Super 8 Languages for Making Movies
(A Functional Pearl)

Leif Andersen

Stephen Chang

Matthias Felleisen

PLT @ Northeastern University

ICFP - Sept 4, 2017

Super 8 : Languages for Making Movies

Super , 8 Languages for Making Movies

~~Super 8~~ : ~~Languages for Making Movies~~

A DSL for Scripting Videos

~~Super~~ , ~~8 Languages for Making Movies~~

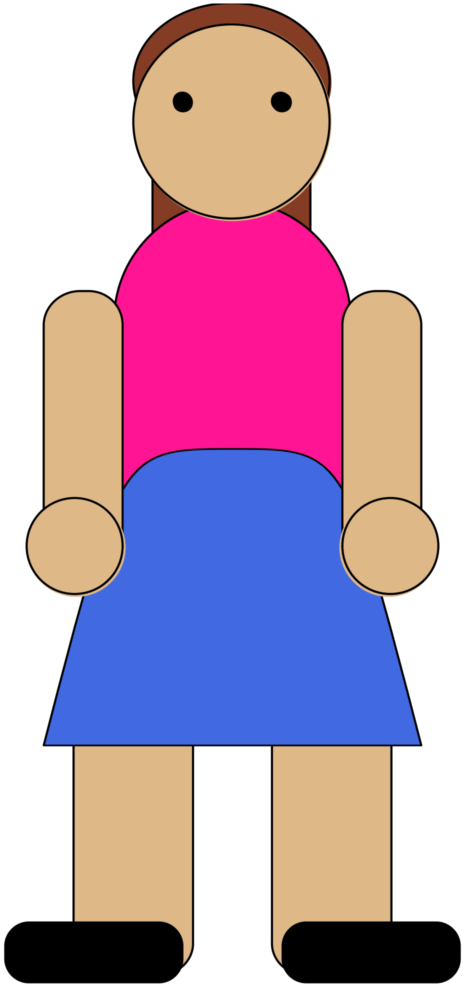
DSL Towers to Solve Multitudes of Problems

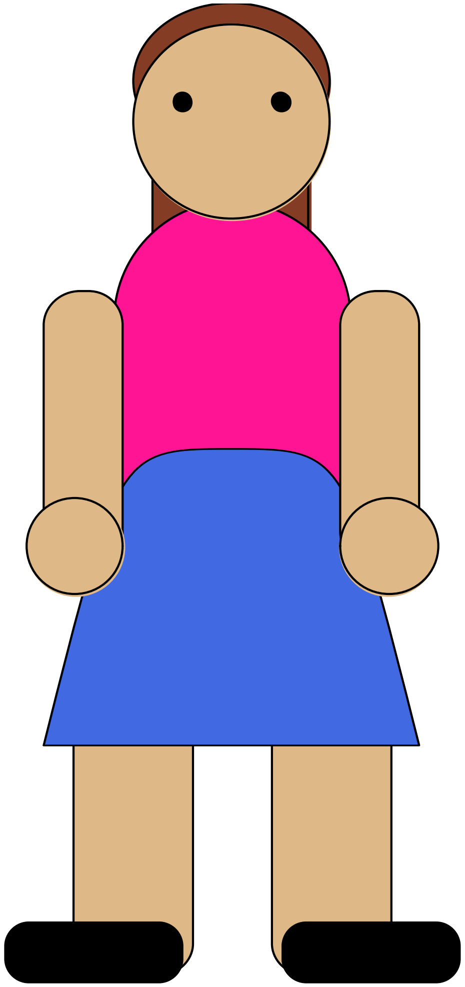
~~Super 8~~: ~~Languages for Making Movies~~

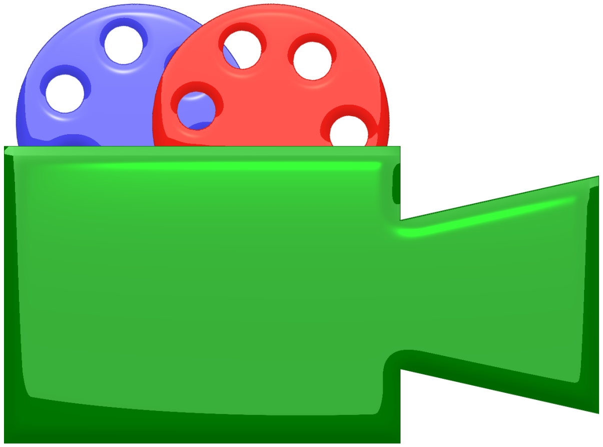
A DSL for Scripting Videos

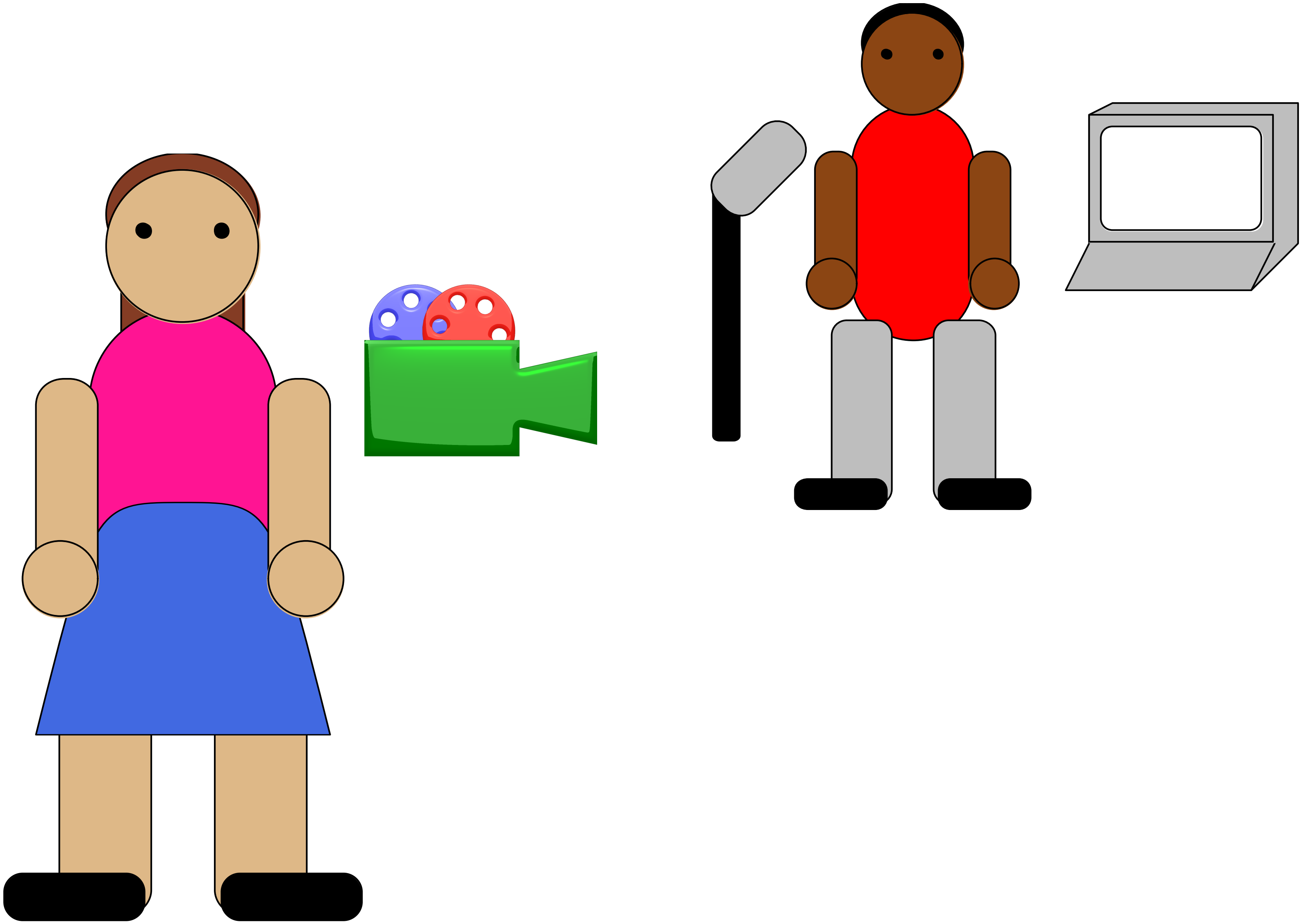
~~Super~~, ~~8 Languages for Making Movies~~

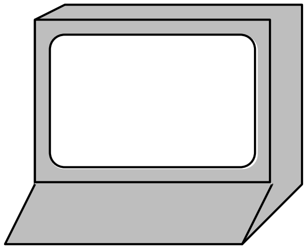
DSL Towers to Solve Multitudes of Problems

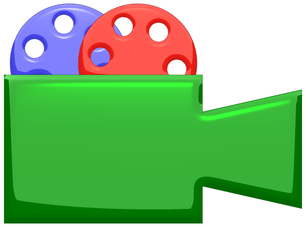


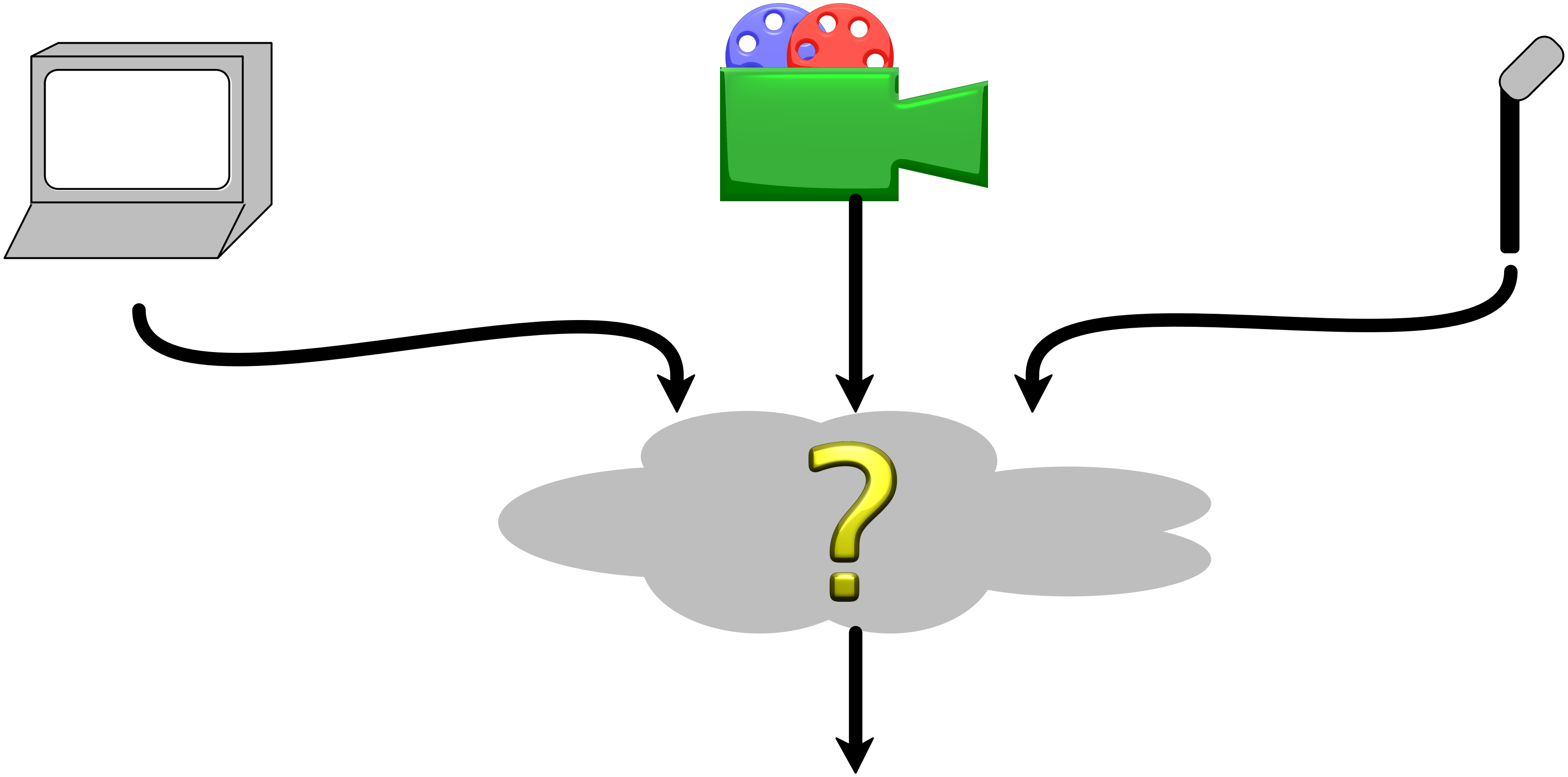












One down

One down
19 more to go…

We Need Automation

We Need ~~Automation~~ ABSTRACTION

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	

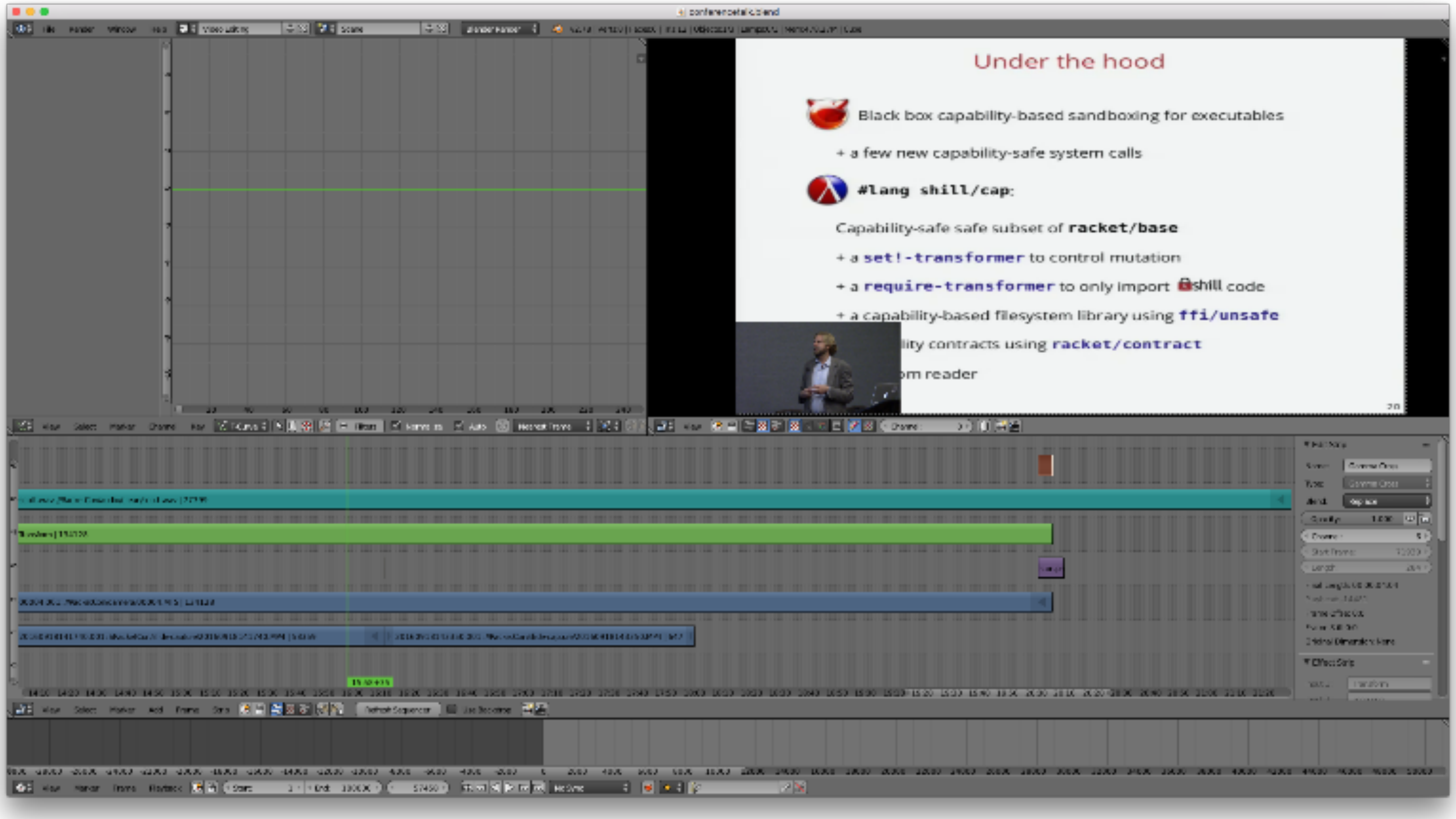
The Landscape

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	😬

Video Editor



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**

lity contracts using **racket/contract**

om reader

Functional Programming Language*

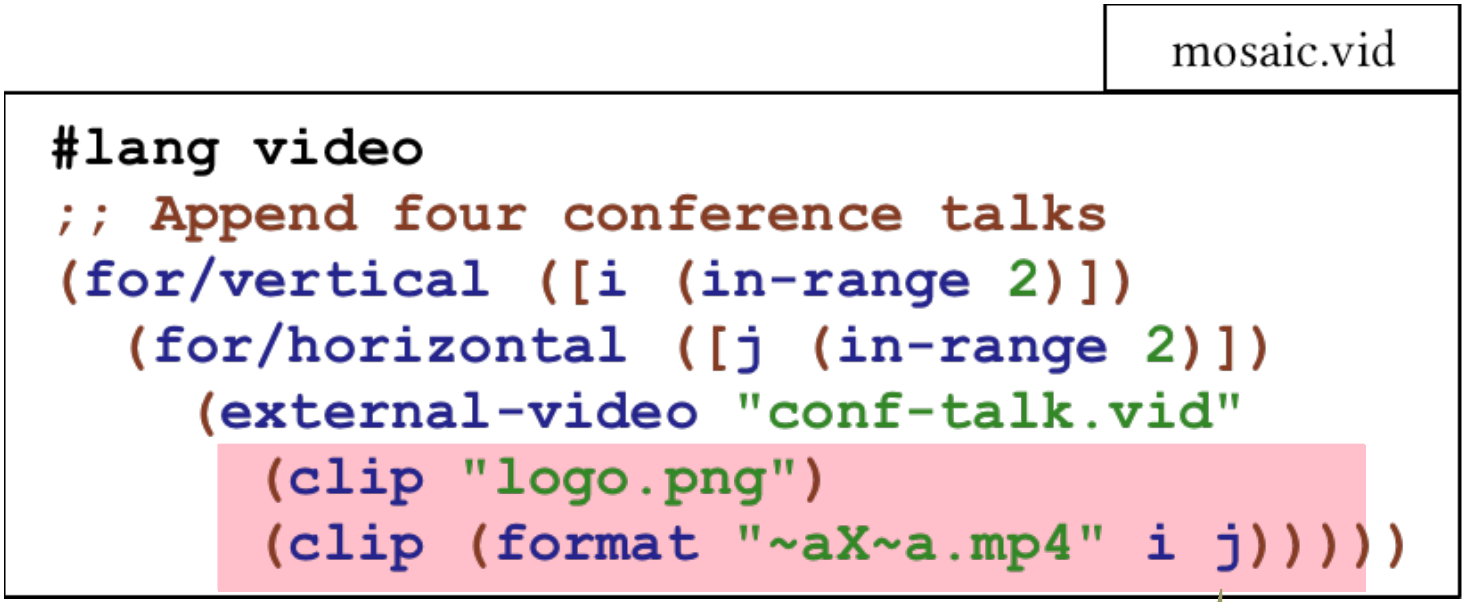
*But bad with abstractions.

Video,

the programming language

mosaic.vid

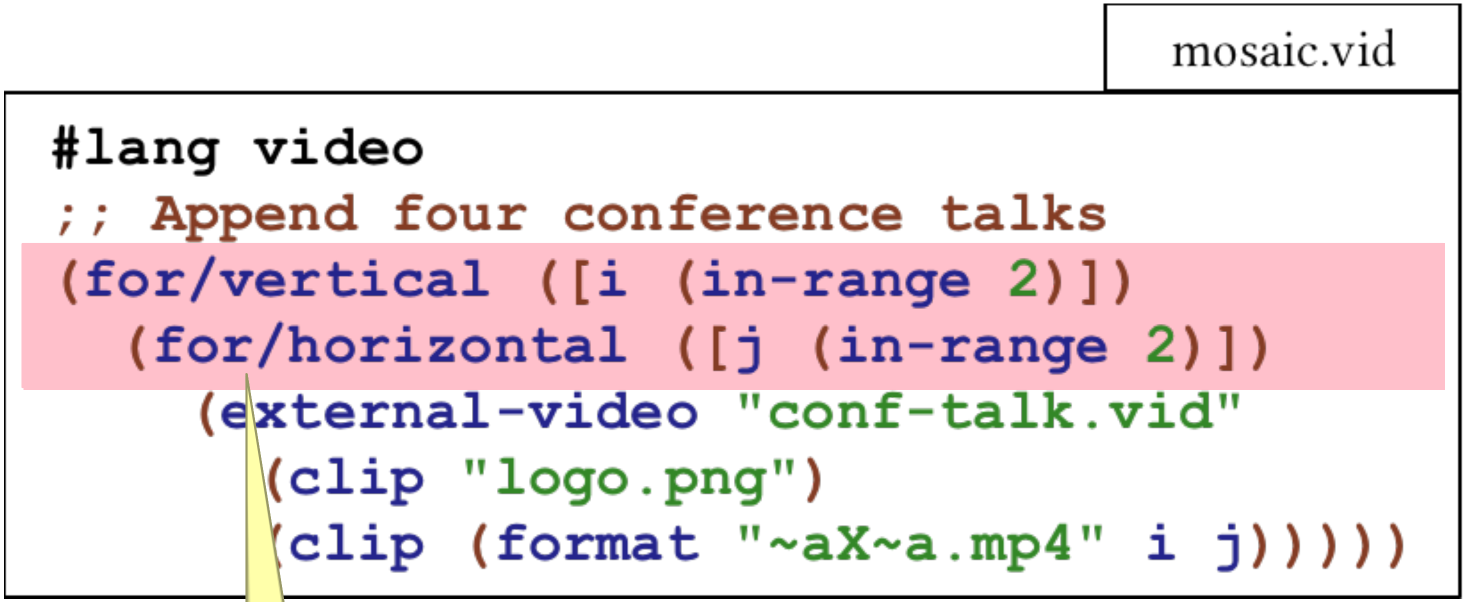
```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "conf-talk.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j))))))
```



mosaic.vid

```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "conf-talk.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

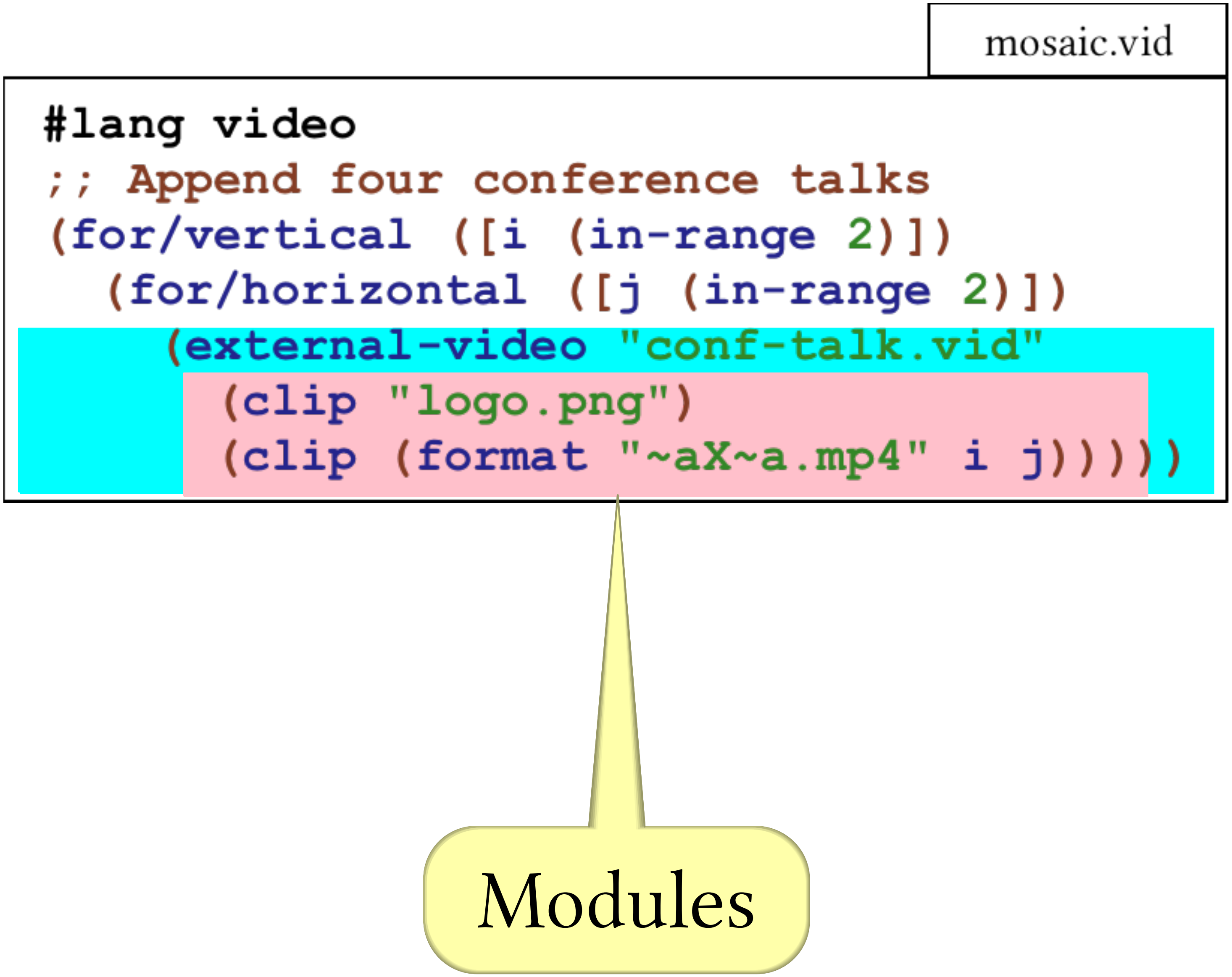
Primitives



mosaic.vid

```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "conf-talk.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j))))))
```

List Comprehensions



mosaic.vid

```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "conf-talk.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j)))))
```

Modules

mosaic.vid

```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "conf-talk.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j))))))
```

conf-talk.vid

```
#lang video/lib
;; Generate a conference talk
(define-video (conf-talk logo slides)
  logo
  (fade-transition 1)
  (multitrack logo
              (overlay 0 0 100 100)
              slides))
```

mosaic.vid

```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
                 video "conf-talk.vid"
                 go.png")
        (clip (  rmat "~aX~a.mp4" i j)))))
```

Functions

conf-talk.vid

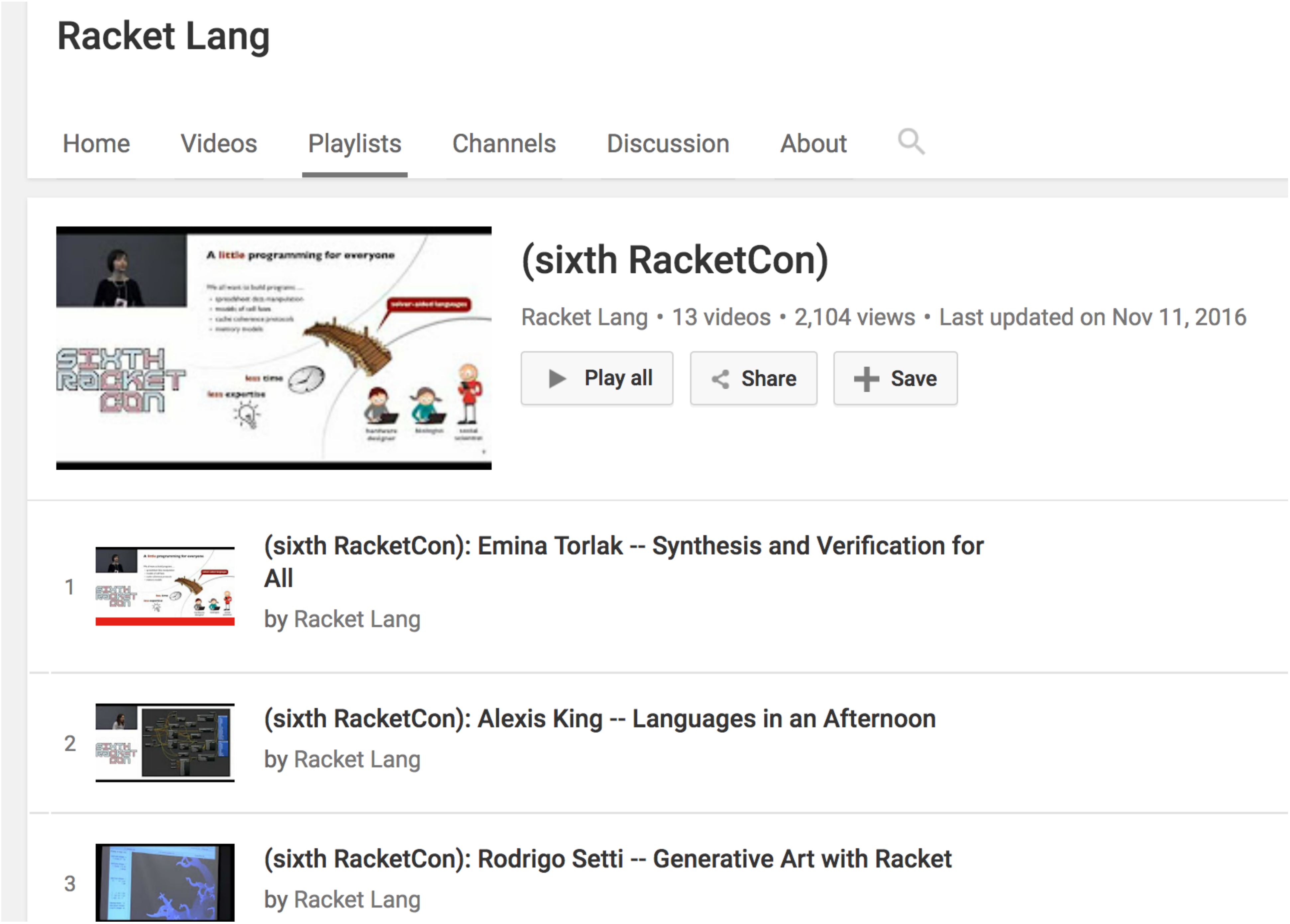
```
#lang video/lib
;; Generate a conference talk
(define-video (conf-talk logo slides)
  logo
  (fade-transition 1)
  (multitrack logo
              (overlay 0 0 100 100)
              slides))
```

mosaic.vid

```
#lang video
;; Append four conference talks
(for/vertical ([i (in-range 2)])
  (for/horizontal ([j (in-range 2)])
    (external-video "conf-talk.vid"
      (clip "logo.png")
      (clip (format "~aX~a.mp4" i j))))))
```

conf-talk.vid

```
#lang video/lib
;; Generate a conference talk
(define-video (conf-talk logo slides)
  logo
  (fade-transition 1)
  (multitrack logo
              (overlay 0 0 100 100)
              slides))
```



Racket Lang

(sixth RacketCon)

Racket Lang • 13 videos • 2,104 views • Last updated on Nov 11, 2016

1 (sixth RacketCon): Emina Torlak -- Synthesis and Verification for All

by Racket Lang

2 (sixth RacketCon): Alexis King -- Languages in an Afternoon

by Racket Lang

3 (sixth RacketCon): Rodrigo Setti -- Generative Art with Racket

by Racket Lang

⏱

Writing Video
+ Editing Talks

(RacketCon 2016)

<

⏱

Editing Talks
Manually

(RacketCon 2015)

~~Super 8 : Languages for Making Movies~~

A DSL for Scripting Videos

~~Super , 8 Languages for Making Movies~~

DSL Towers to Solve Multitudes of Problems

~~Super 8: Languages for Making Movies~~

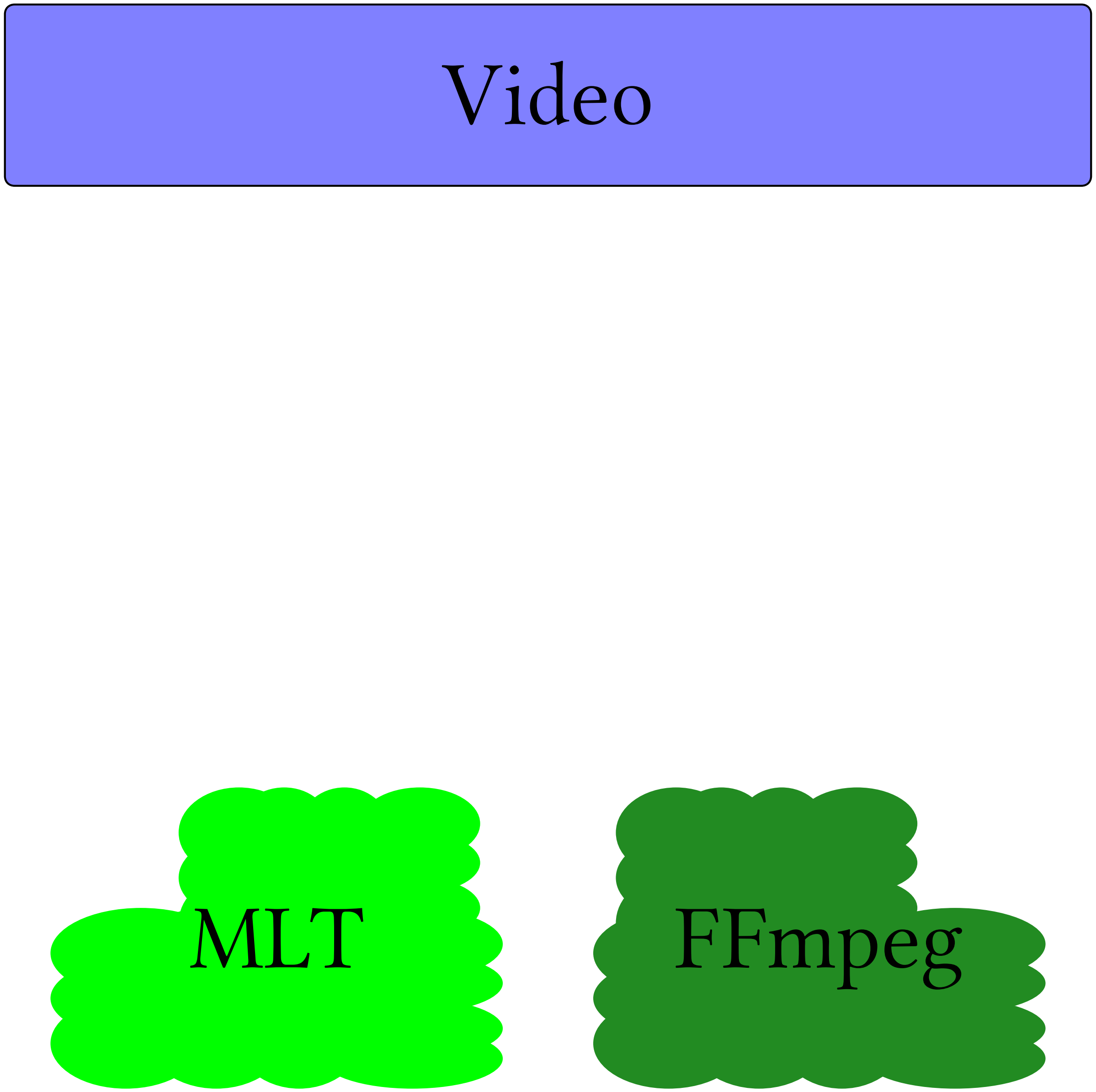
A DSL for Scripting Videos

~~Super, 8 Languages for Making Movies~~

DSL Towers to Solve Multitudes of Problems

Video,

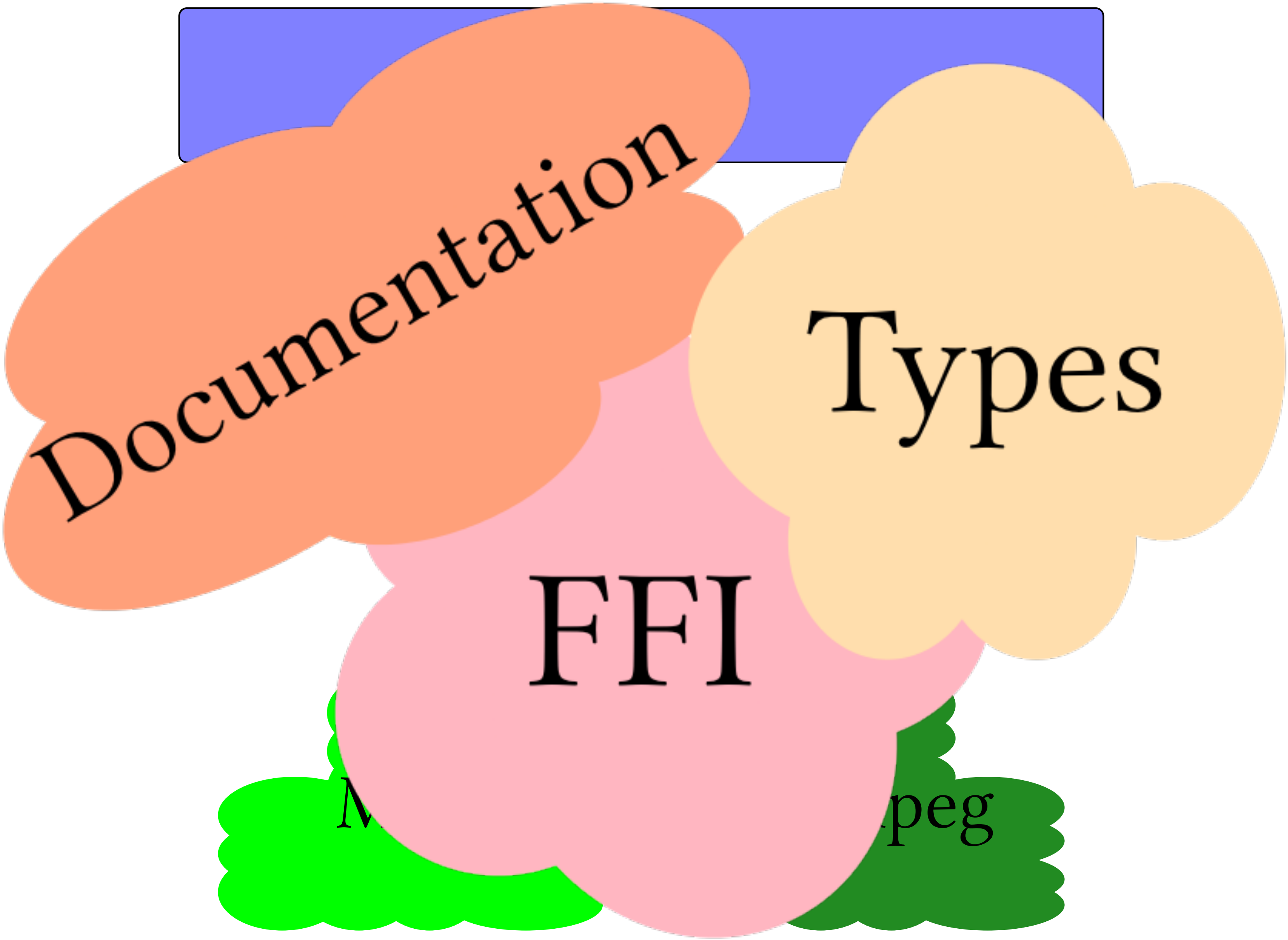
the tower of languages



Video

MLT

FFmpeg



Documentation

Types

FFI

M

peg

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...

DSLs are the

"Ultimate Abstraction"

Paul Hudak

Documentation FFI Types

We have a problem…

Documentation

FFI

Types

We have a problem...

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

Documentation FFI Types

We have a ∨ problem...

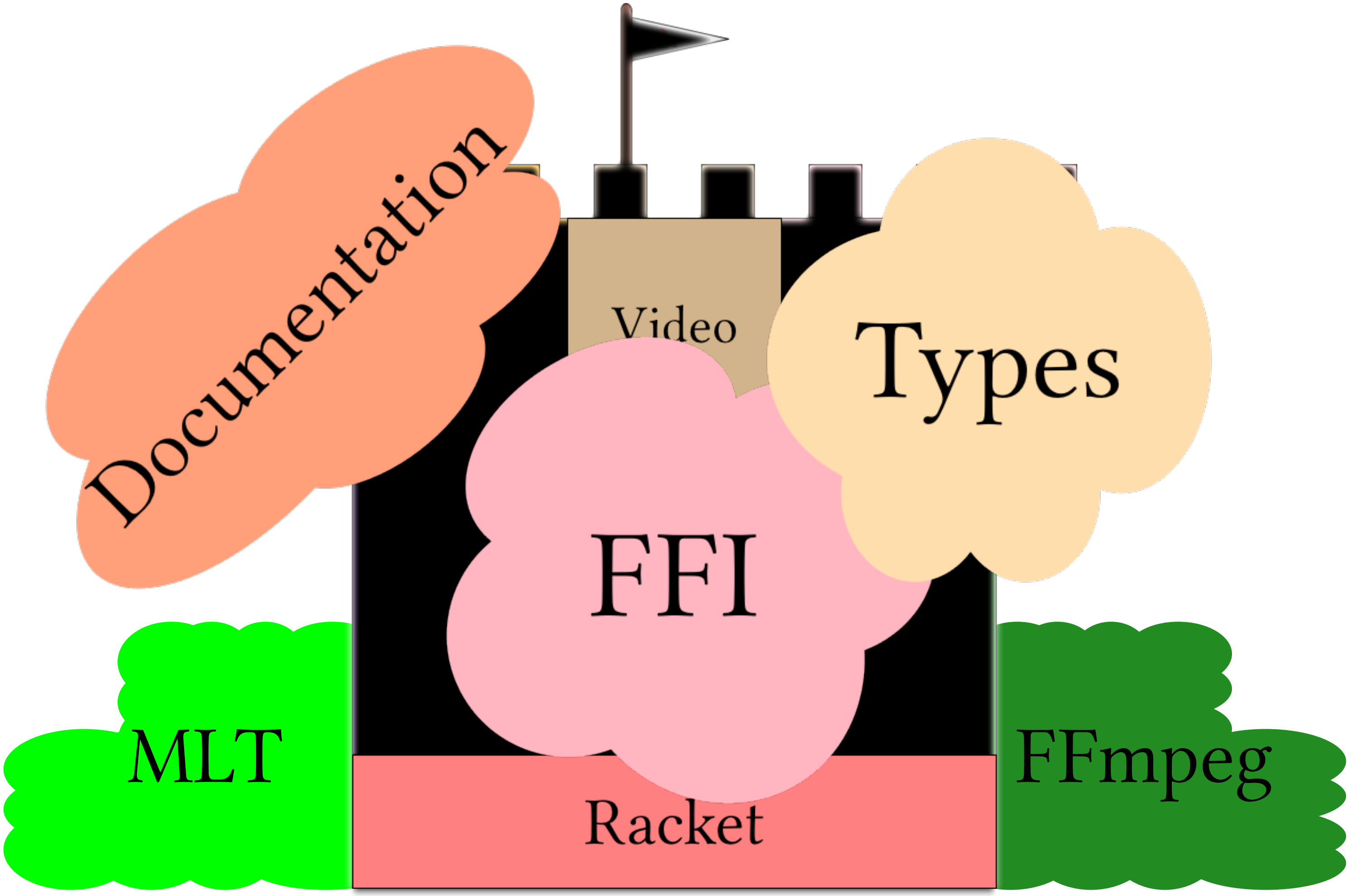
We want to solve it in the problem domain's own language...

Tower of DSLs

Language Oriented Programming

We want to make DSLs quickly…

Use Racket, a programmable programming language

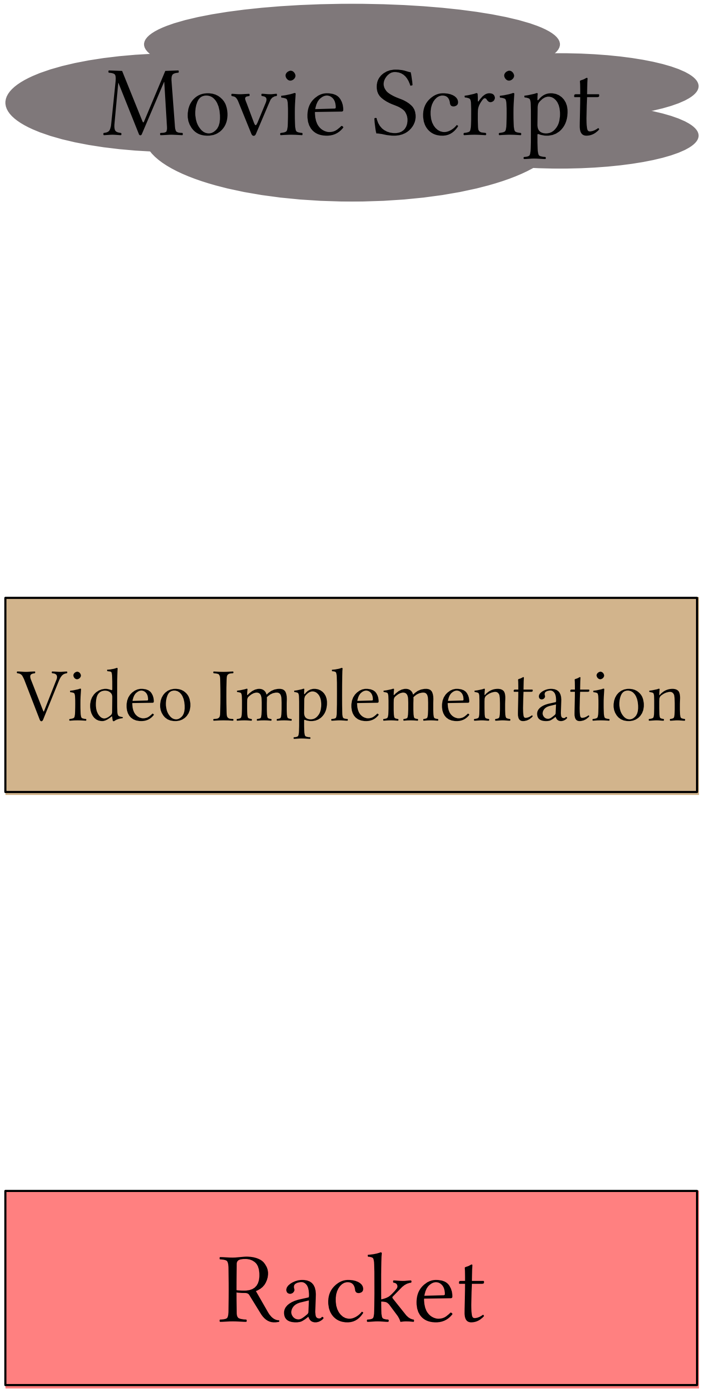


We make DSLs using

Linguistic Inheritance

We make DSLs using

Linguistic Inheritance



Movie Script

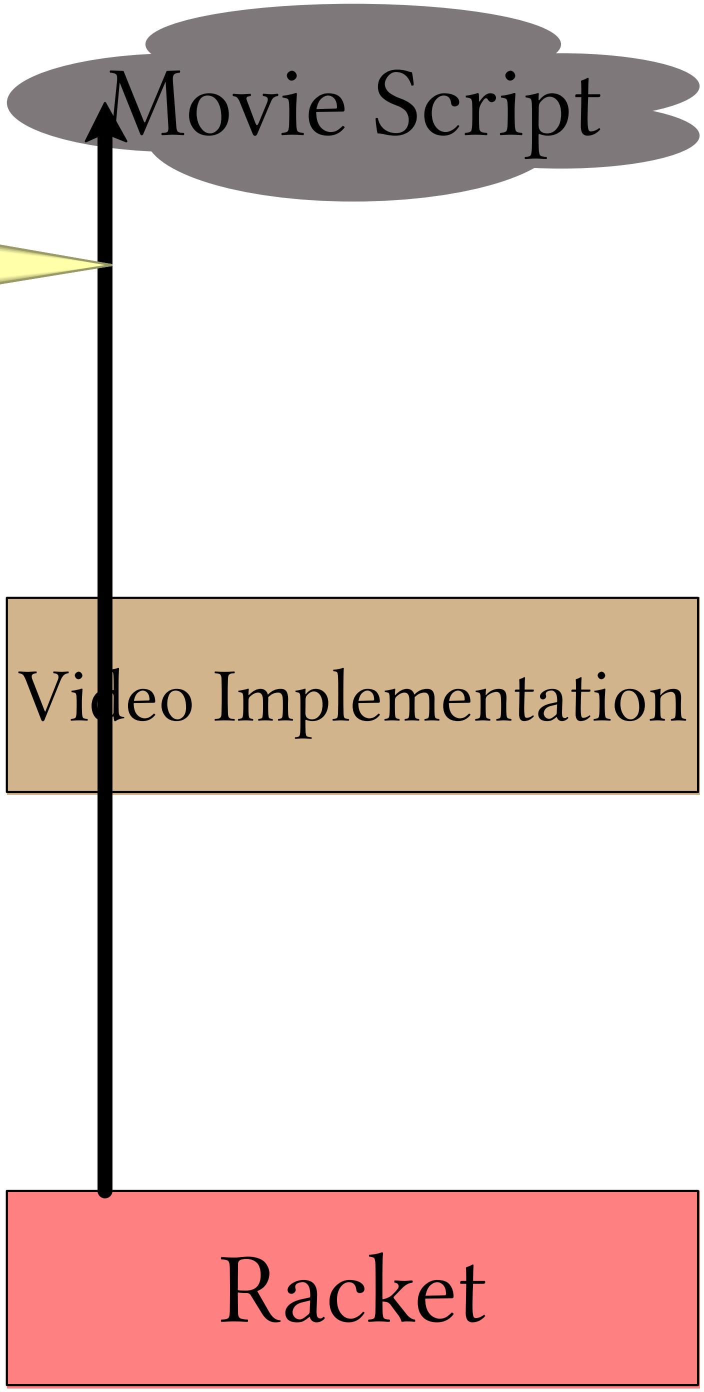
Video Implementation

Racket

We make DSLs using

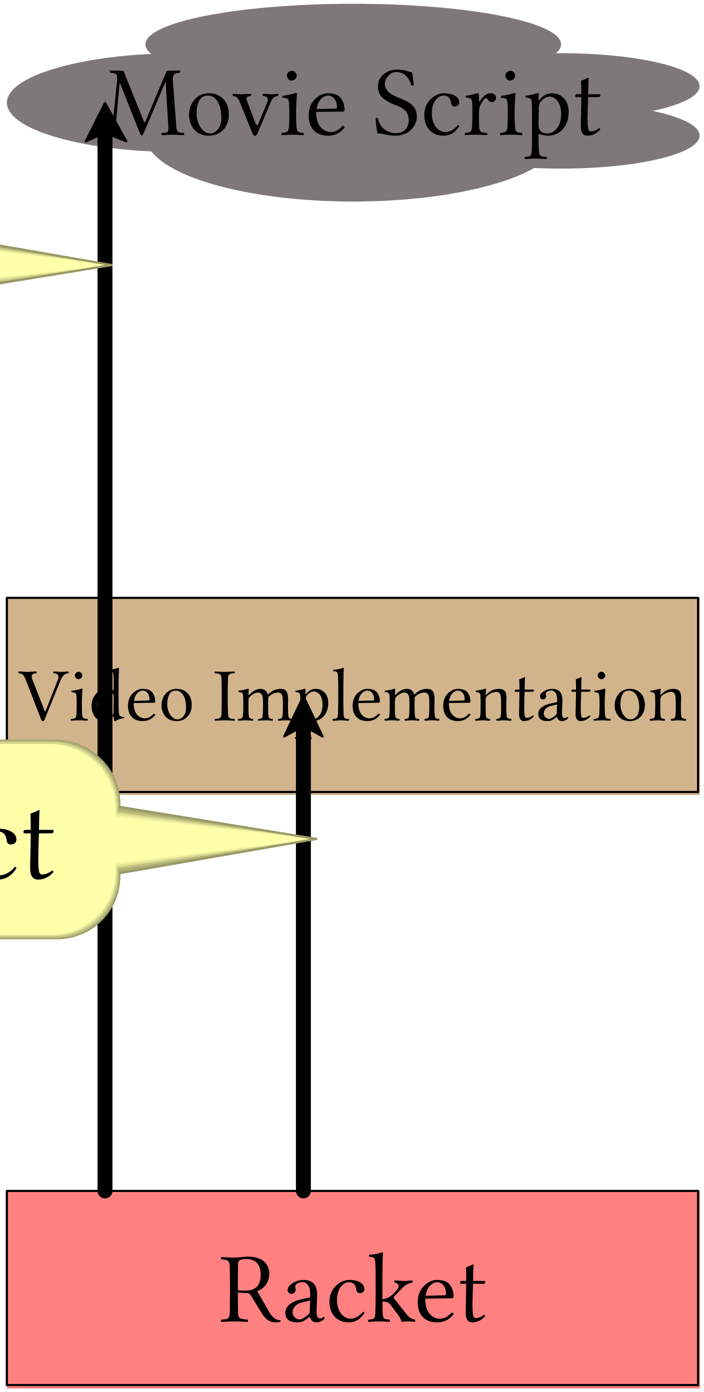
Linguistic Inheritance

Re-export construct



We make DSLs using

Linguistic Inheritance

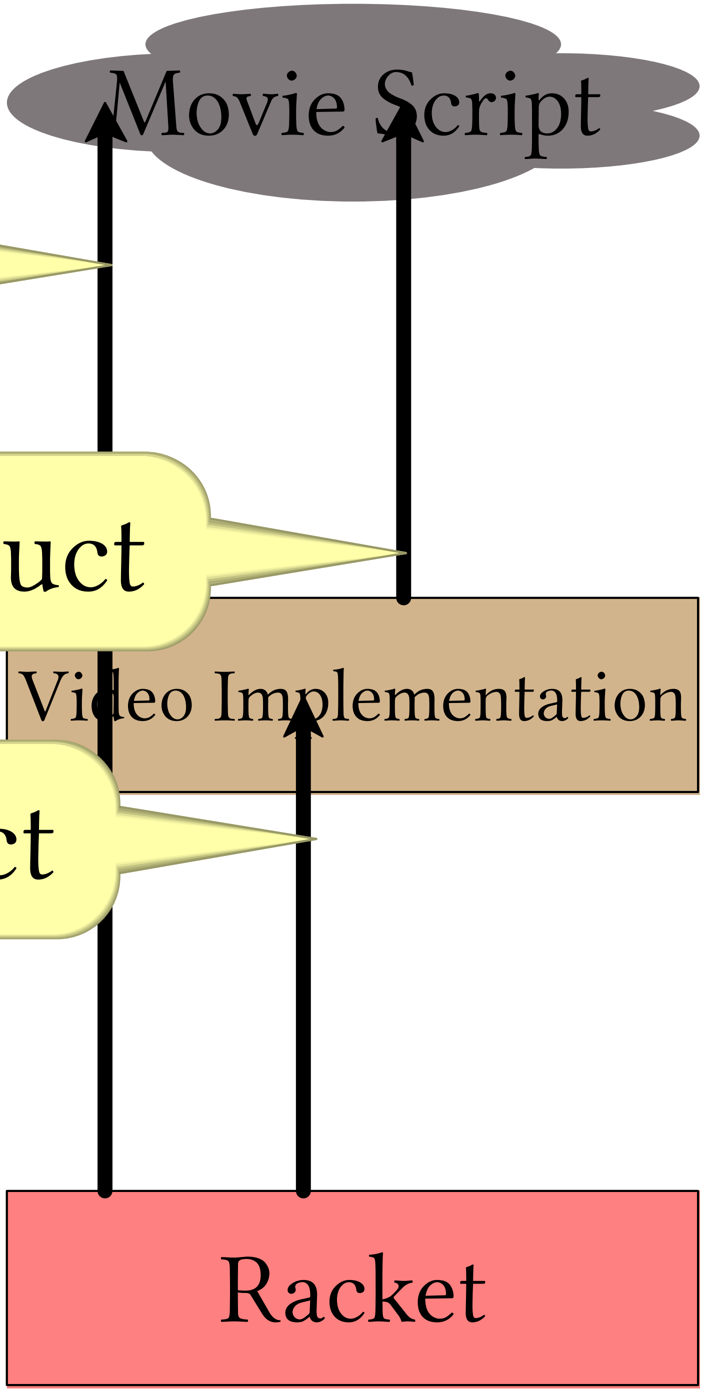


Re-export construct

Remove construct

We make DSLs using

Linguistic Inheritance



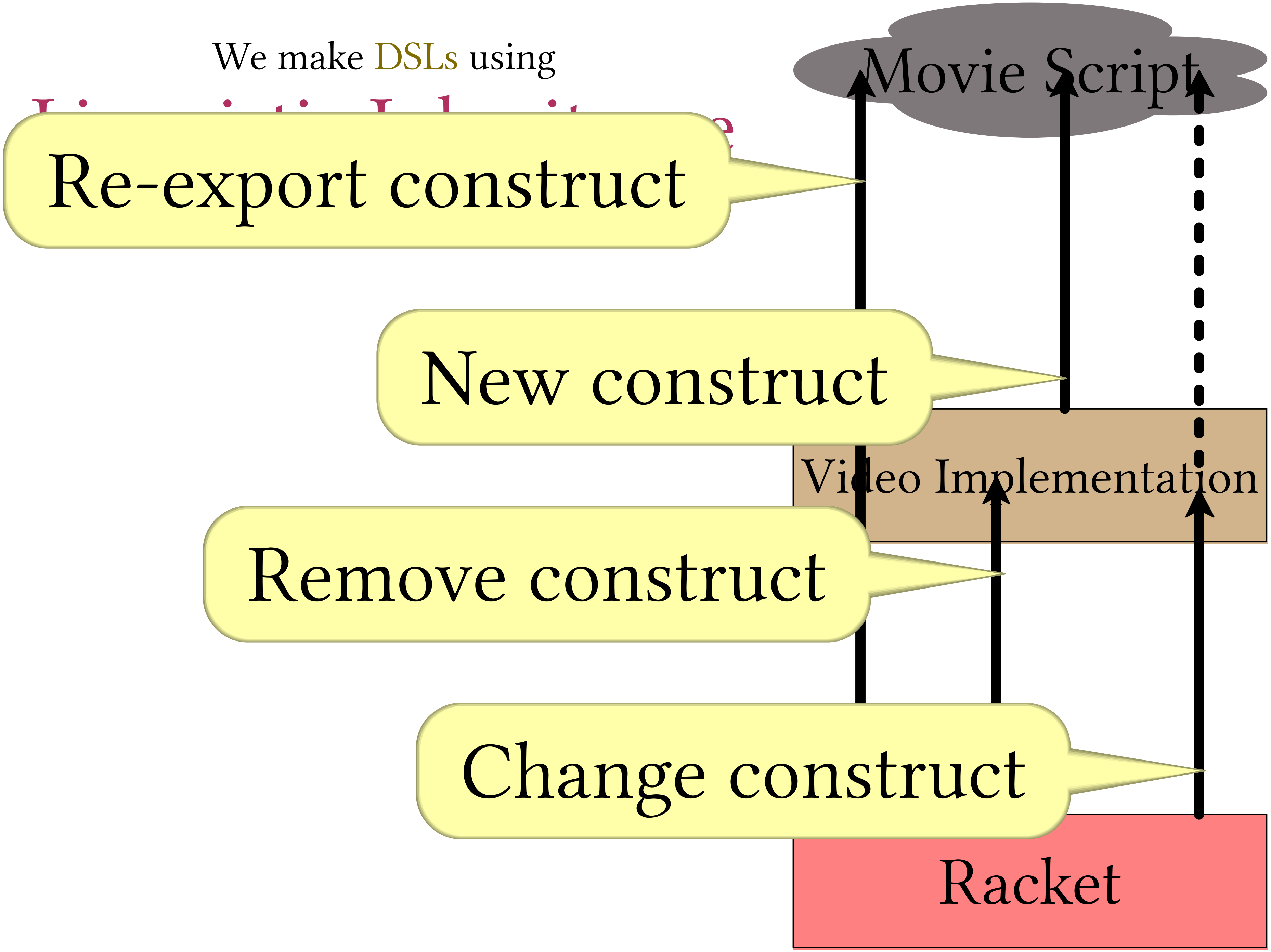
Re-export construct

New construct

Remove construct

We make DSLs using

Linguistic Inheritance



Re-export construct

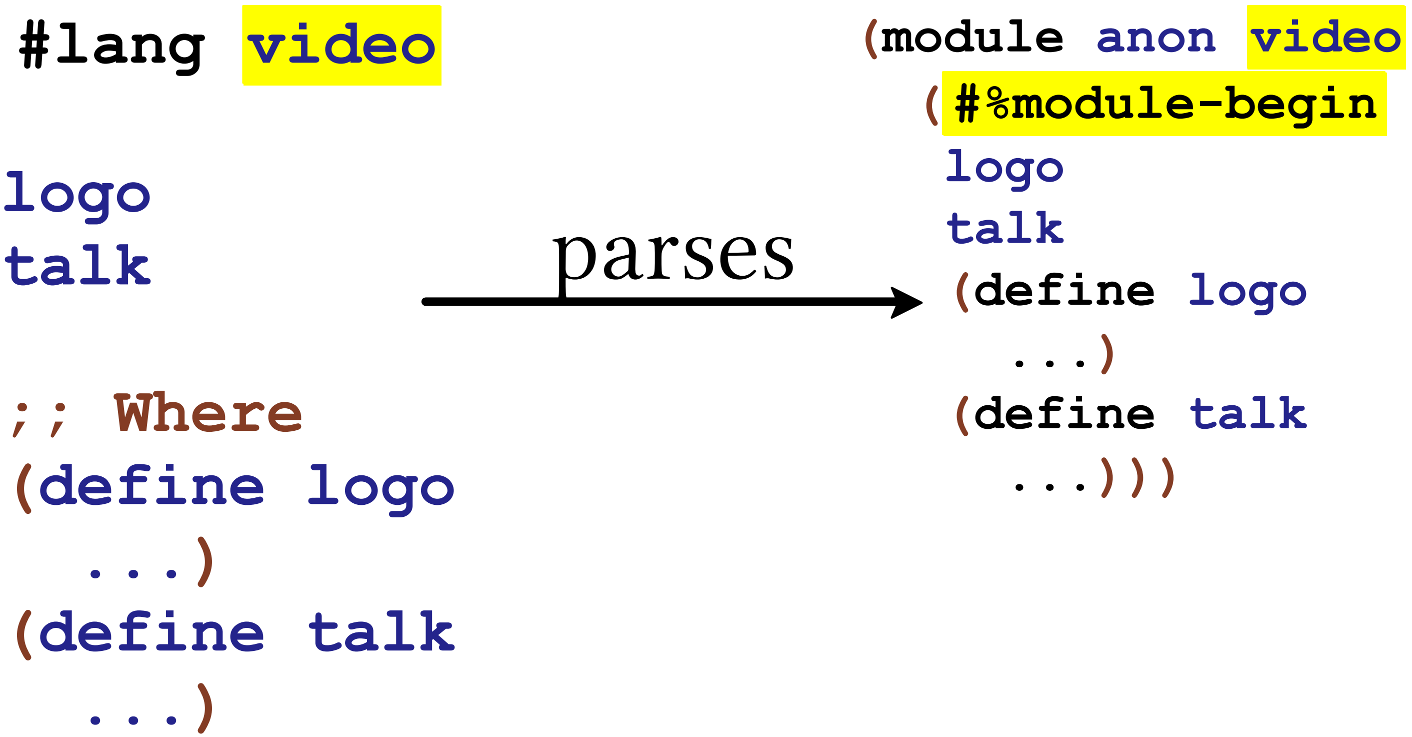
New construct

Remove construct

Change construct

Change construct

Interposition Points

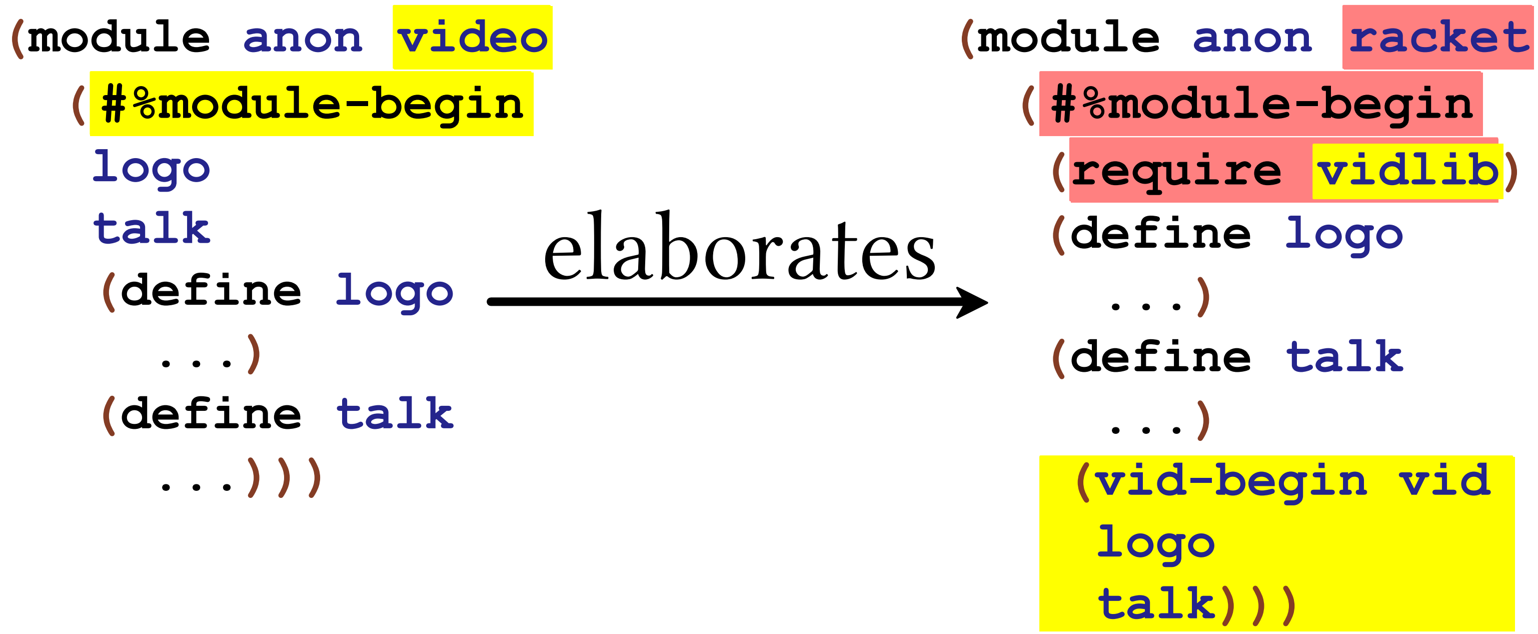


```
#lang video

logo
talk

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

parses

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

Interposition Points

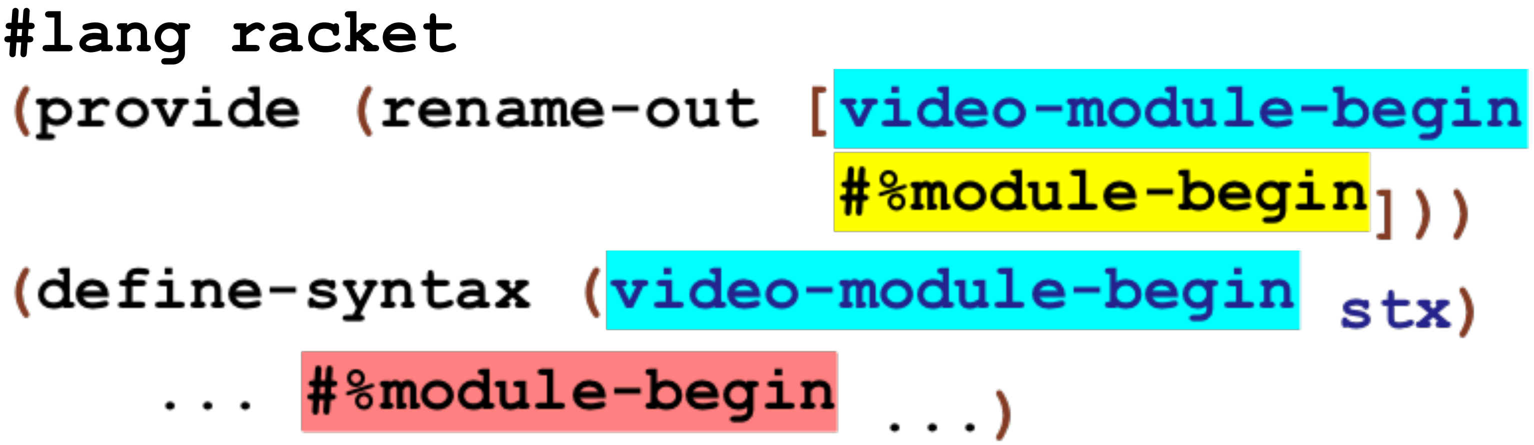


```
(module anon video
  ( #%module-begin
   logo
   talk
   (define logo
     ...)
   (define talk
     ...)))
```

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 ...)
```

FFI

An FFI DSL

```
mlt_repository
mlt_factory_init(const char *directory);
```

(Scheme Wrksp., 2004)

An FFI DSL

```
mlt_repository
mlt_factory_init(const char *directory);
```

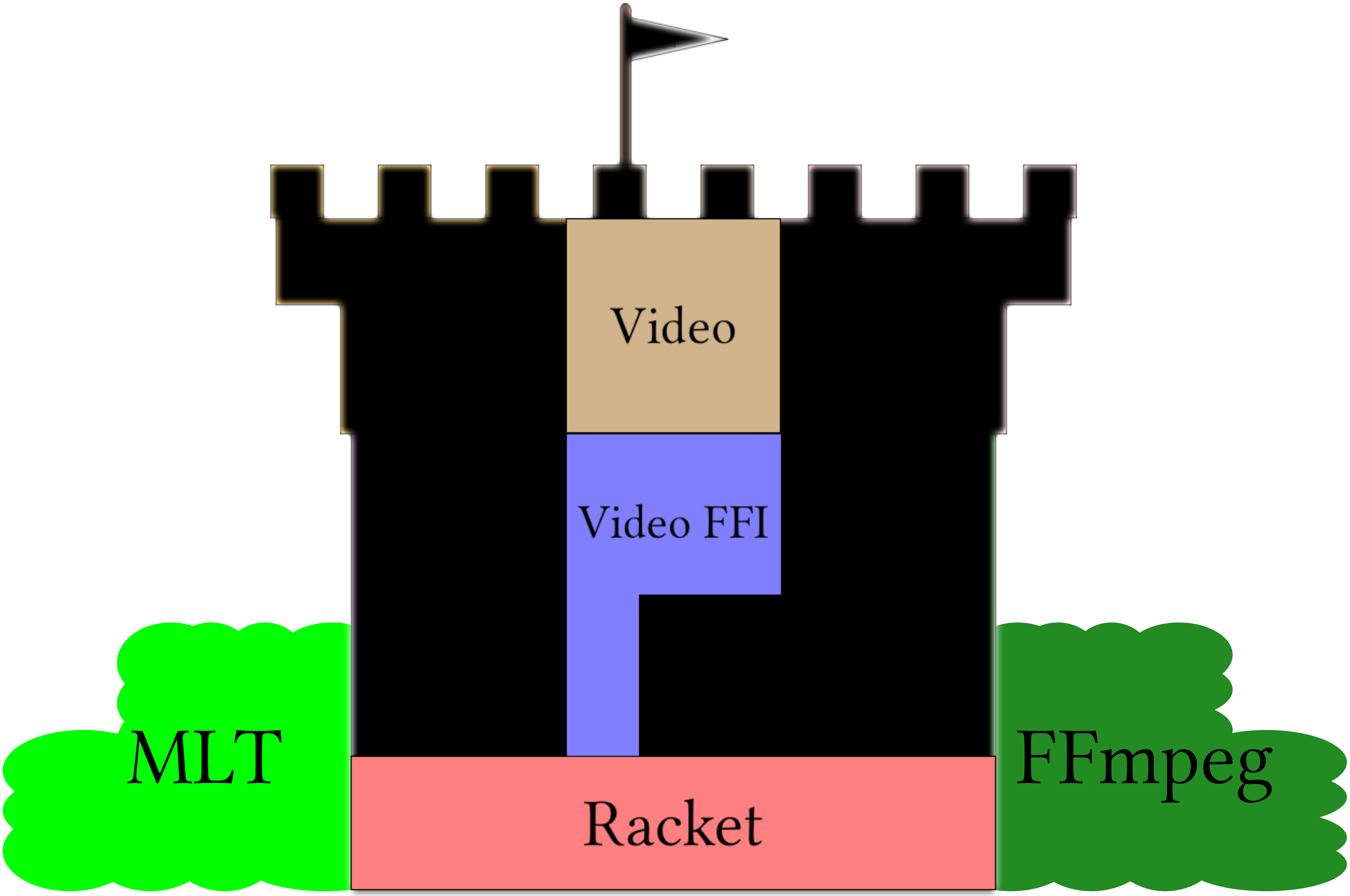
```
(define-mlt mlt-factory-init
  (_fun [p : _path]
        -> [ret : _mlt-repository/null]
        -> (maybe-error? ret)))
```

(Scheme Wrksp., 2004)

An Object DSL

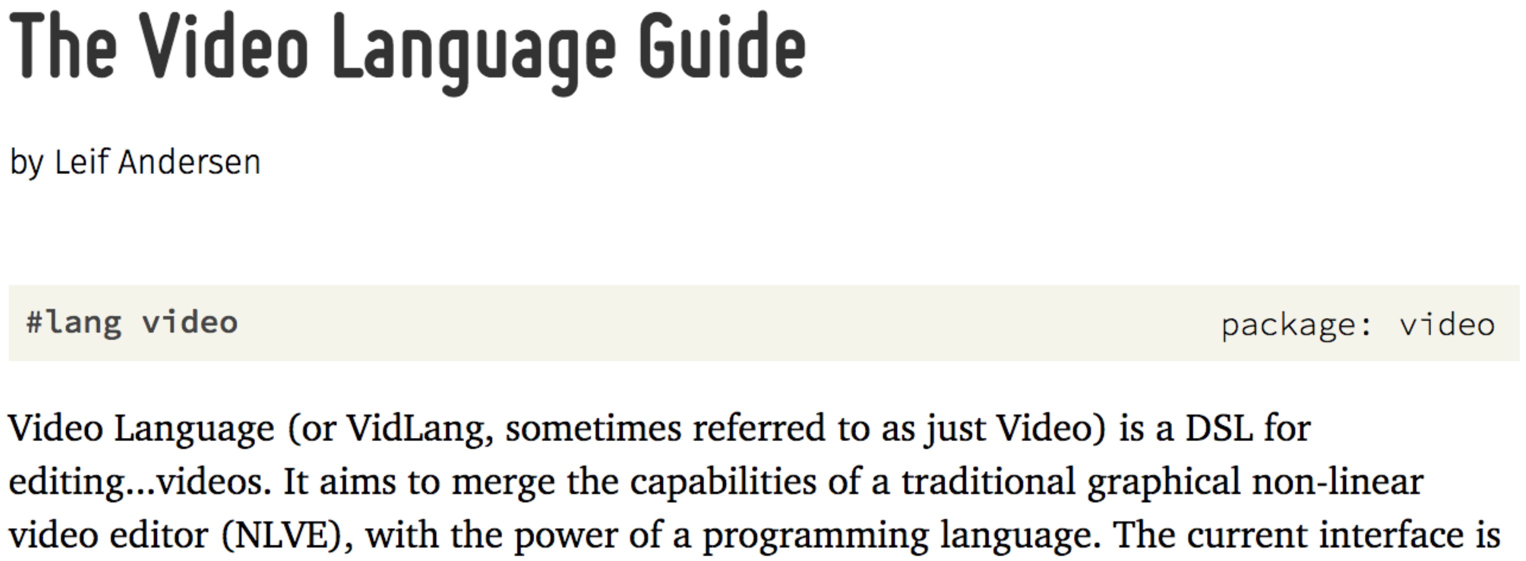
```
(define-mlt mlt-factory-init ...)
(define-mlt mlt-factory-close ...)
```

```
(define-constructor clip video
  ... mlt-factory-init ...
      mlt-factory-close ...)
```


Video
Video FFI
MLT
FFmpeg
Racket

Documentation

A Documentation DSL



The Video Language Guide

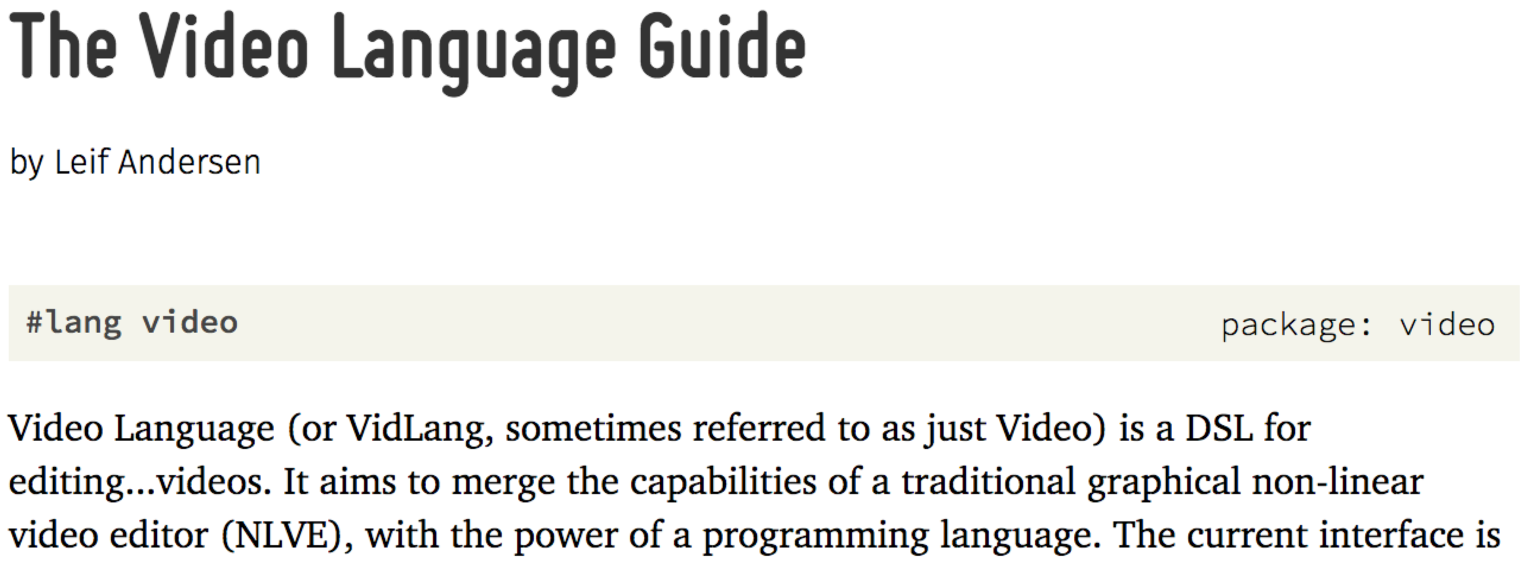
by Leif Andersen

```
#lang video                                                package: video
```

Video Language (or VidLang, sometimes referred to as just Video) is a DSL for editing...videos. It aims to merge the capabilities of a traditional graphical non-linear video editor (NLVE), with the power of a programming language. The current interface is

(ICFP, 2009)

A Documentation DSL



The Video Language Guide

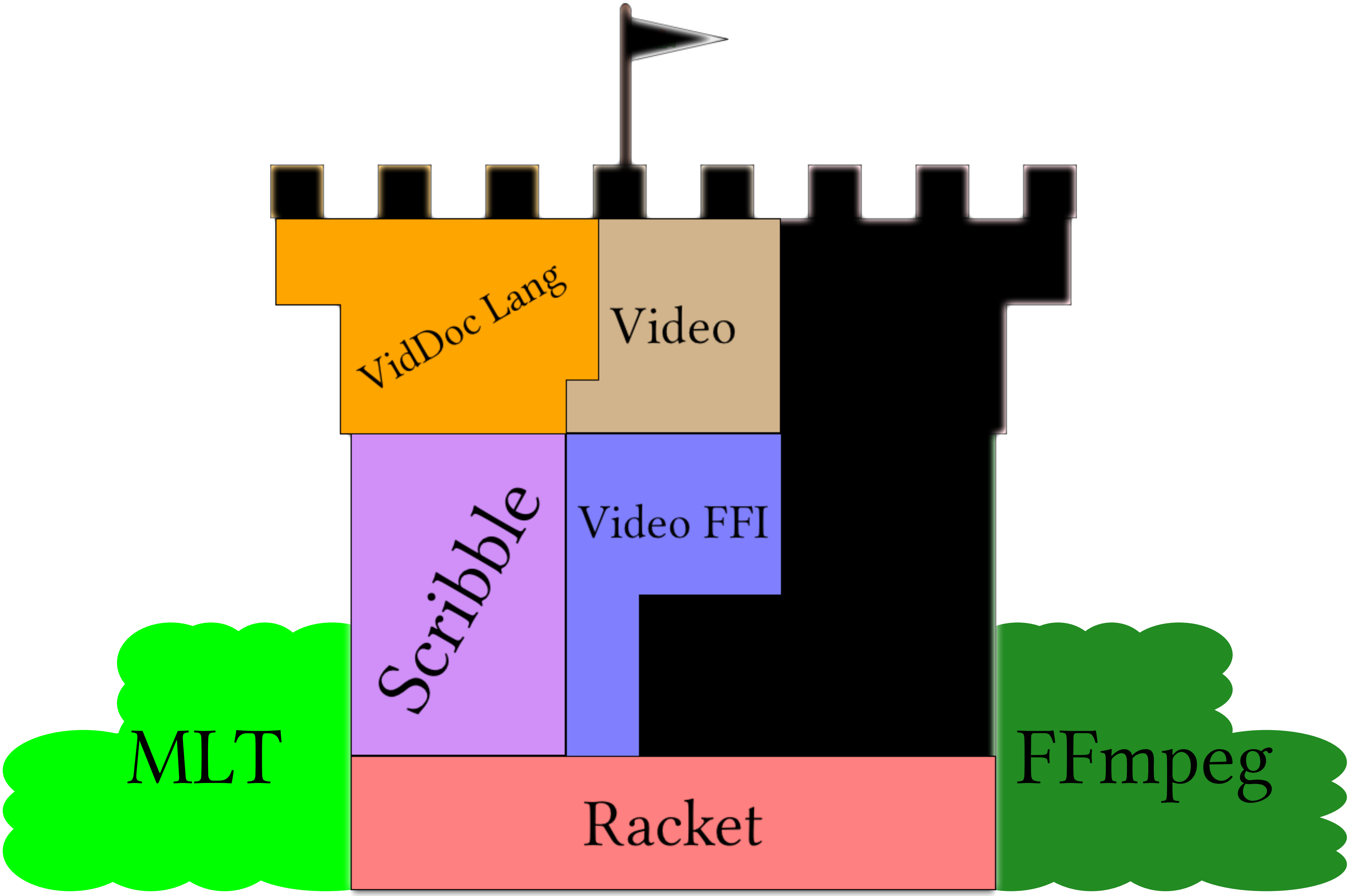
by Leif Andersen

```
#lang video                                                package: video
```

Video Language (or VidLang, sometimes referred to as just Video) is a DSL for editing...videos. It aims to merge the capabilities of a traditional graphical non-linear video editor (NLVE), with the power of a programming language. The current interface is

```
#lang video/documentation
@title{Video: The Language}
@(defmodulelang video)

Video Language (or VidLang, sometimes referred
to as just Video) is a DSL for editing...videos.
It aims to merge the capabilities of a traditional
```

(ICFP, 2009)


VidDoc Lang
Video
Scribble
Video FFI
MLT
FFmpeg
Racket

Types

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

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

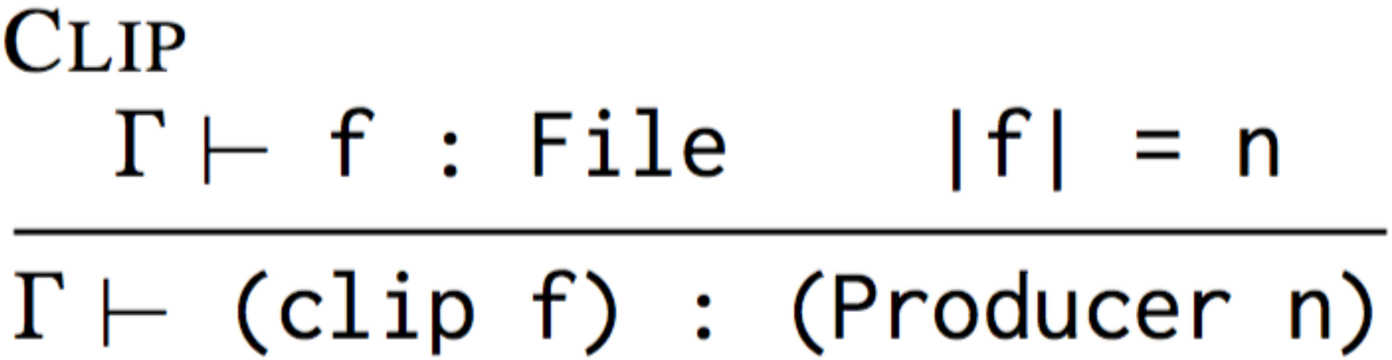
❌

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

A Typed DSL

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

A Typed DSL

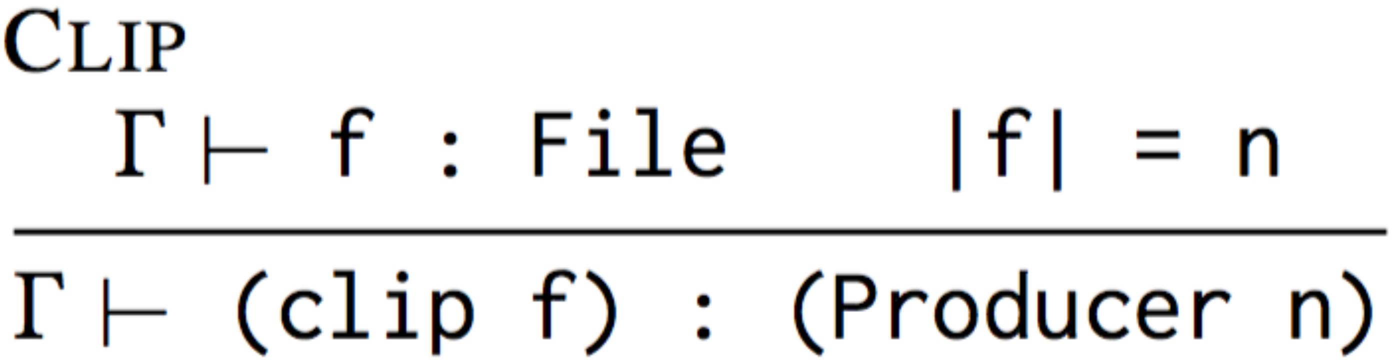


Clip

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

(POPL, 2016)

A Type Implementation DSL

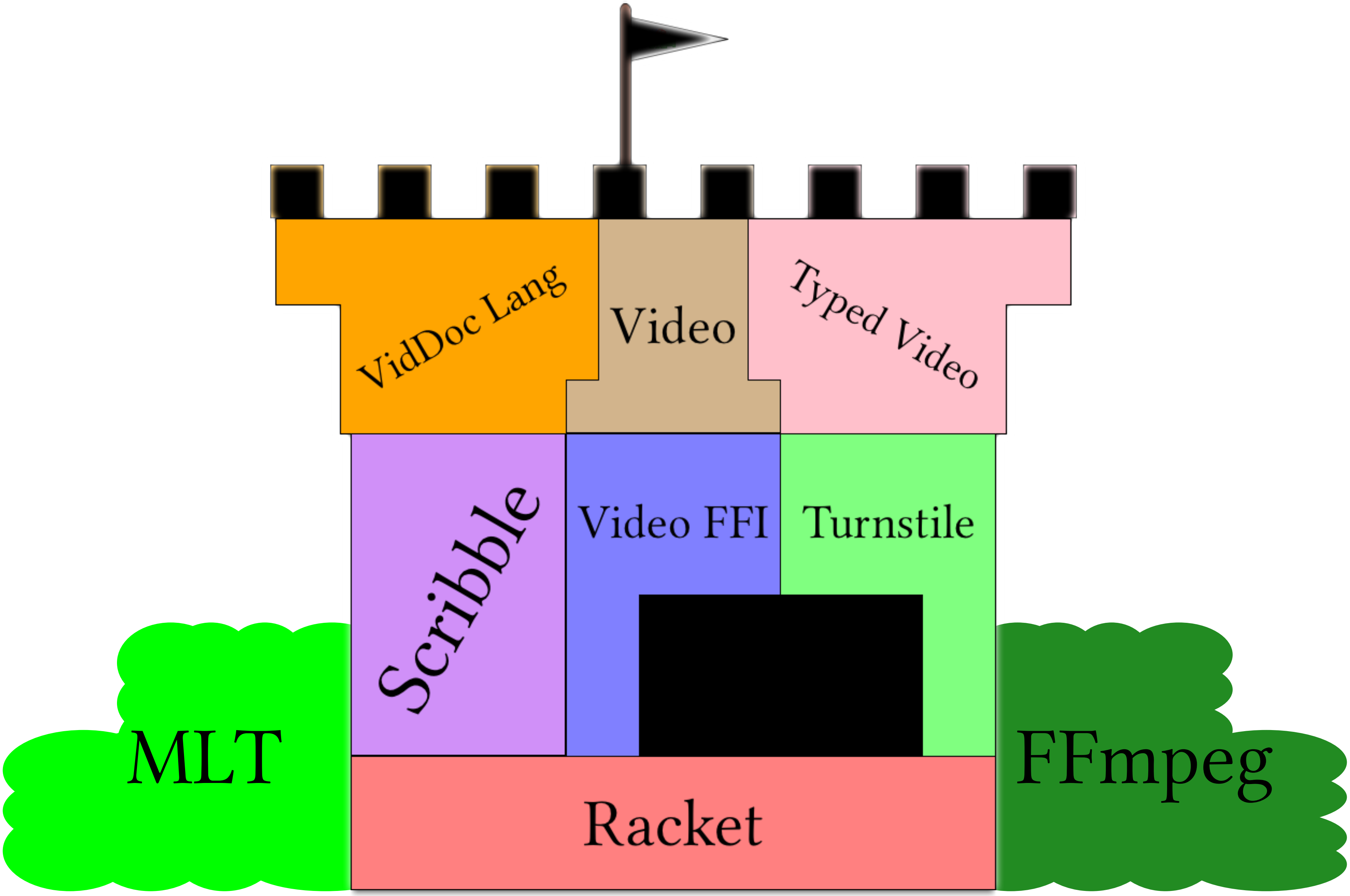


Clip

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

```
(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
MLT
FFmpeg
Racket

We have a DSL problem...

We have a DSL problem...

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

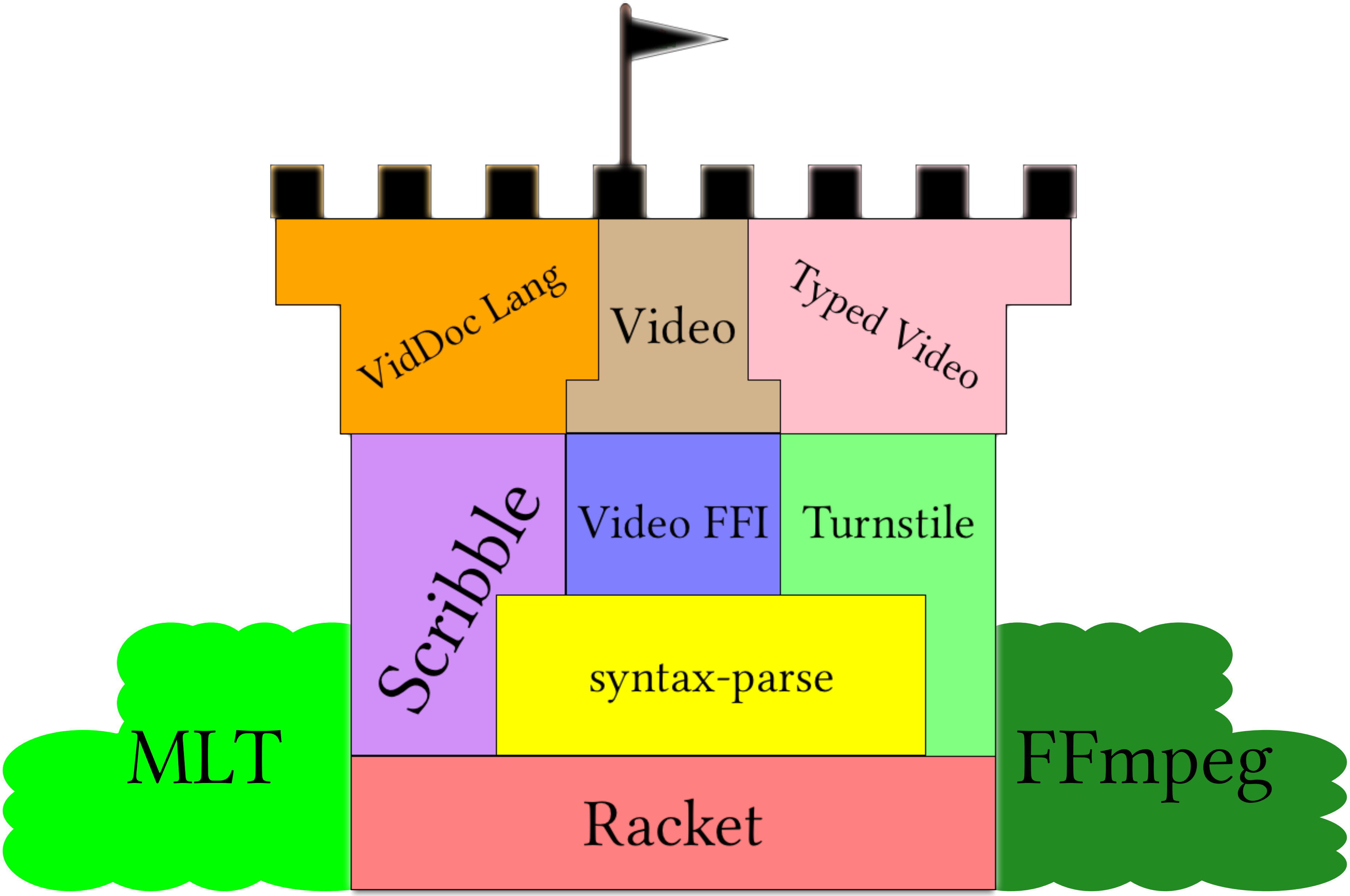
We have a DSL problem...

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

syntax-parse

A DSL for making DSLs

(ICFP, 2010)


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

EDITOR

We have a problem...

We have a EDITOR problem...

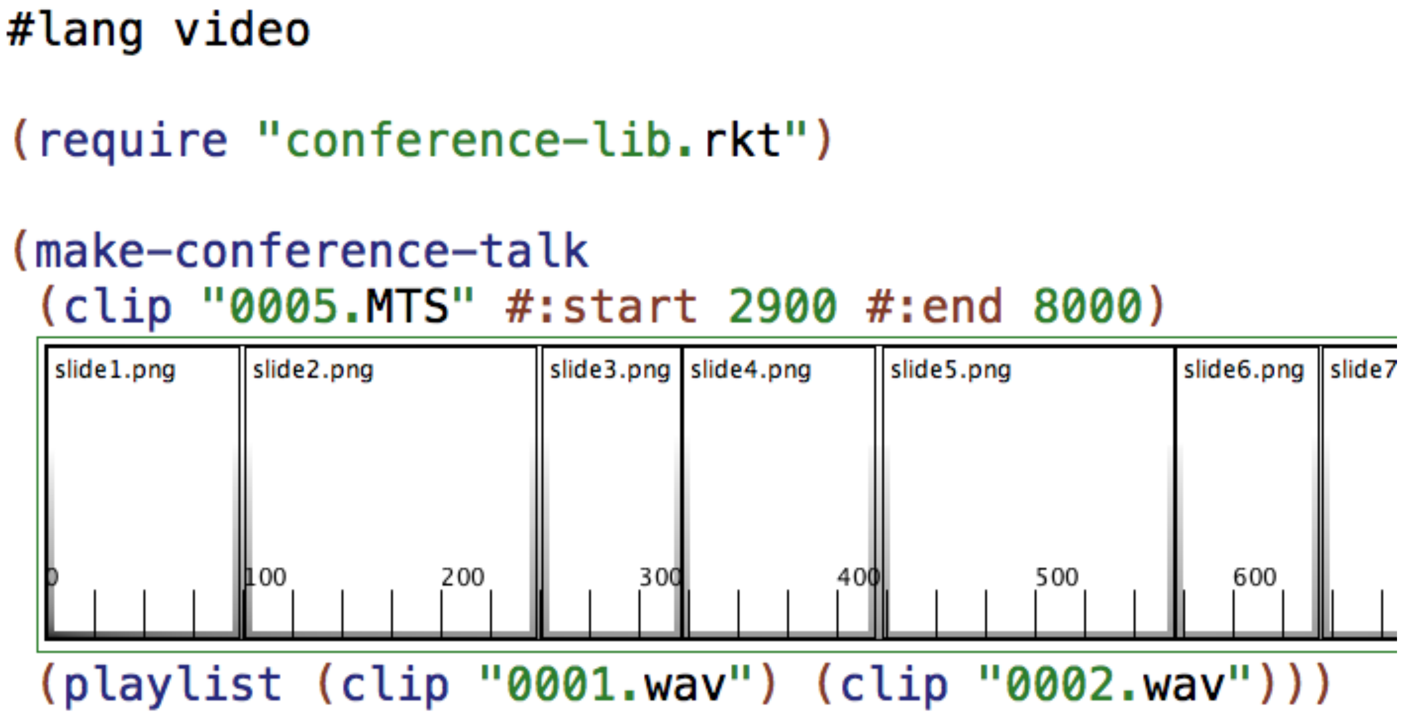
We want to solve it in the problem domain's own language...

We have a EDITOR problem...

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

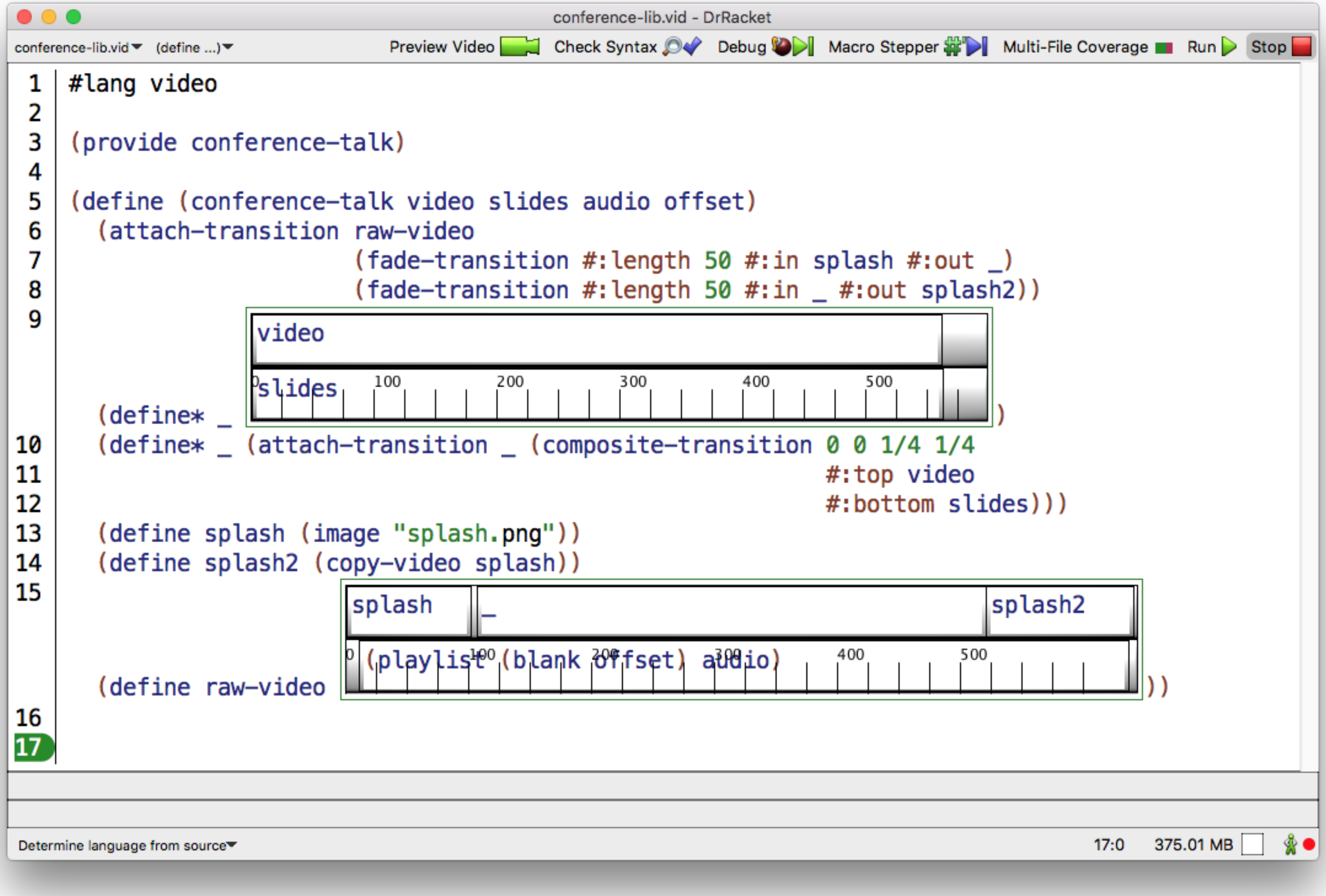
We make DSLs using

? Linguistic Inheritance ?



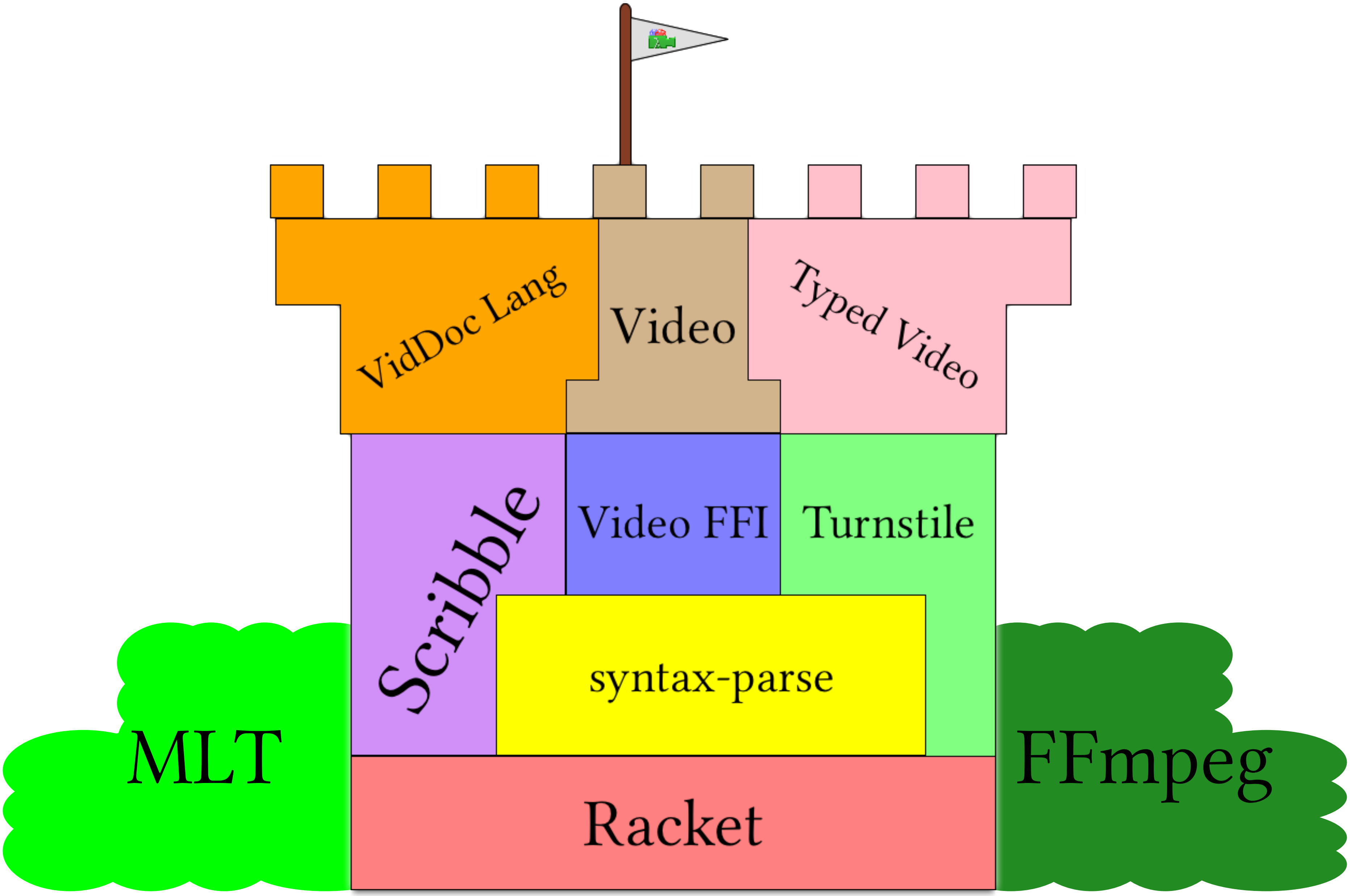
```
#lang video

(require "conference-lib.rkt")

(make-conference-talk
 (clip "0005.MTS" #:start 2900 #:end 8000)
```

```
 (playlist (clip "0001.wav") (clip "0002.wav")))
```



```
#lang video

(provide conference-talk)

(define (conference-talk video slides audio offset)
  (attach-transition raw-video
                     (fade-transition #:length 50 #:in splash #:out _)
                     (fade-transition #:length 50 #:in _ #:out splash2))
             video
             slides
  (define* _                                                      )
  (define* _ (attach-transition _ (composite-transition 0 0 1/4 1/4
                                                        #:top video
                                                        #:bottom slides)))

  (define splash (image "splash.png"))
  (define splash2 (copy-video splash))
                     splash   _                                splash2
                     (playlist (blank offset) audio)
  (define raw-video                                                    ))
```

Future Work

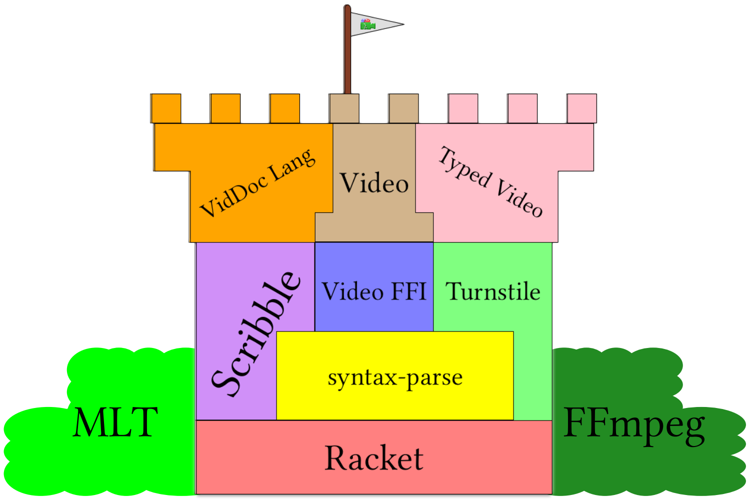

VidDoc Lang
Video
Typed Video
Scribble
Video FFI
Turnstile
syntax-parse
MLT
Racket
FFmpeg

Thanks For Watching

http://lang.video
@videolang

We make DSLs using

Linguistic Inheritance


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