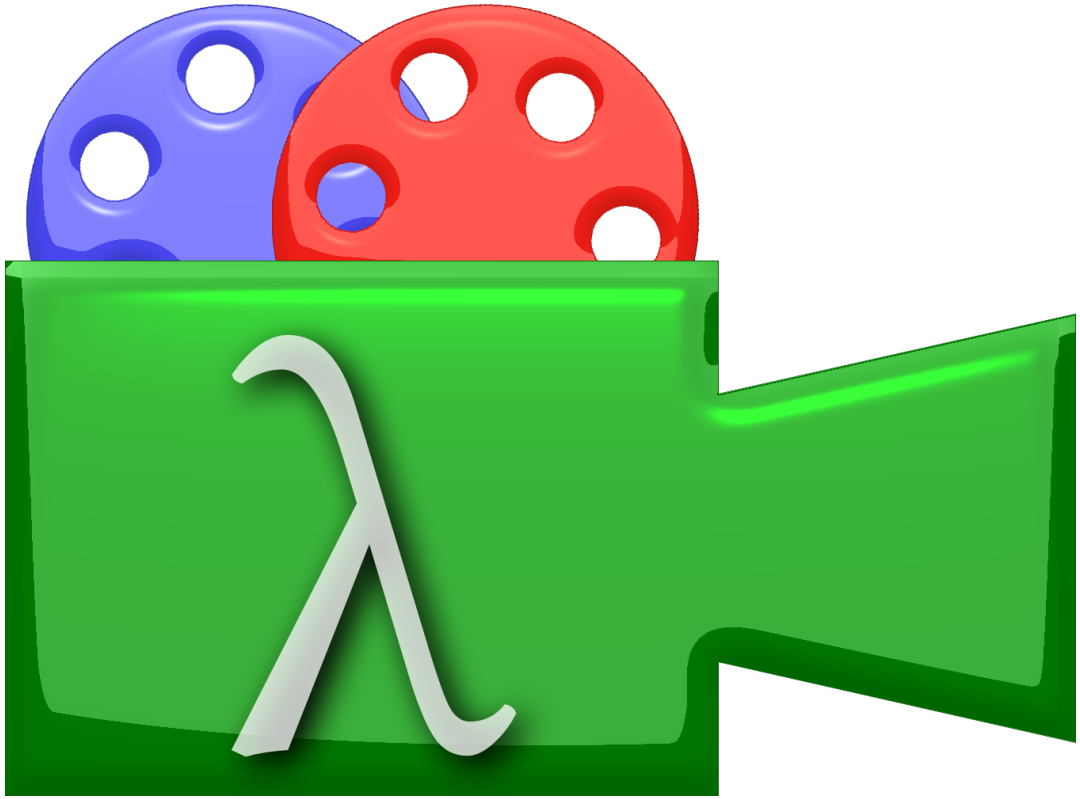
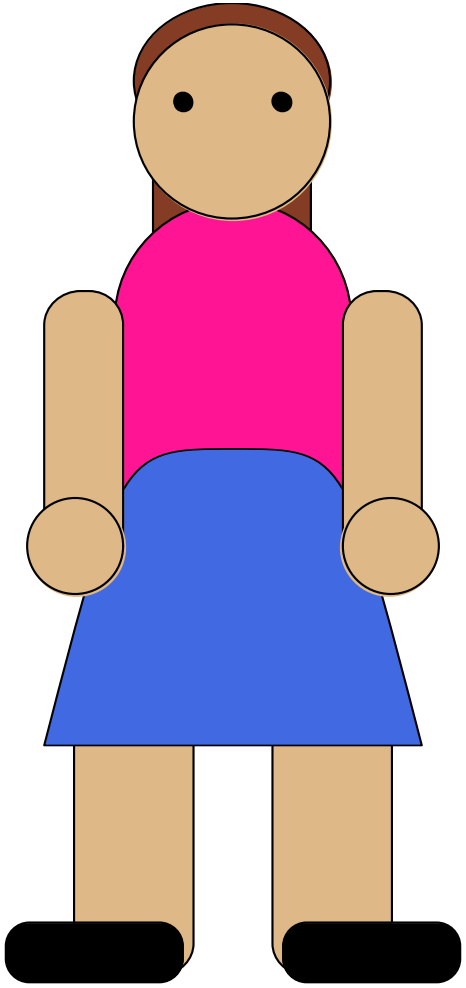
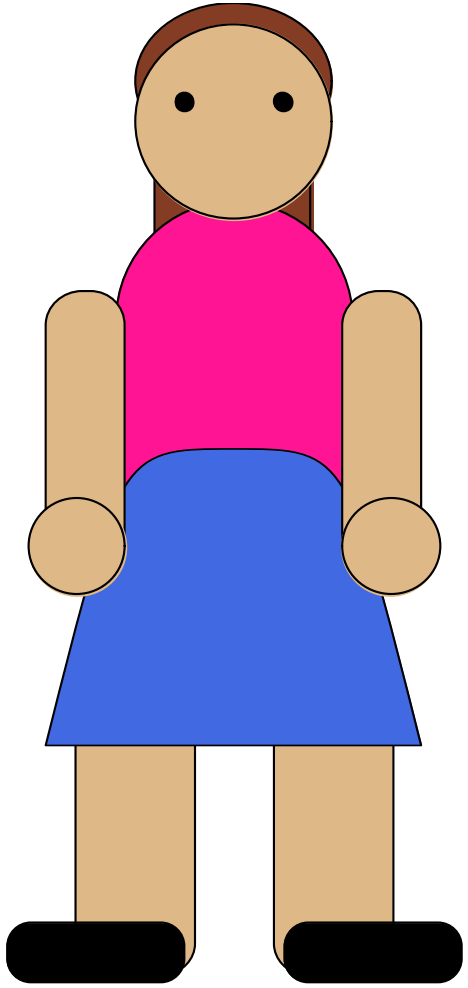
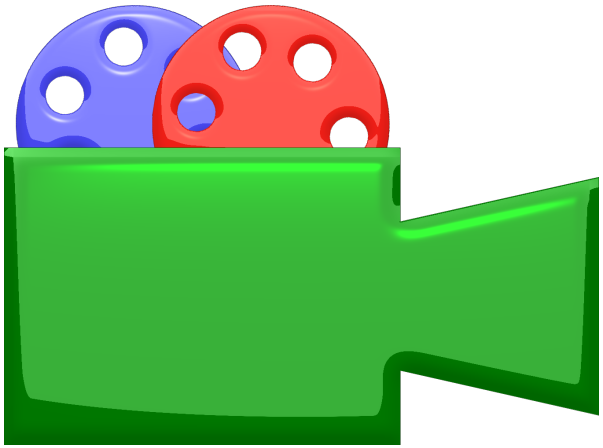
Movies as Programs

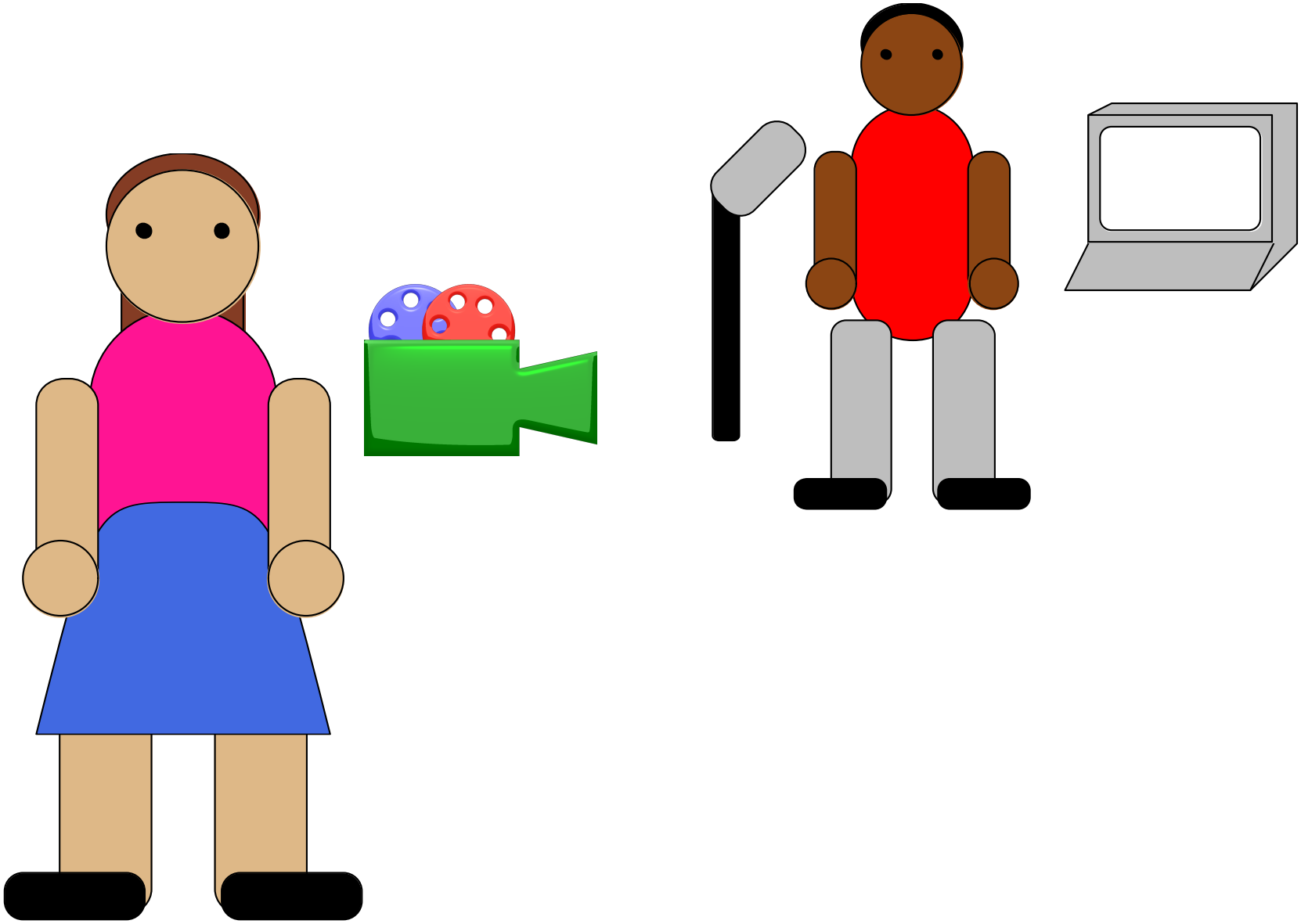


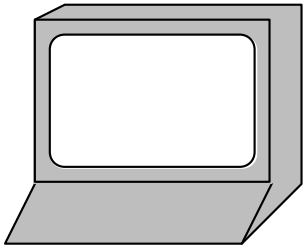
Leif Andersen

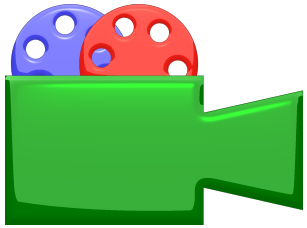


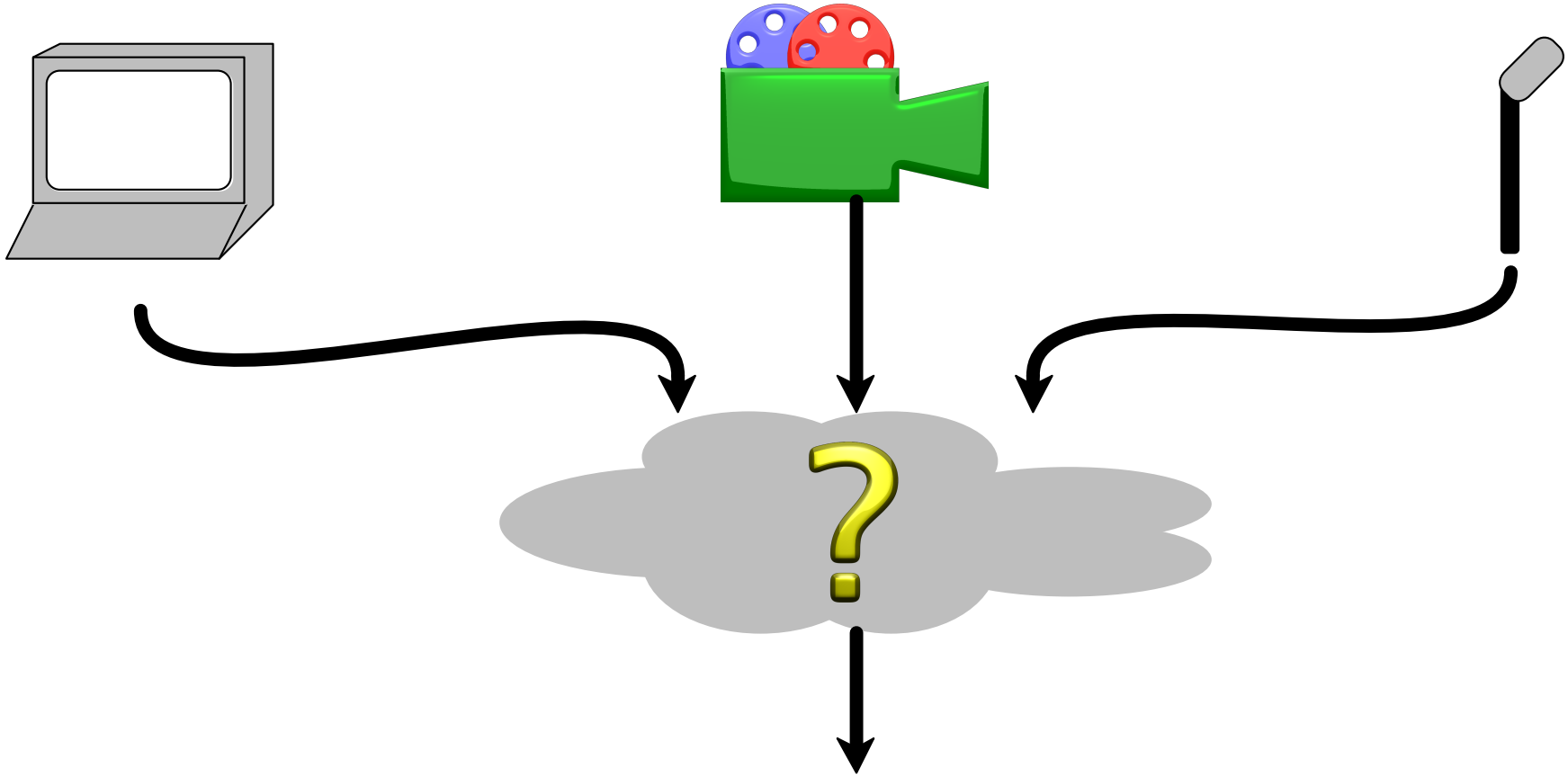


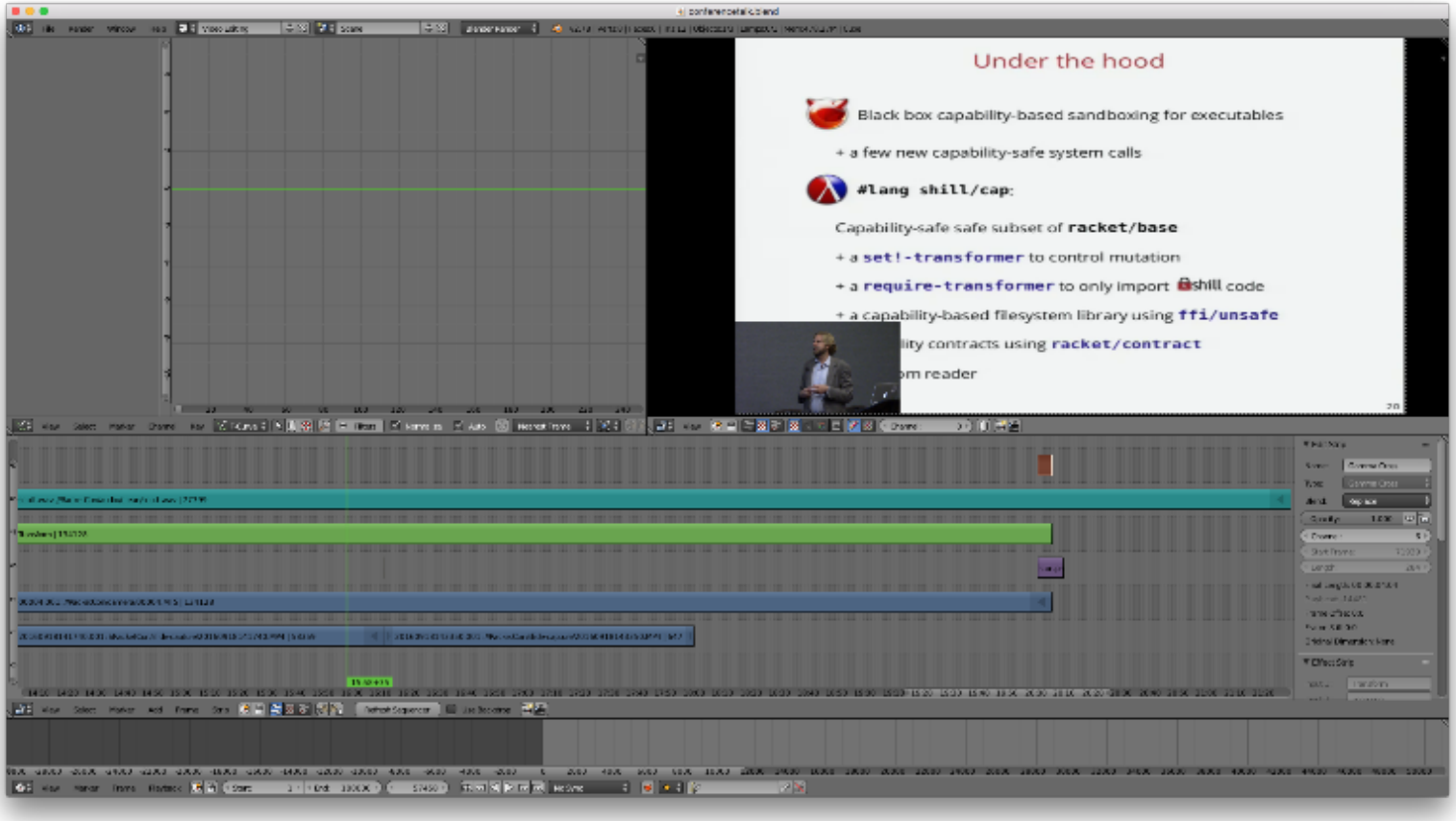











Under the hood
Black box capability-based sandboxing for executables
+ a few new capability-safe system calls
#lang shill/cap:
Capability-safe safe subset of racket/base
+ a set!-transformer to control mutation
+ a require-transformer to only import shill code
+ a capability-based filesystem library using ffi/unsafe
racket/contract
reader

One down

One down

19 more to go…

We Need Automation

The Landscape

Tool	Example	Experience
Plugin-Ins	Blender Script, AE Script	
UI Automation (Macros)	Apple Script	
Shell Scripts	FFmpeg, AVISynth	

The Landscape

Tool	Example	Experience
Plugin-Ins	Blender Script, AE Script	
UI Automation (Macros)	Apple Script	
Shell Scripts	FFmpeg, AVISynth	

The Landscape

Tool	Example	Experience
Plugin-Ins	Blender Script, AE Script	
UI Automation (Macros)	Apple Script	
Shell Scripts	FFmpeg, AVISynth	

The Landscape

Tool	Example	Experience
Plugin-Ins	Blender Script, AE Script	
UI Automation (Macros)	Apple Script	
Shell Scripts	FFmpeg, AVISynth	

We have a problem...

We have a problem...

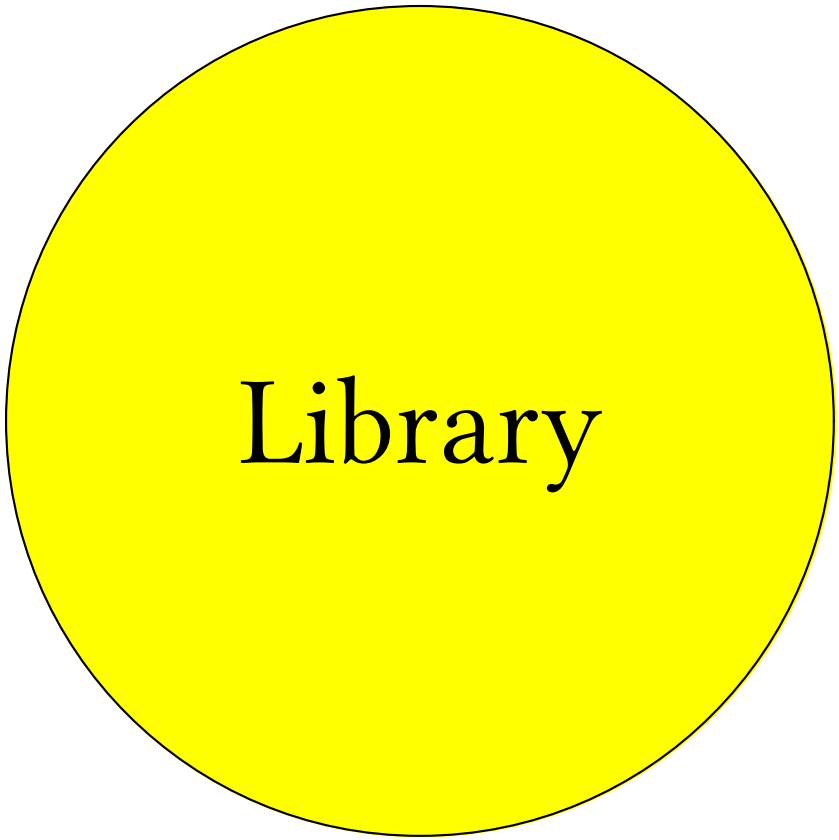
We want to solve it in the problem domain's own language...

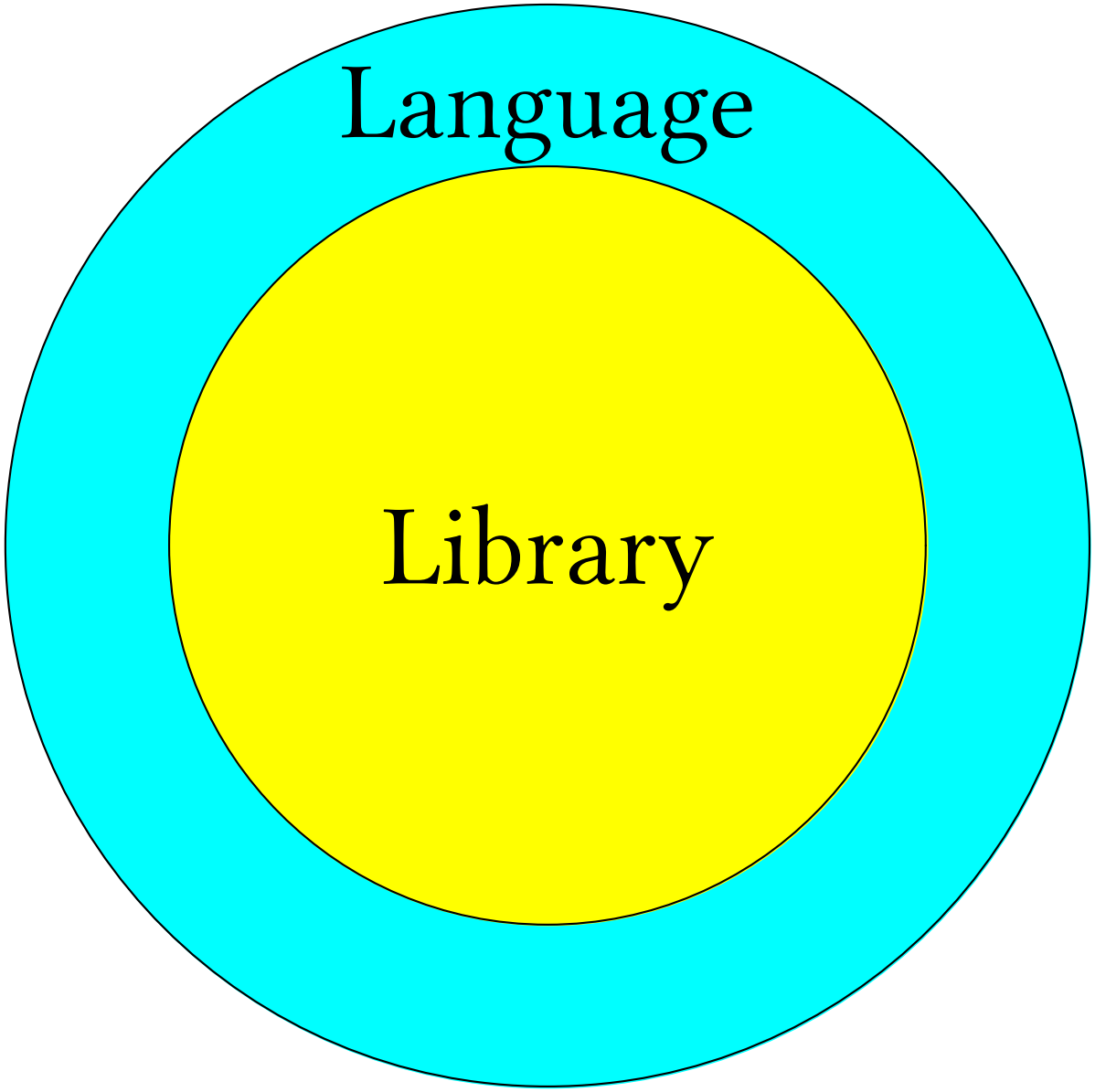
We have a problem...

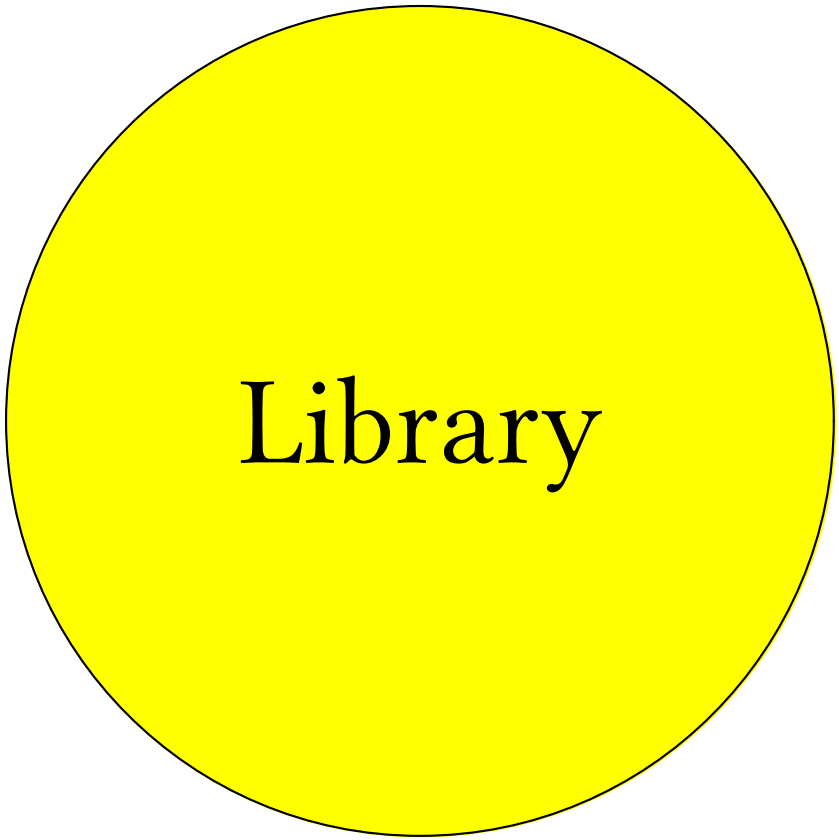
We want to solve it in the problem domain's own language...

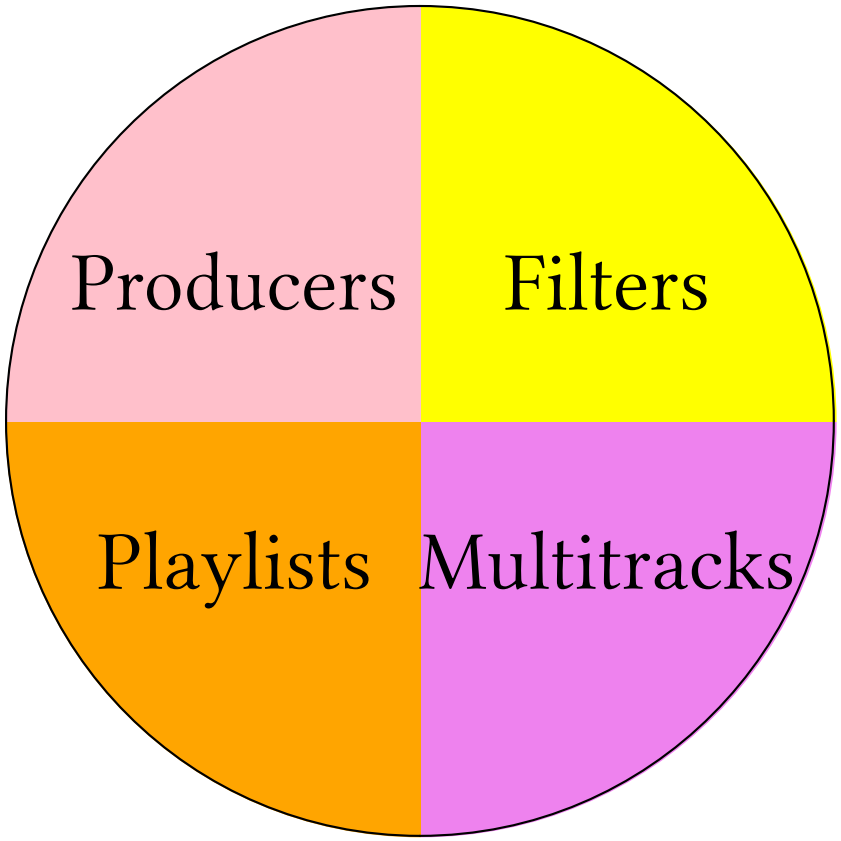
Make a DSL!

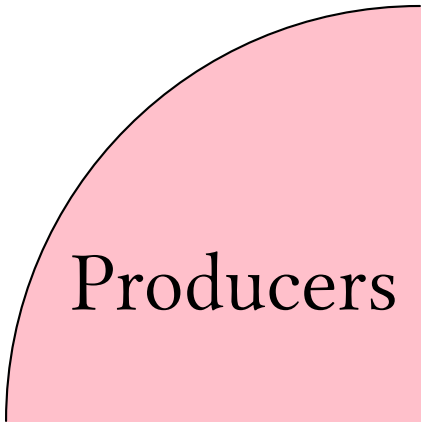
Make a DSL!


Library


Language
Library

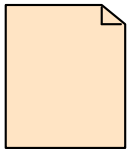

Library


Producers
Filters
Playlists
Multitracks

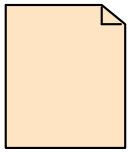

Producers

Producers

Producers

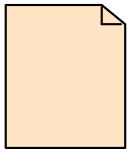
`render : Producer →` 

Producers

`render : Producer →` 

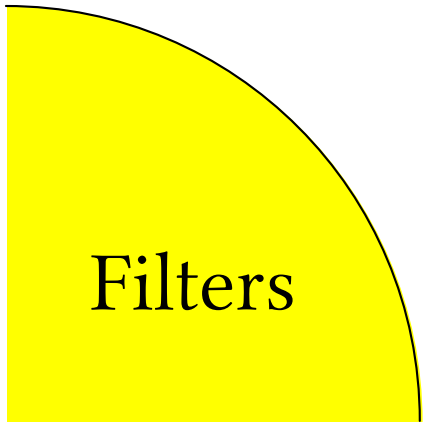
`clip : String → Producer`

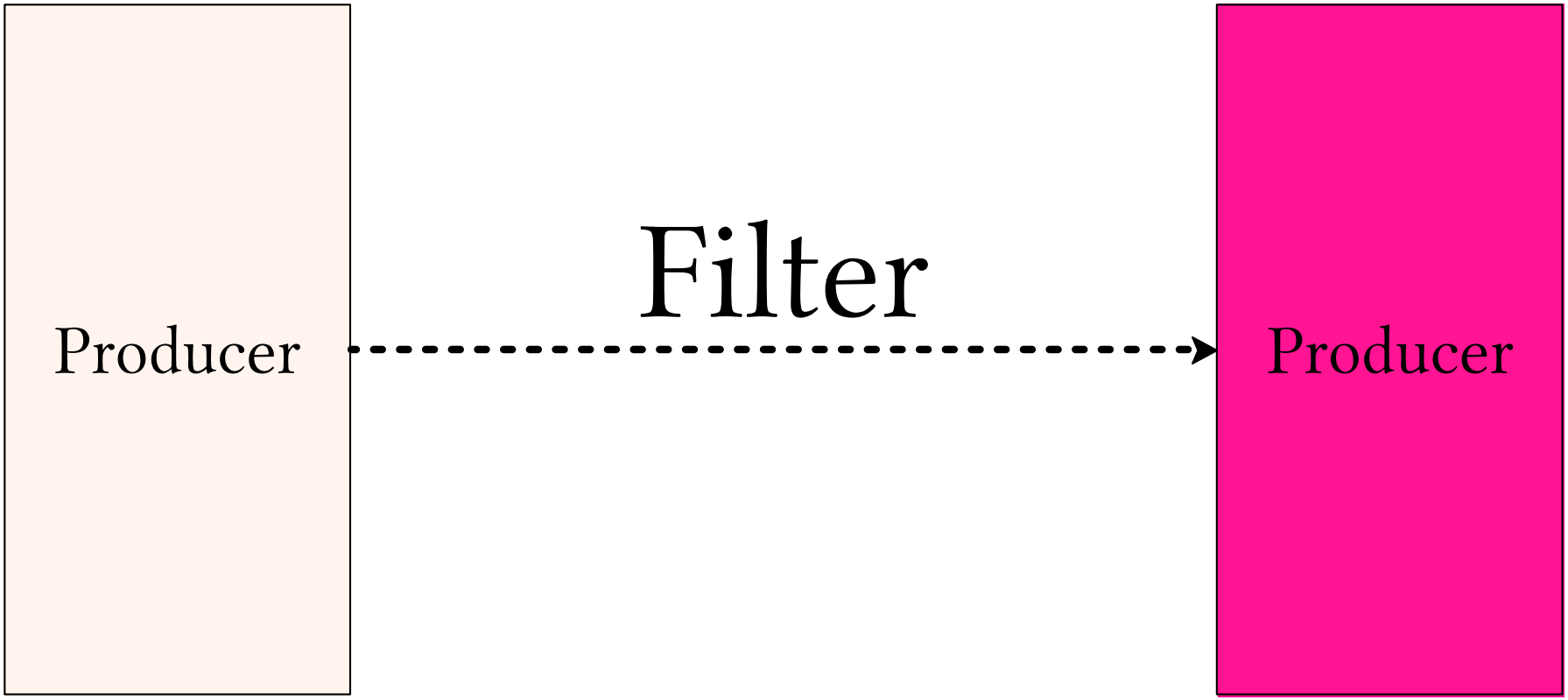
Producers

`render : Producer →` 

`clip : String → Producer`

```
(render (clip "demo.mp4")) ⇒
```

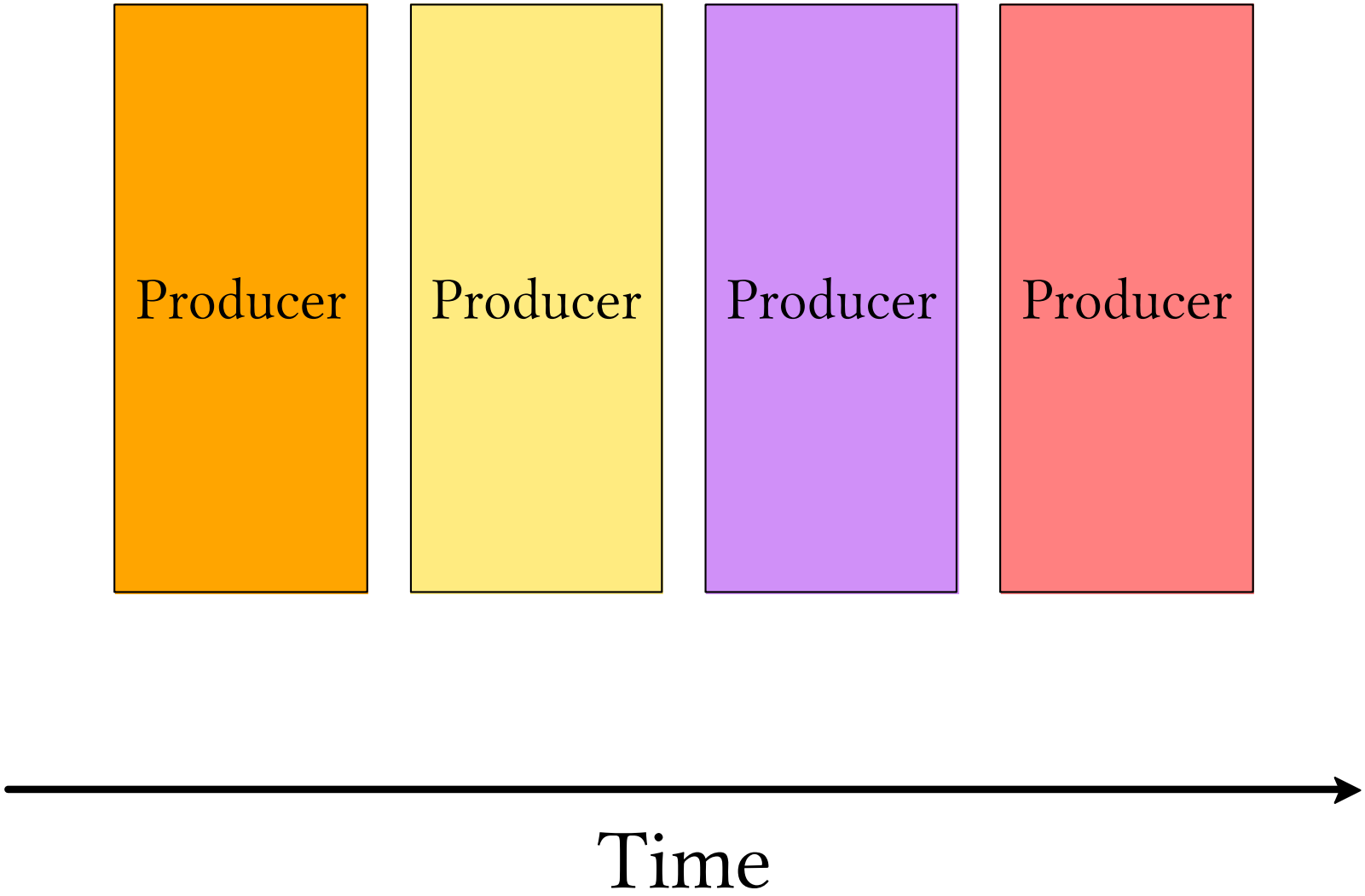

Filters


Producer
Filter
Producer

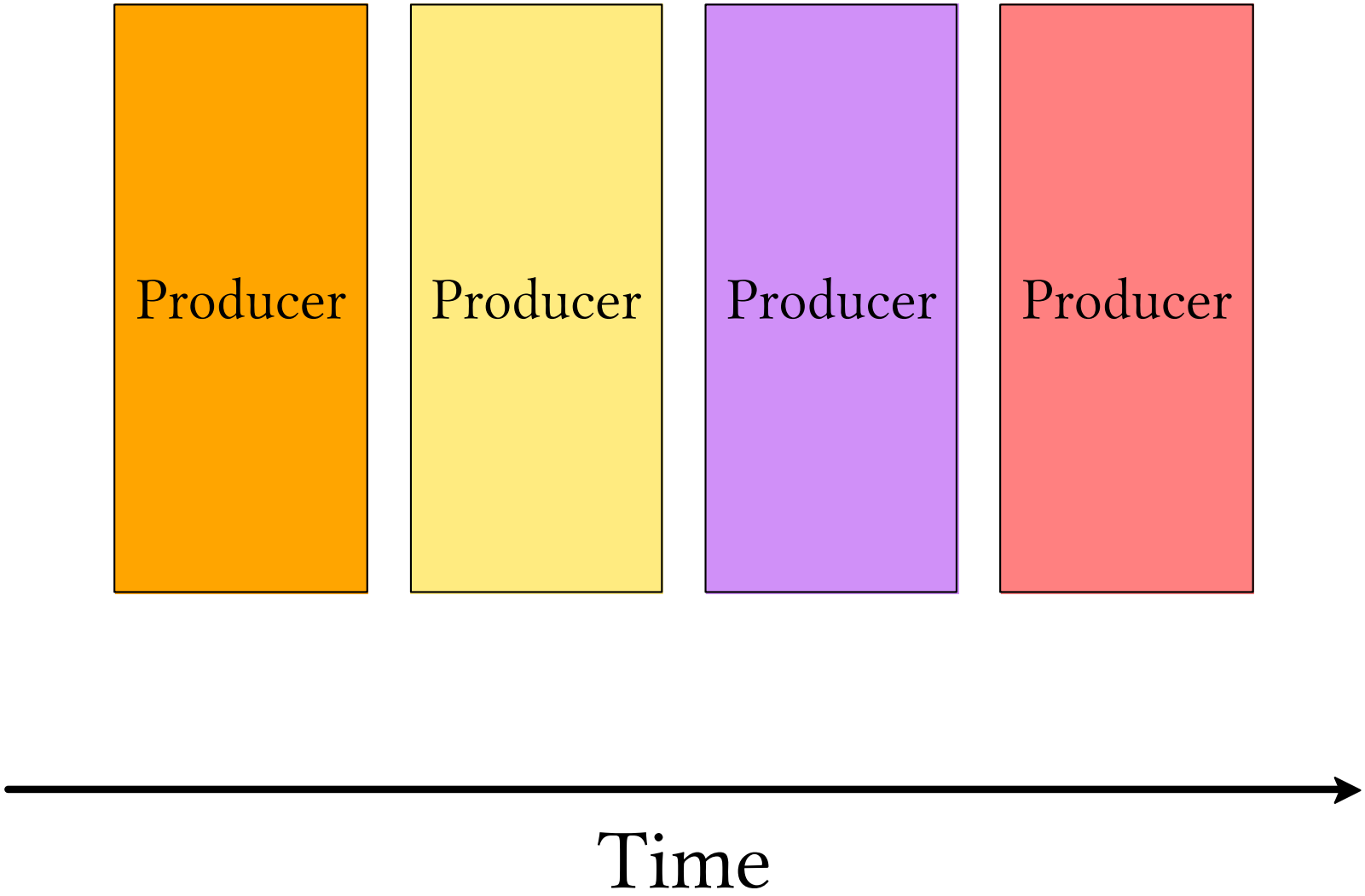
```
(attach-filter bunny-clip (sepia-filter))
```

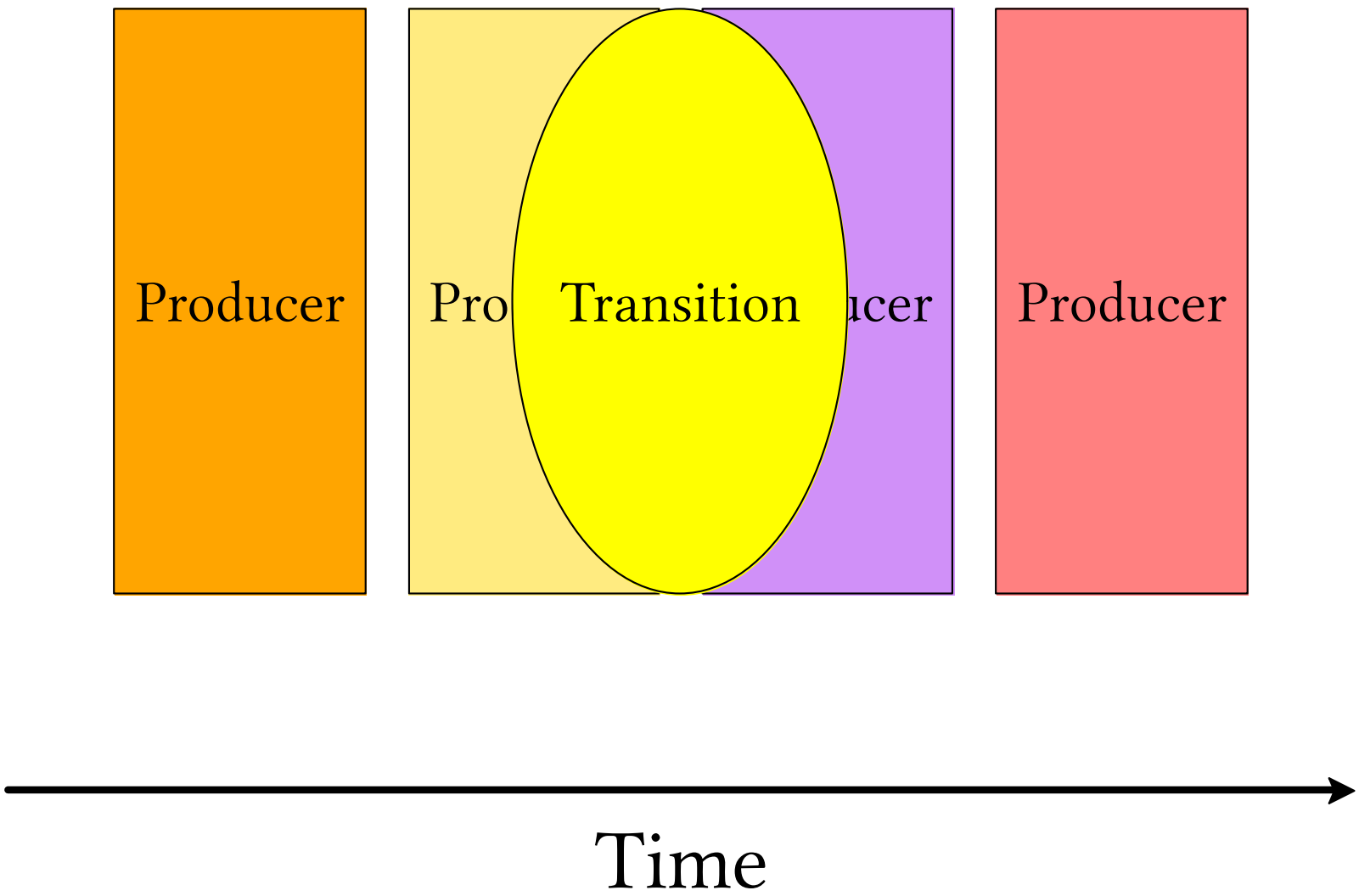
```
(attach-filter bunny-clip (sepia-filter))
```


Playlists


Producer
Producer
Producer
Producer
Time

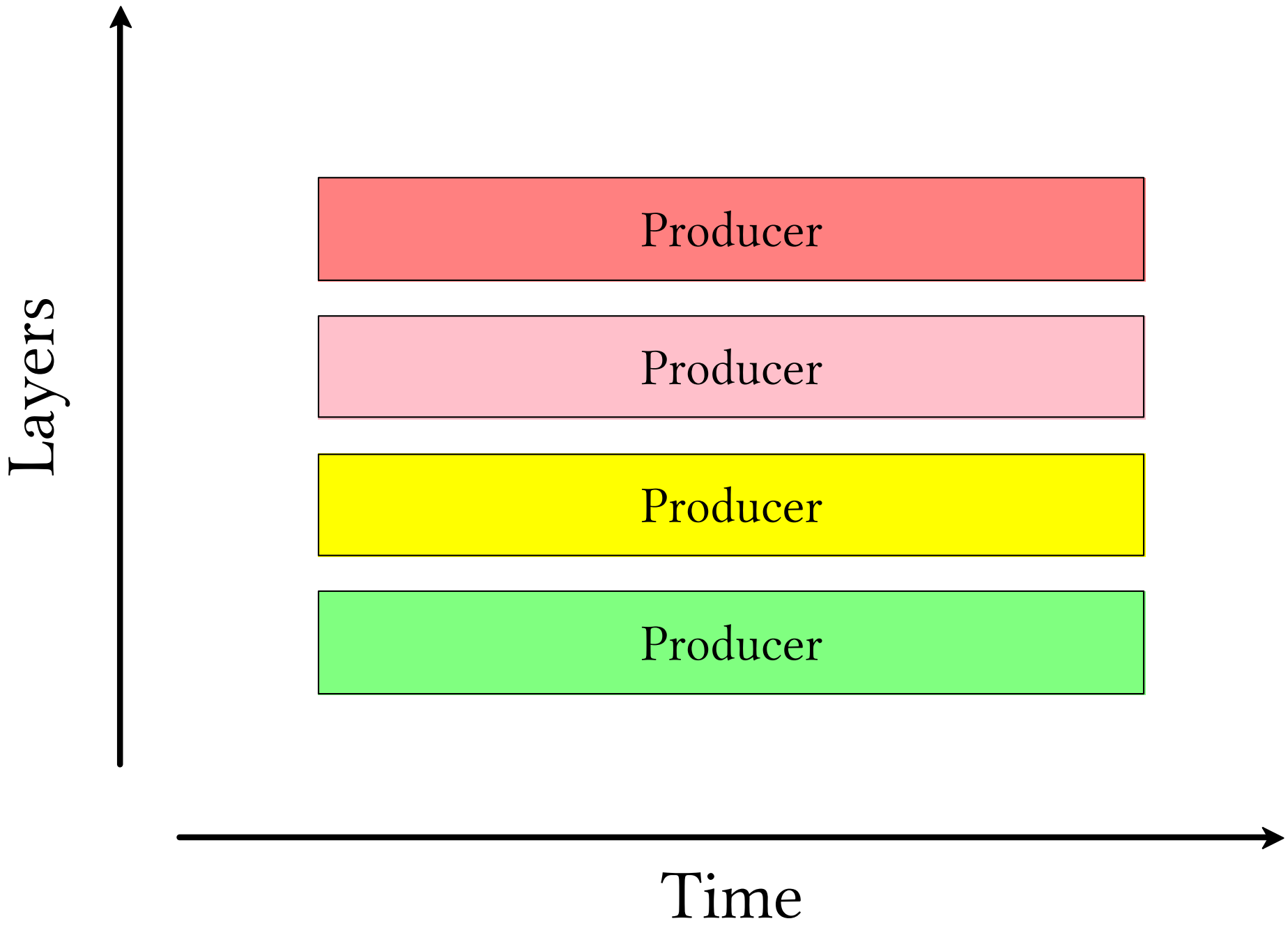
```
(playlist (clip "jumping.mp4")
          (clip "flying.mp4"))
```

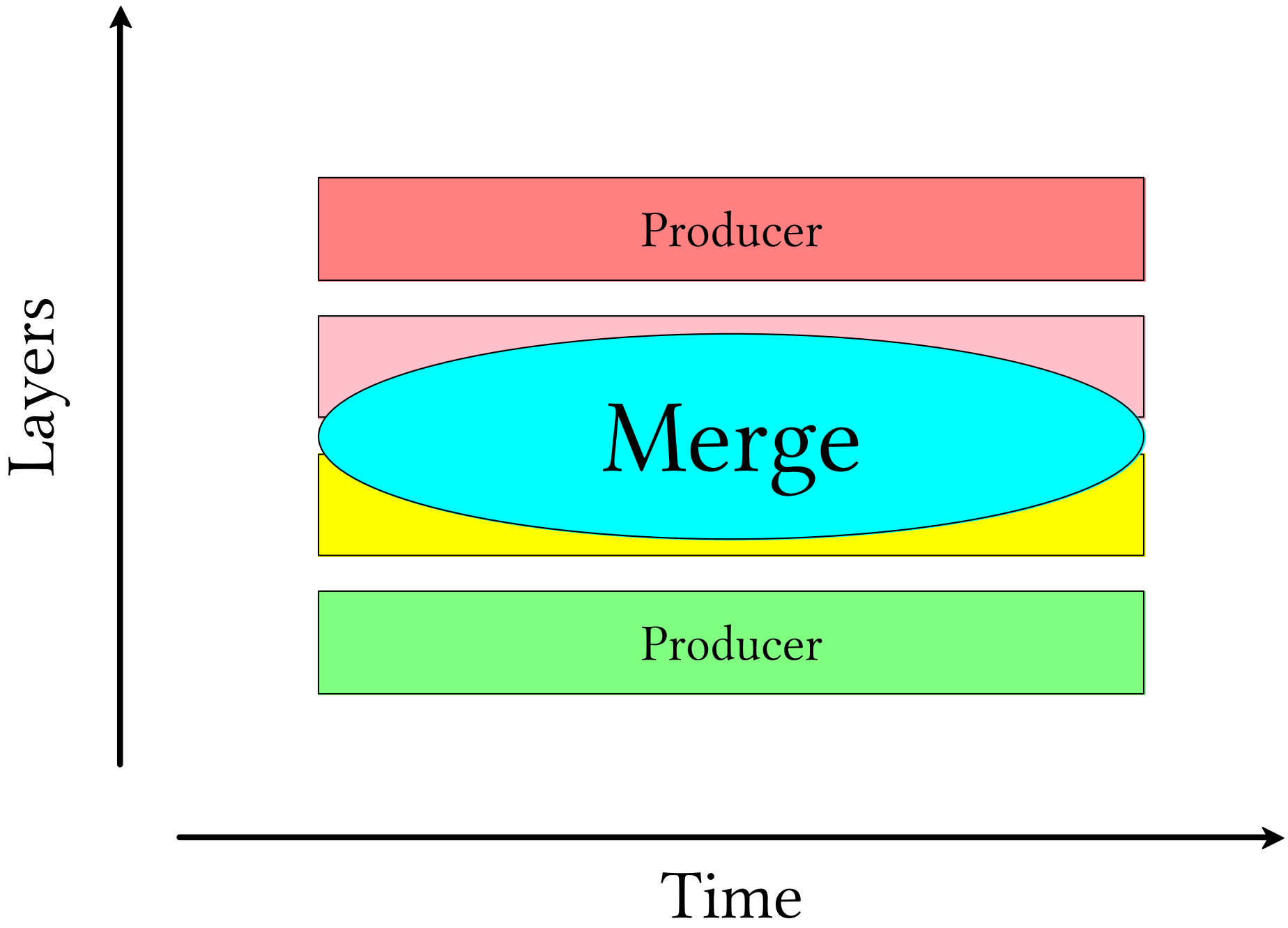

Producer
Producer
Producer
Producer
Time


Producer
Pro
Transition
ucer
Producer
Time

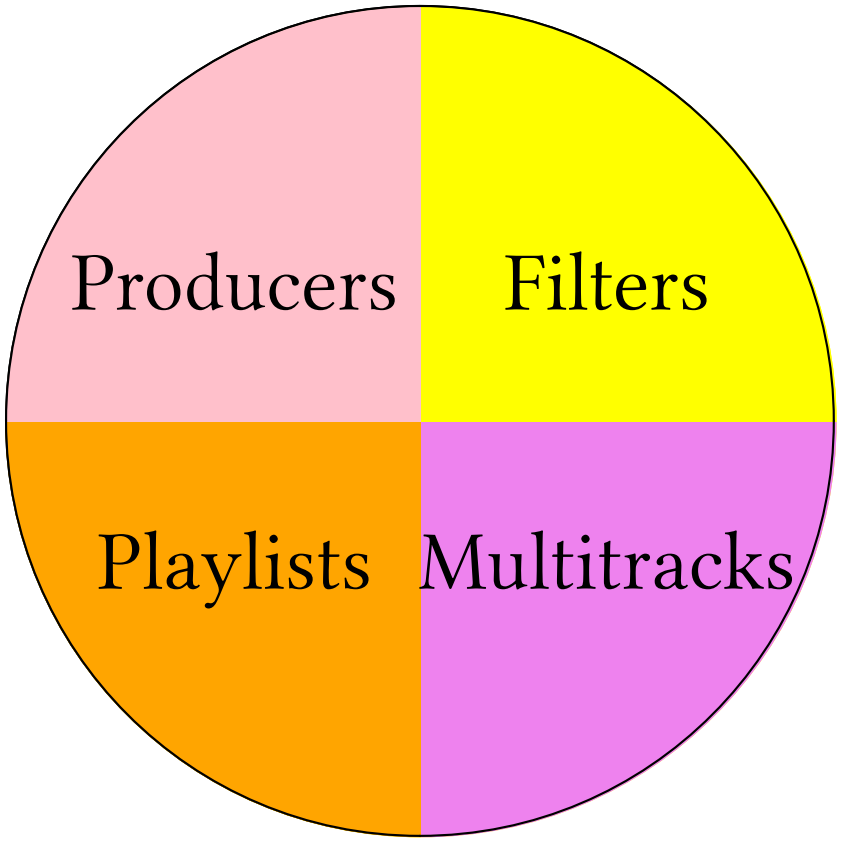
```
(playlist (clip "jumping.mp4")
          (fade-transition 1)
          (clip "flying.mp4"))
```

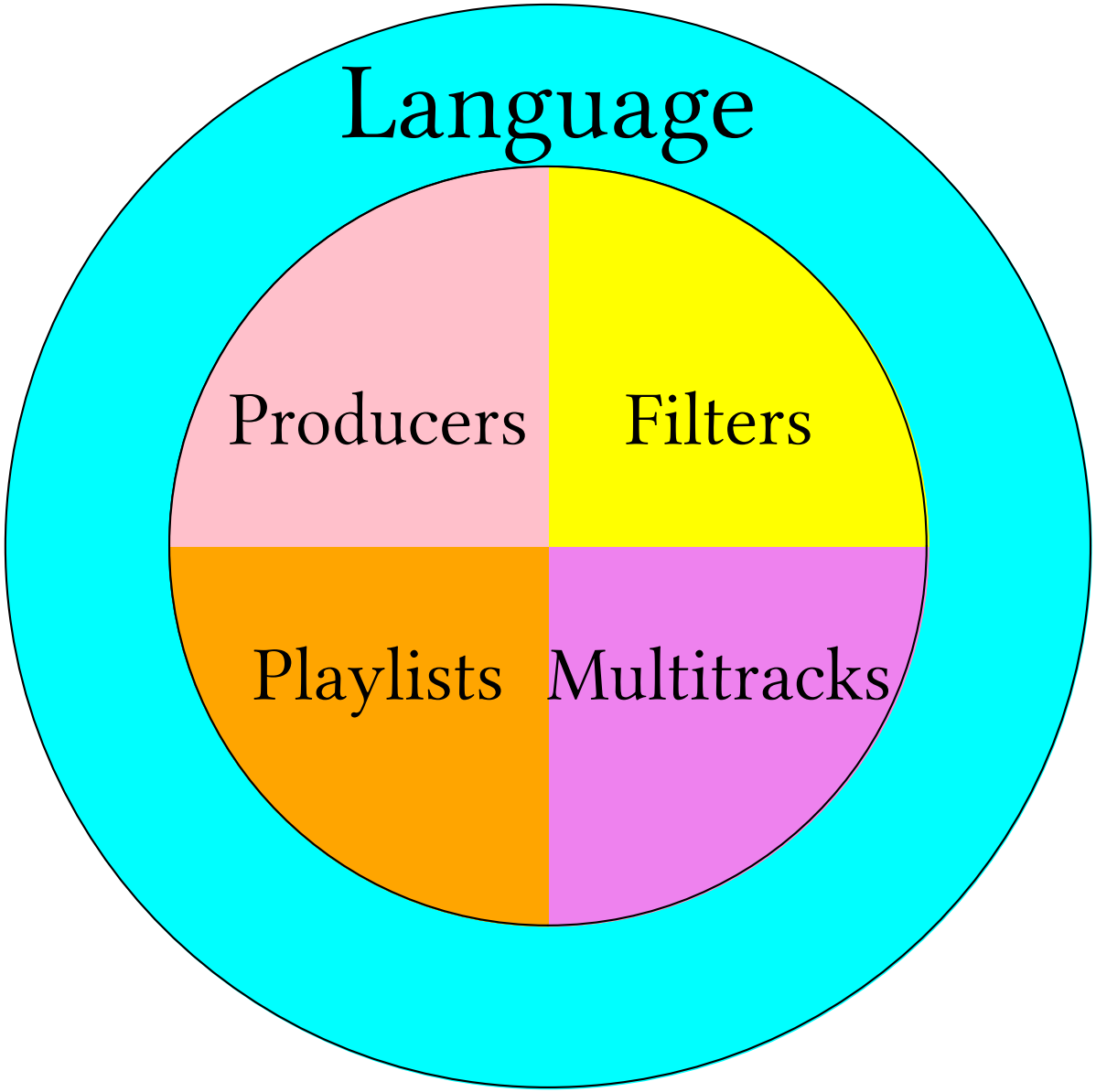
Multitracks


Layers
Producer
Producer
Producer
Producer
Time


Layers
Producer
Merge
Producer
Time

```
(define WIDTH 1920)
(define HEIGHT 1080)
(playlist (color "black")
          (overlay-merge 0 0 (/ WIDTH 2) HEIGHT)
          (clip "running.mp4")
          (overlay-merge (/ WIDTH 2) 0 (/ WIDTH 2) HEIGHT)
          (clip "flying.mp4"))
```


Producers
Filters
Playlists
Multitracks


Language
Producers
Filters
Playlists
Multitracks

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "branded.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "branded.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

Primitives

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "branded.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

List Comprehensions

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "branded.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

Modules

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "branded.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

branded.vid

```
#lang video/lib
;; Generate a branded video
(define-video (branded logo vid)
  logo
  (fade-transition 1)
  (multitrack logo
              (overlay 0 0 100 100)
              vid))
```

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (video "branded.vid"
      (logo.png")
        (clip (format "~aX~a.mp4" i j)))))
```

Functions

branded.vid

```
#lang video/lib
;; Generate a branded video
(define-video (branded logo vid)
  logo
  (fade-transition 1)
  (multitrack logo
              (overlay 0 0 100 100)
              vid))
```

mosaic.vid

```
#lang video
;; Create a mosaic of four videos
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "branded.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

branded.vid

```
#lang video/lib
;; Generate a branded video
(define-video (branded logo vid)
  logo
  (fade-transition 1)
  (multitrack logo
              (overlay 0 0 100 100)
              vid))
```

Movies as Programs: The Story of a Racket

We make DSLs using

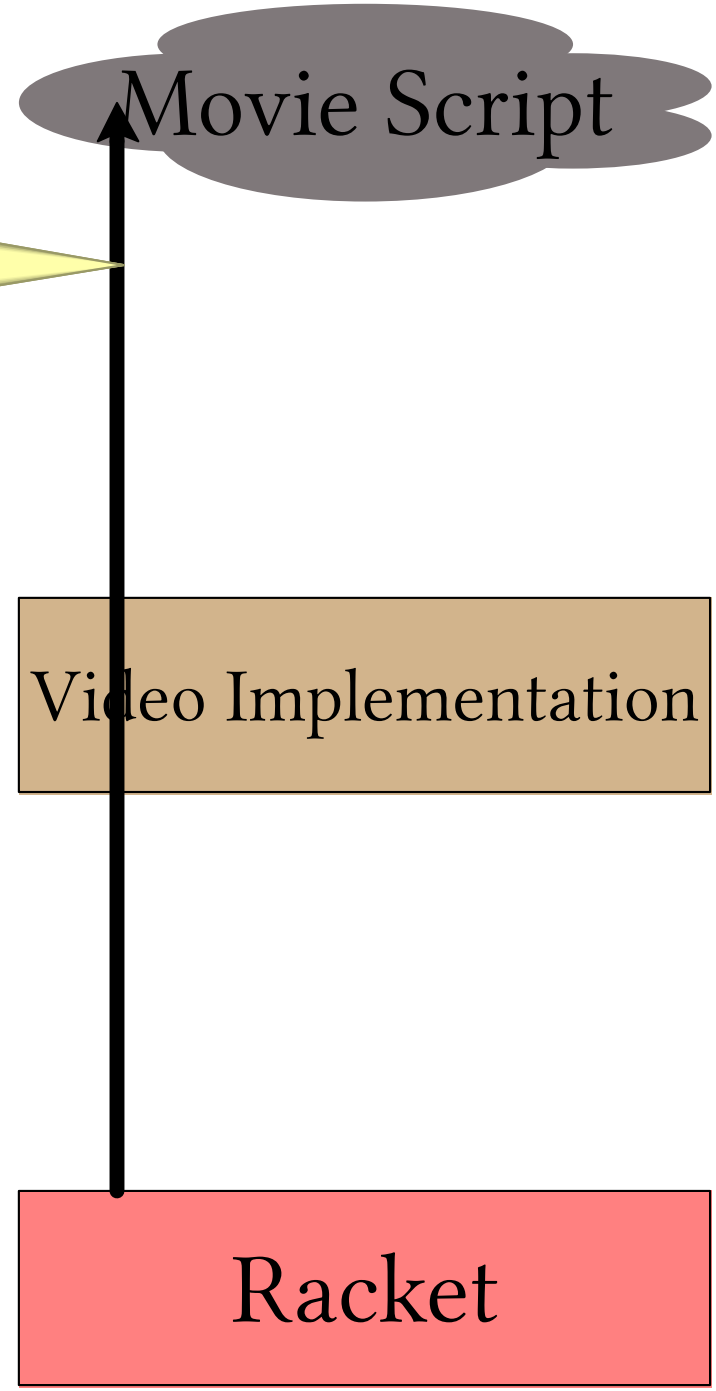
Linguistic Inheritance

We make DSLs using

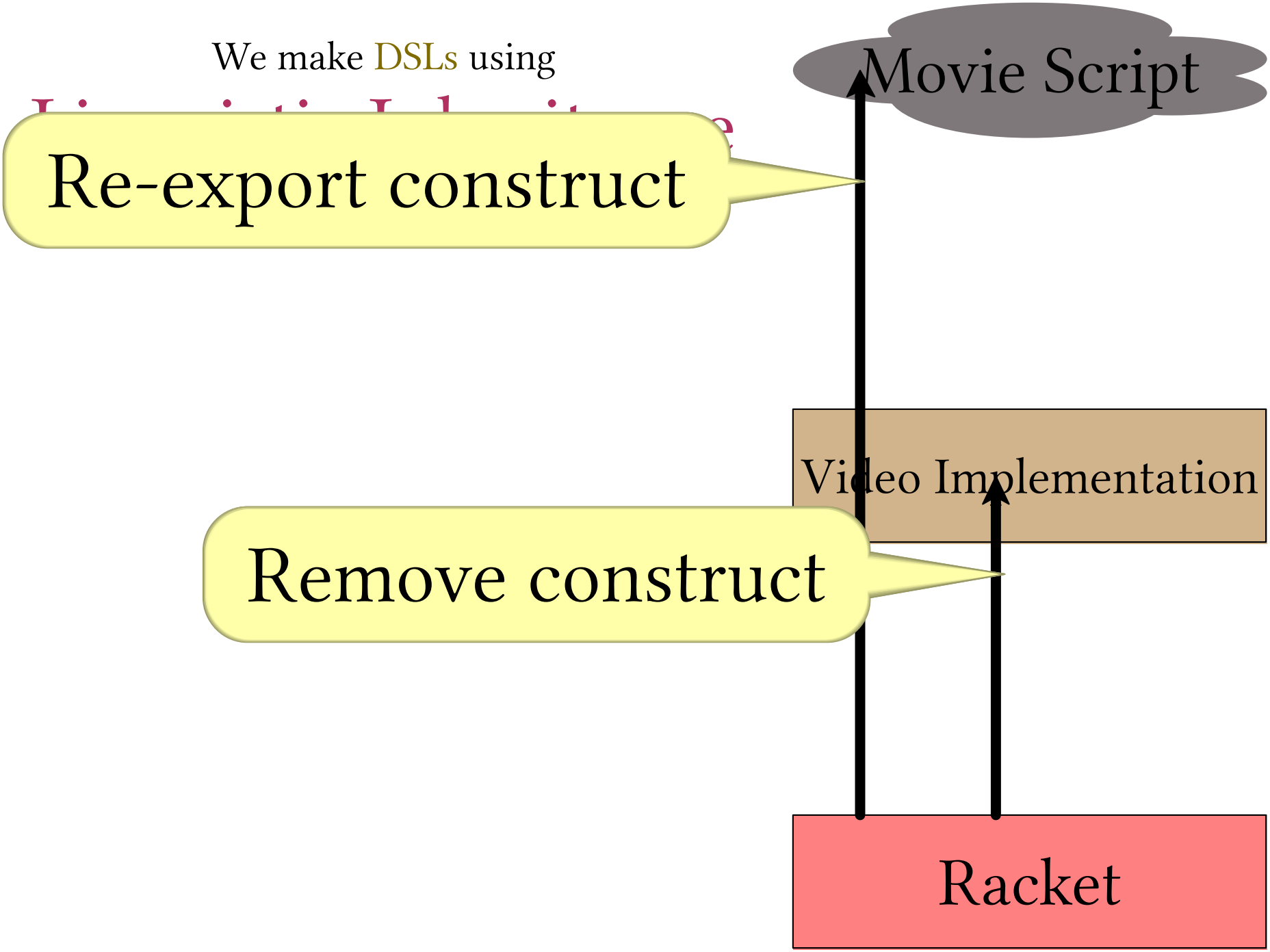
Linguistic Inheritance

We make DSLs using

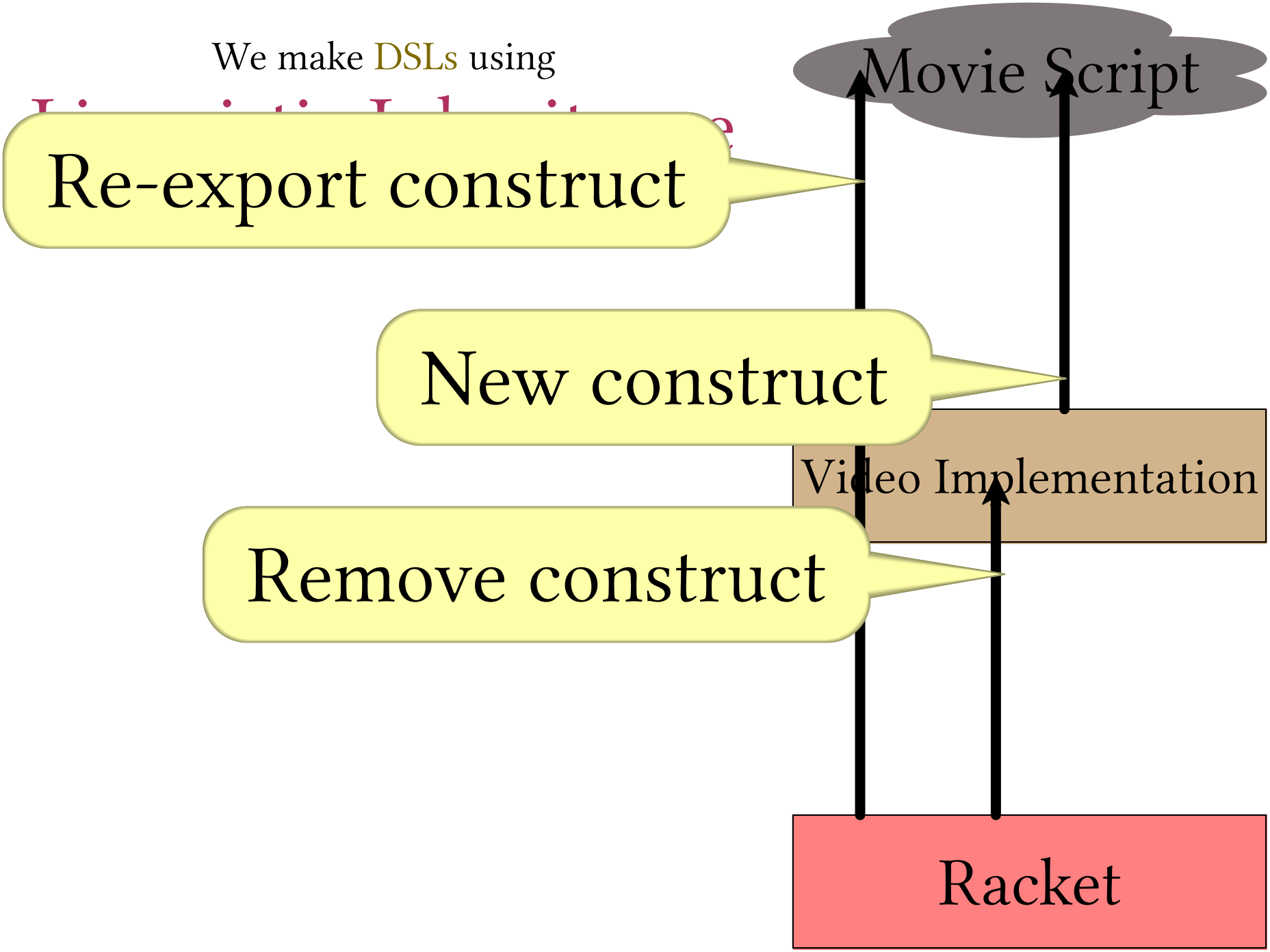
Re-export construct



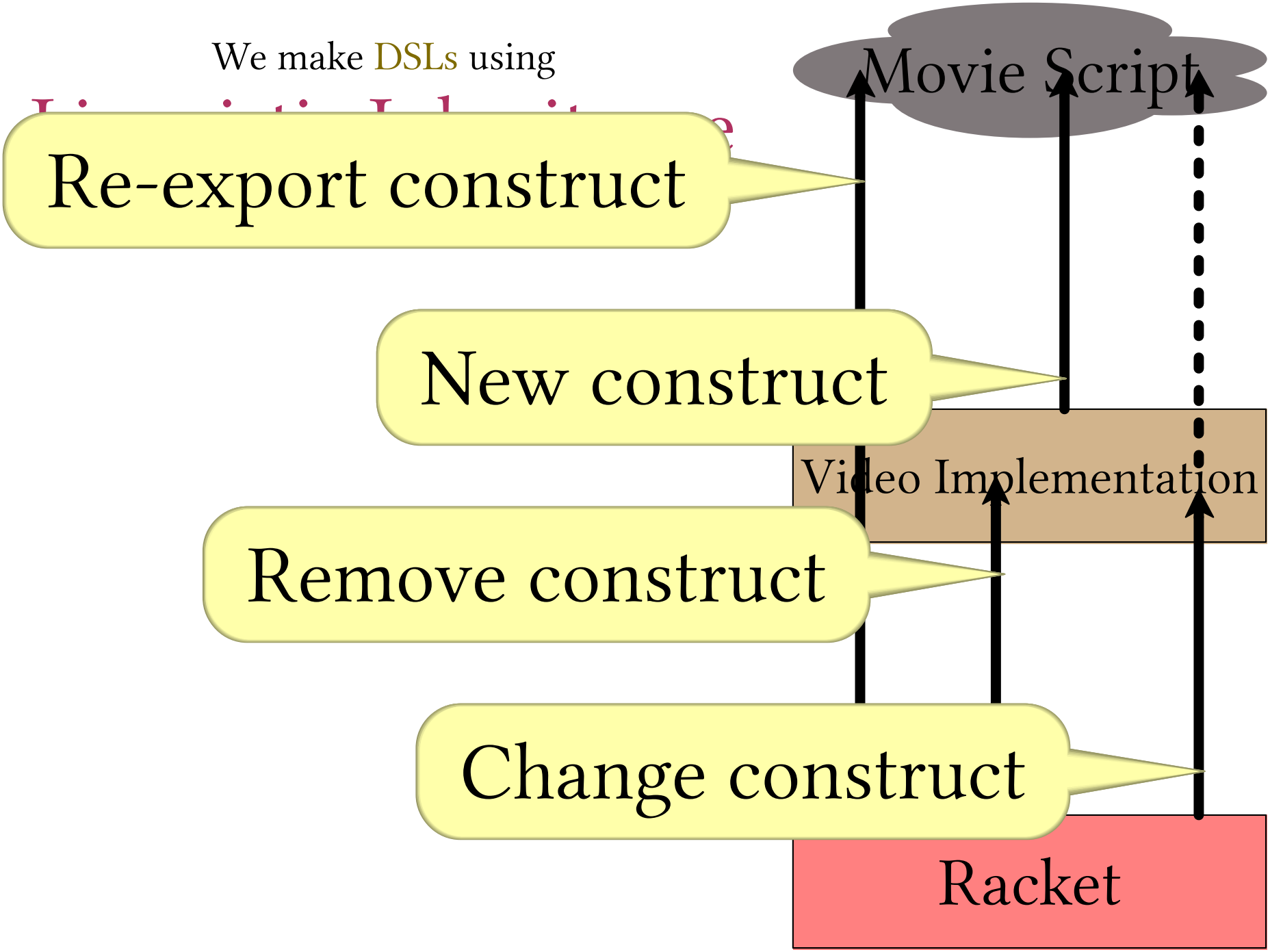
We make DSLs using



We make DSLs using



We make DSLs using



Change construct

Interposition Points

```
#lang video

logo
talk

;; Where
(define logo
  ...)
(define talk
  ...)
```

parses →

```
(module anon video
  (#%module-begin
   logo
   talk
   (define logo
     ...)
   (define talk
     ...)))
```

Interposition Points

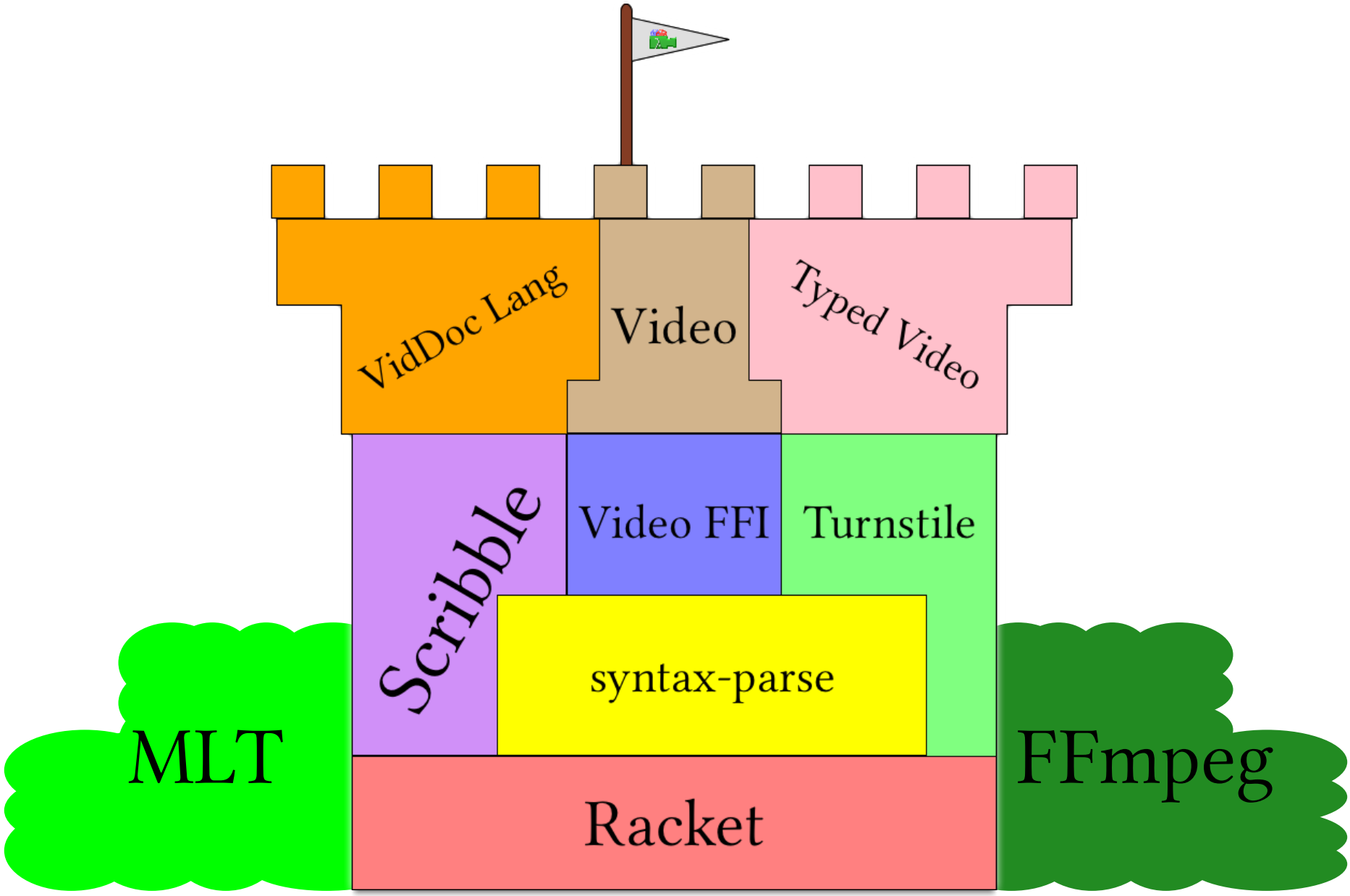
```
(module anon video
  (#%module-begin
   logo
   talk
   (define logo
     ...)
   (define talk
     ...)))
```

elaborates →

```
(module anon racket
  (#%module-begin
   (require vidlib)
   (define logo
     ...)
   (define talk
     ...)
   (vid-begin vid
    logo
    talk)))
```

Implementing Interposition Points

```
#lang racket
(provide (rename-out [video-module-begin
                      #%module-begin]))
(define-syntax (video-module-begin stx)
    ... #%module-begin ...)
```


VidDoc Lang
Video
Typed Video
Scribble
Video FFI
Turnstile
syntax-parse
MLT
Racket
FFmpeg

```
(clip "clip.mp4"
      #:start 0
      #:end 50)
```

```
(cut-producer (clip "clip.mp4"
                    #:start 0
                    #:end 50)
              #:start 0
              #:end 100)
```

```
(cut-produc    (cl    "clip.mp4"
                      #:start 0
                      #:end 50)
                      t 0
            #:e       100)
```

A Typed DSL

$$\frac{\texttt{m >= n}}{\texttt{(Producer m) <: (Producer n)}}$$

A Typed DSL

CLIP

$$\frac{\Gamma \vdash \texttt{f} : \texttt{File} \qquad |\texttt{f}| = \texttt{n}}{\Gamma \vdash (\texttt{clip f}) : (\texttt{Producer n})}$$

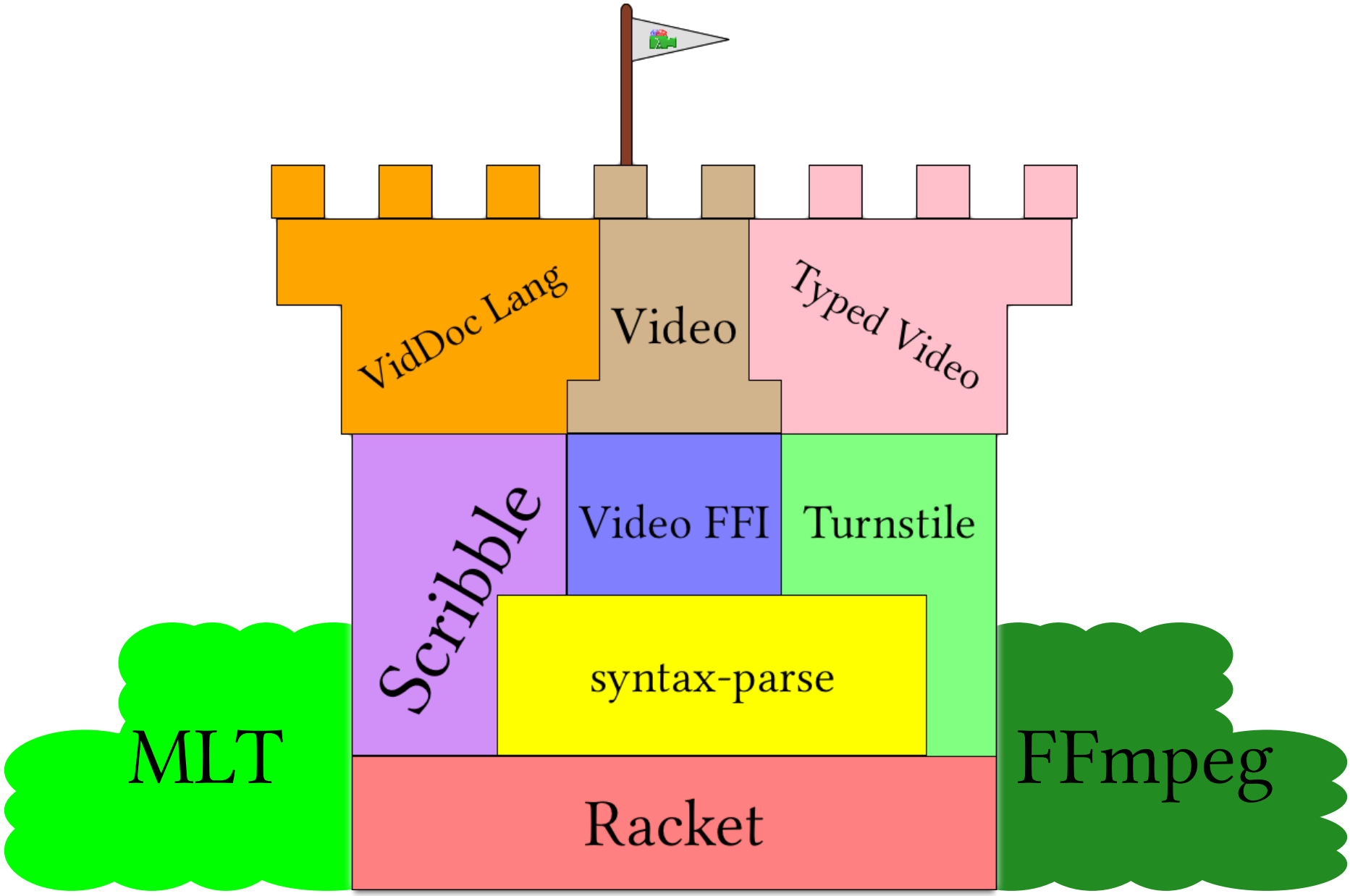
(POPL, 2016)

A Type Implementation DSL

$$\textsc{Clip}\quad \frac{\Gamma \vdash \texttt{f} : \texttt{File} \qquad |\texttt{f}| = \texttt{n}}{\Gamma \vdash \texttt{(clip f)} : \texttt{(Producer n)}}$$

```
(define-typed-syntax (clip f) ≫
  [⊢ f ≫ _ ⇐ File] #:where n (length f)
  ------------------------------------
  [⊢ (untyped:clip f) ⇒ (Producer n)])
```

(POPL, 2016)

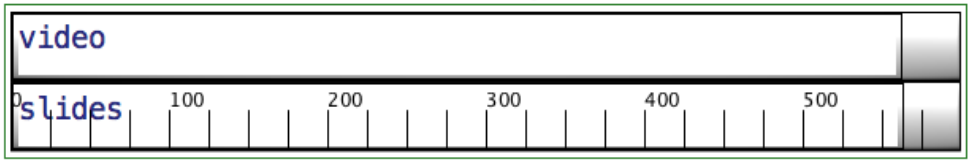

VidDoc Lang
Video
Typed Video
Scribble
Video FFI
Turnstile
syntax-parse
MLT
Racket
FFmpeg

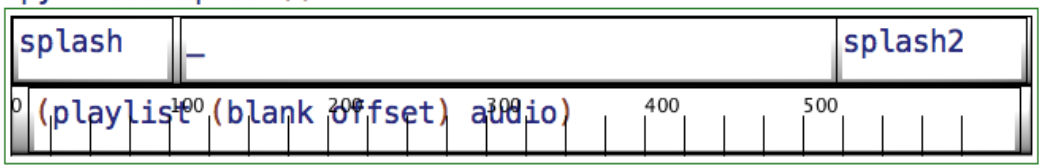
The Future...

conference-lib.vid - DrRacket

conference-lib.vid (define ...) Preview Video Check Syntax Debug Macro Stepper Multi-File Coverage Run Stop

```
1  #lang video
2
3  (provide conference-talk)
4
5  (define (conference-talk video slides audio offset)
6    (attach-transition raw-video
7                       (fade-transition #:length 50 #:in splash #:out _)
8                       (fade-transition #:length 50 #:in _ #:out splash2))
9
   (define* _                                                          )
10 (define* _ (attach-transition _ (composite-transition 0 0 1/4 1/4
11                                                       #:top video
12                                                       #:bottom slides)))
13 (define splash (image "splash.png"))
14 (define splash2 (copy-video splash))
15
   (define raw-video                                                   ))
16
17
```

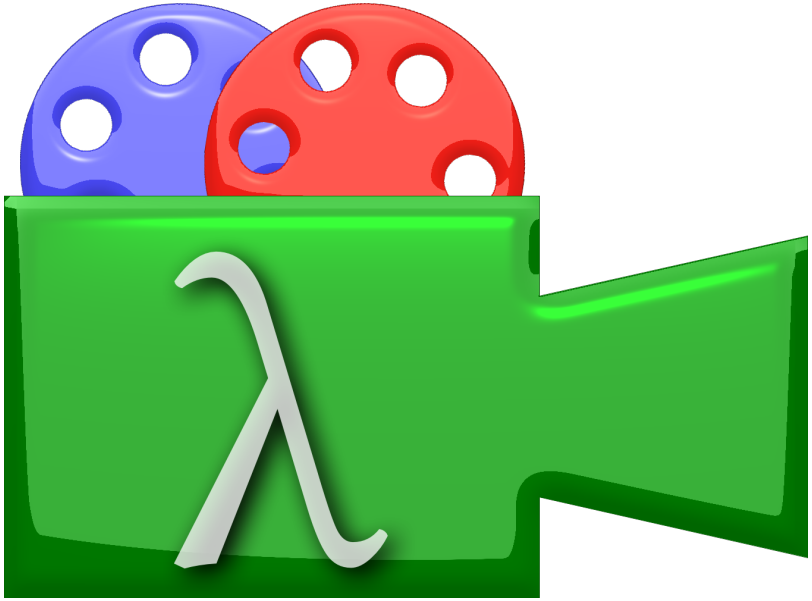




Determine language from source 17:0 375.01 MB

Thanks For Watching

http://lang.video
@videolang



We make DSLs using

Linguistic Inheritance