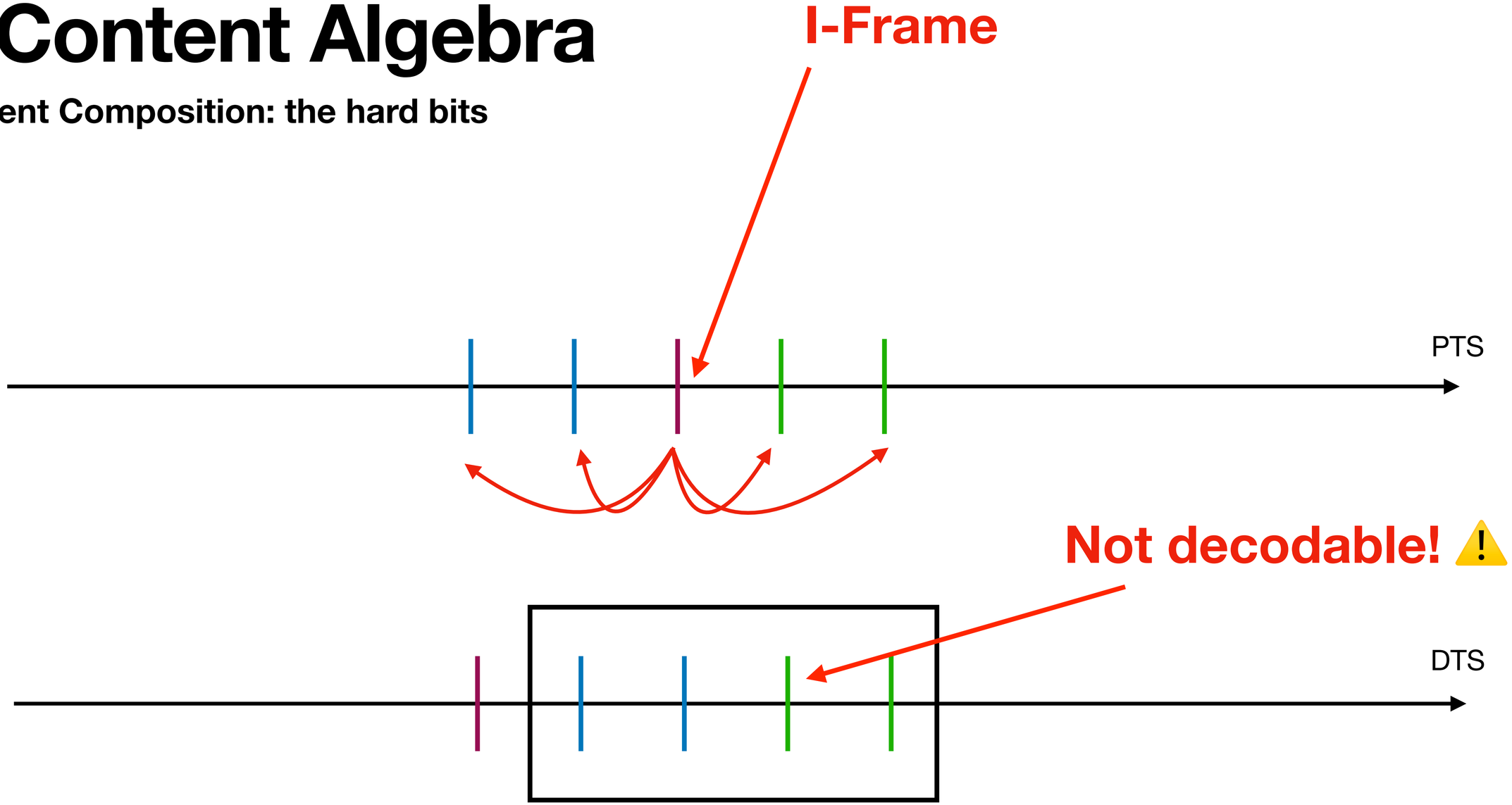
A Content Algebra

Content Composition: the hard bits



A Content Algebra

Content Composition: the hard bits



A Content Algebra

Content Composition: the hard bits

- Frame size change, i.e. 1280p to 1080p etc.?
- Pixel format change?
- Audio sample format change?
- Global codec header vs. per-frame headers (FFmpeg: extradata)
- Automatic bitstream manipulation e.g. FFmpeg bitstream filters: h264_mp4toannexb, extract_extradata, aac_adtstoasc, etc..
- Optimal encoder parameters (I-Frame frequency, etc.)

A Content Algebra

Content Composition: the hard bits

Questions?

Remarks?