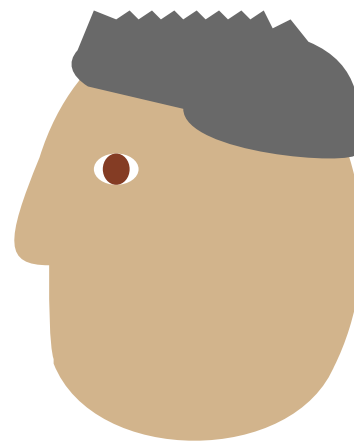
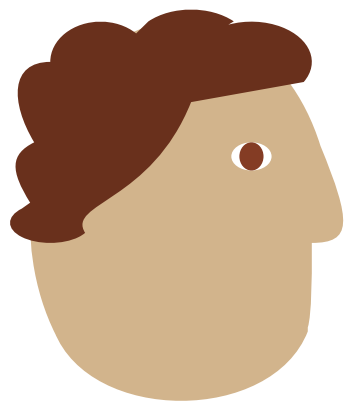
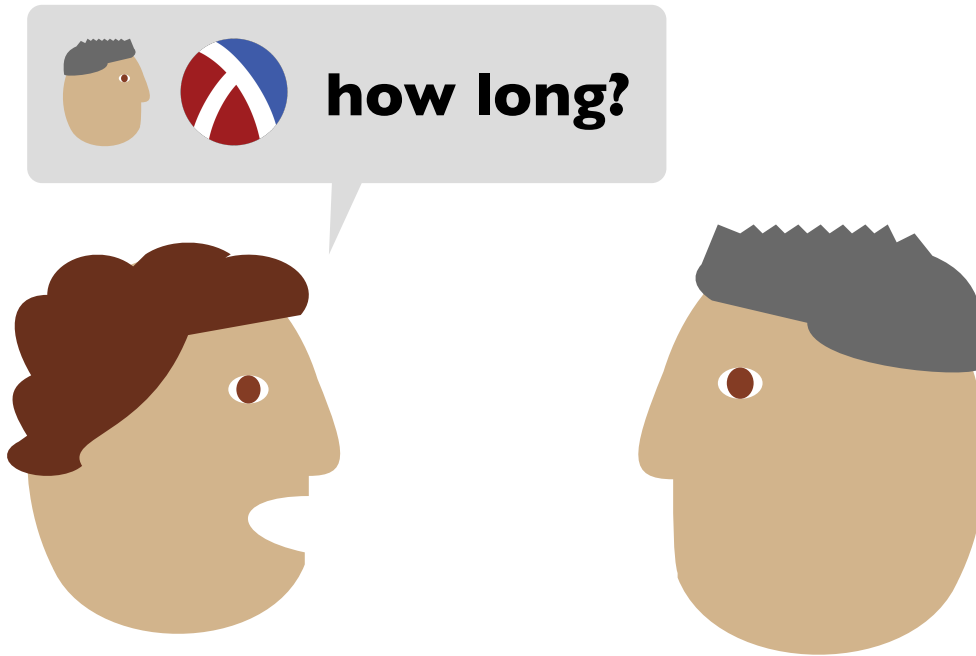


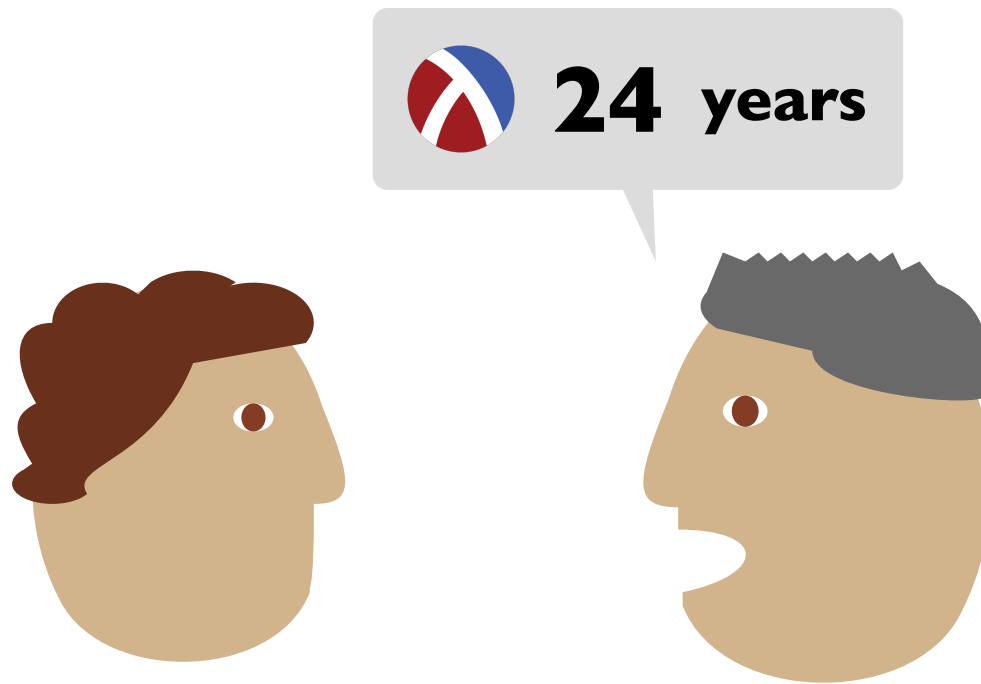
A Racket Perspective on Research, Education, and Production

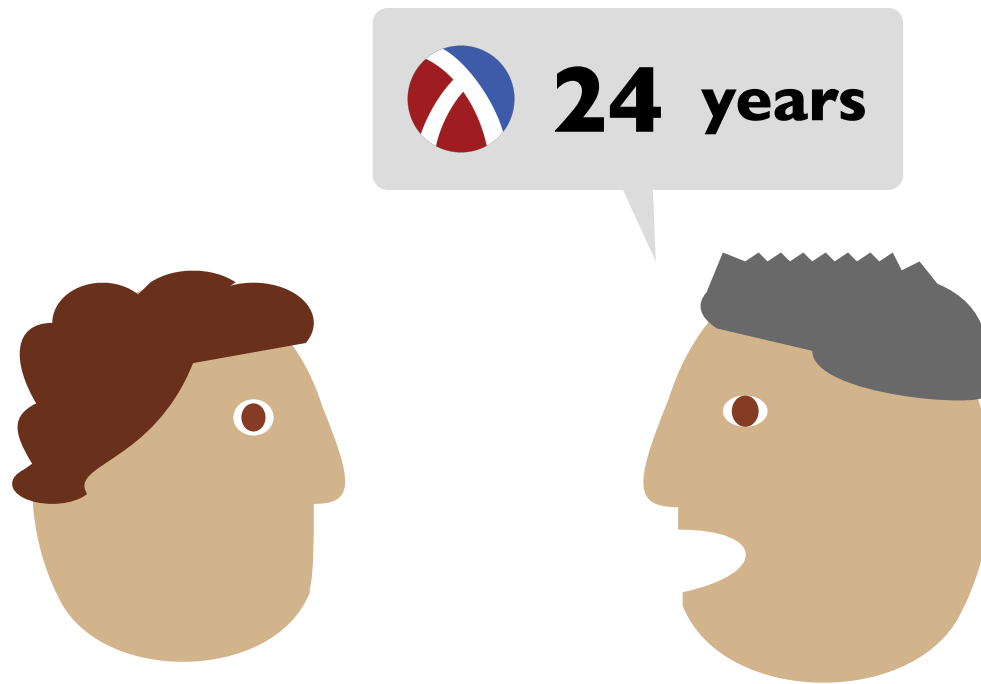


Matthew Flatt
University of Utah

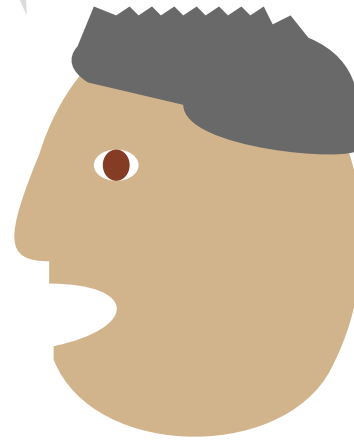
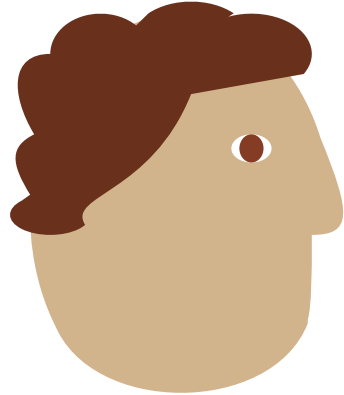


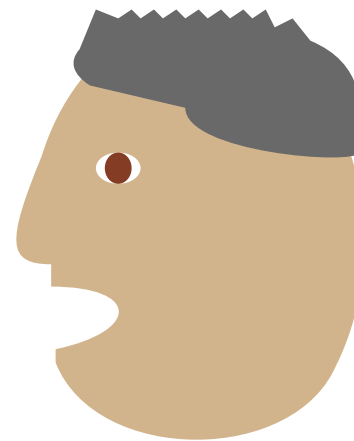
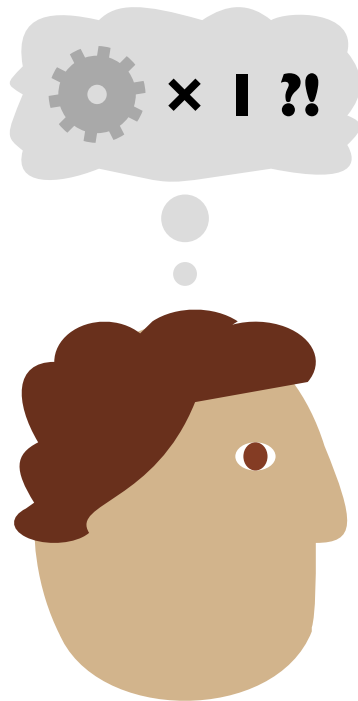






Back in 1995...







`#lang racket`

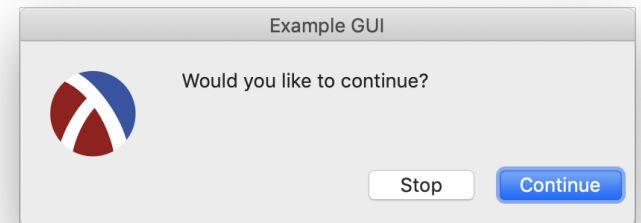


#lang racket



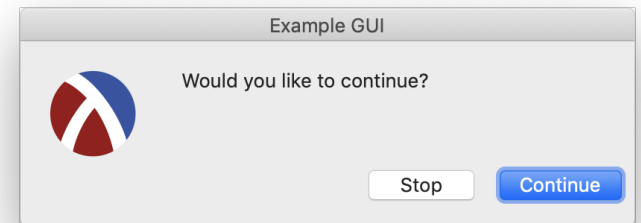
```
subi  (imm 16), %sfp
mov   %ac0, %rcx
mov   (disp 8 %sfp), %ac0
cmpi  (imm 6), %rcx
```

`#lang racket`



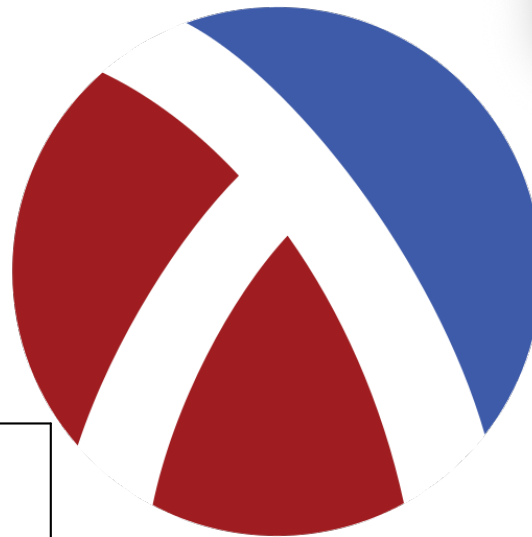
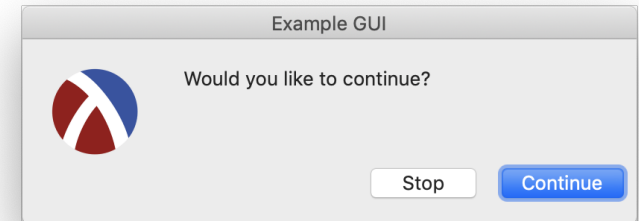
```
subi  (imm 16), %sfp
mov   %ac0, %rcx
mov   (disp 8 %sfp), %ac0
cmpi  (imm 6), %rcx
```

`#lang racket`



```
subi (imm 16), %sfp
mov  %ac0, %rcx
mov  (disp 8 %sfp), %ac0
cmpi (imm 6), %rcx
```

#lang racket



```
subi    (imm 16), %sfp
mov     %ac0, %rcx
mov     (disp 8 %sfp), %ac0
cmpi    (imm 6), %rcx
```

...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

1 Getting Started

2 @ Syntax

3 High-Level Scribble API

4 Scribbling Documentation

5 Literate Programming

6 Low-Level Scribble API

7 Running scribble

Index

ON THIS PAGE:
Scribble: The Racket Documentation Tool

Scribble: The Racket Documentation Tool

by Matthew Flatt and Eli Barzilay

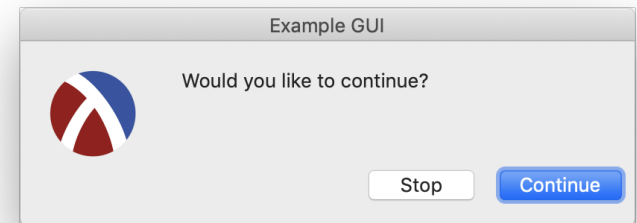
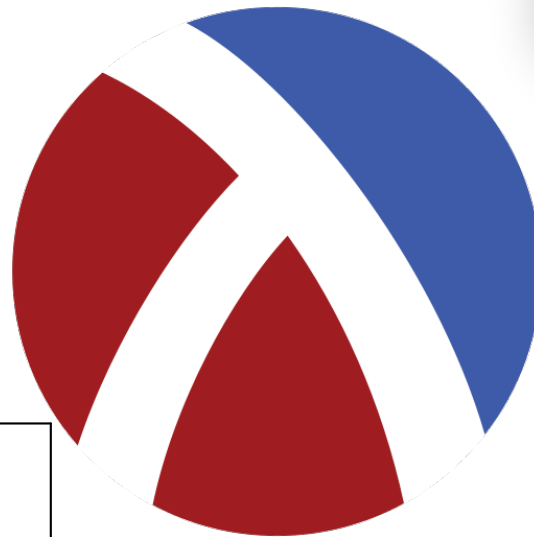
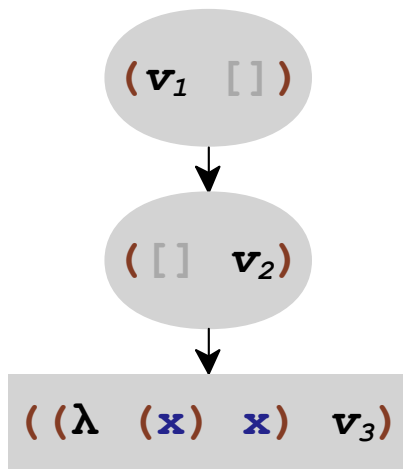
Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via LaTeX) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scribblings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

- 1.1 A First Example
- 1.2 Multiple Sections
- 1.3 Splitting the Document Source
- 1.4 Document Styles

#lang racket



...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

- 1 Getting Started
- 2 @ Syntax
- 3 High-Level Scribble API
- 4 Scribbling Documentation
- 5 Literate Programming
- 6 Low-Level Scribble API
- 7 Running scribble
- Index

ON THIS PAGE:
Scribble: The Racket Documentation Tool

Scribble: The Racket Documentation Tool

by Matthew Flatt and Eli Barzilay

Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via LaTeX) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

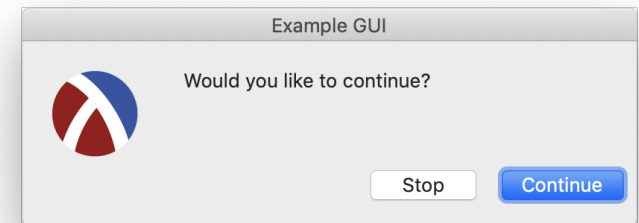
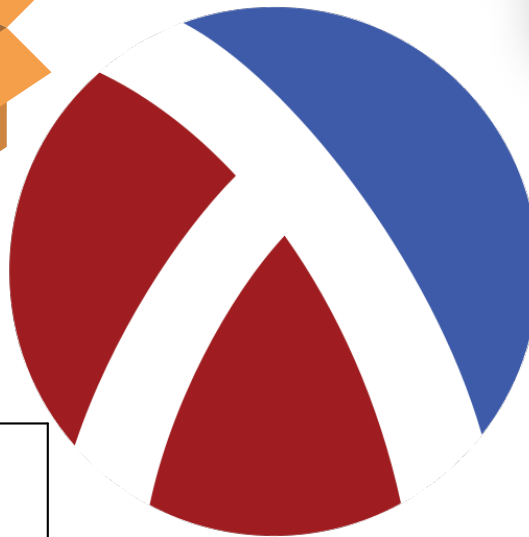
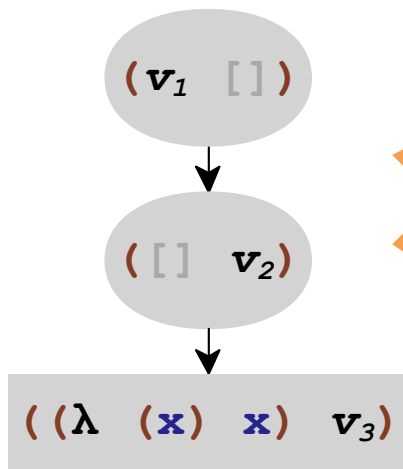
This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scribbings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

- 1.1 A First Example
- 1.2 Multiple Sections
- 1.3 Splitting the Document Source
- 1.4 Document Styles

```
subi    (imm 16), %sfp
mov     %ac0, %rcx
mov     (disp 8 %sfp), %ac0
cmpi    (imm 6), %rcx
```

#lang racket



...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

- 1 Getting Started
- 2 @ Syntax
- 3 High-Level Scribble API
- 4 Scribbling Documentation
- 5 Literate Programming
- 6 Low-Level Scribble API
- 7 Running scribble

Index

ON THIS PAGE:

Scribble: The Racket Documentation Tool

Scribble: The Racket Documentation Tool

by Matthew Flatt and Eli Barzilay

Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via LaTeX) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

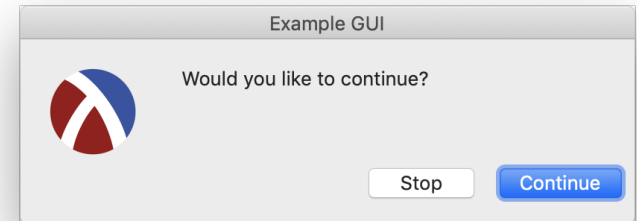
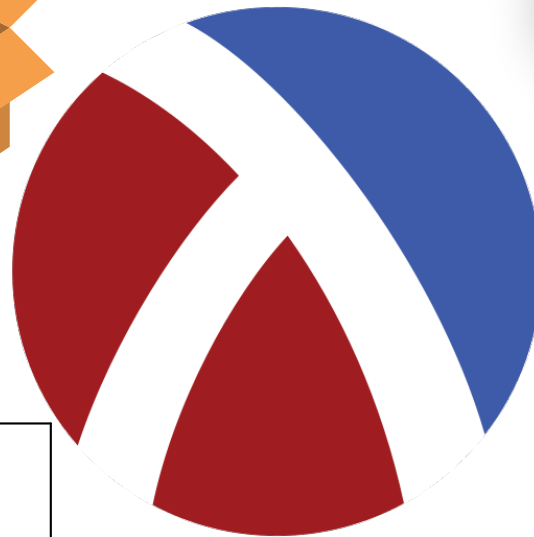
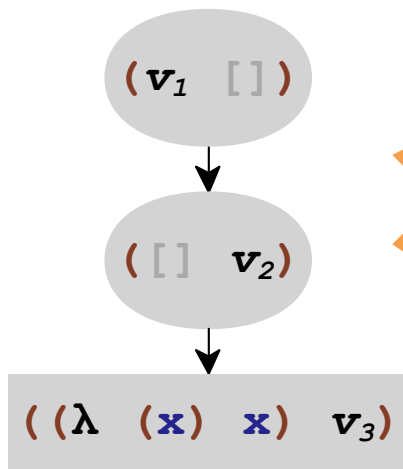
This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scribbings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

- 1.1 A First Example
- 1.2 Multiple Sections
- 1.3 Splitting the Document Source
- 1.4 Document Styles

```
subi    (imm 16), %sfp
mov     %ac0, %rcx
mov     (disp 8 %sfp), %ac0
cmpi    (imm 6), %rcx
```

#lang racket



```
(define-syntax (slides stx)
  (syntax-parse stx
    [(_ id) #'(void)]
    [(_ id e es ...)
     #'(let ([id e])
         (slide id)
         (slides id es ...)))]))
```

```
subi    (imm 16), %sfp
mov     %ac0, %rcx
mov     (disp 8 %sfp), %ac0
cmpi    (imm 6), %rcx
```



...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

- 1 Getting Started
- 2 @ Syntax
- 3 High-Level Scribble API
- 4 Scribbling Documentation
- 5 Literate Programming
- 6 Low-Level Scribble API
- 7 Running scribble
- Index

ON THIS PAGE:

Scribble: The Racket Documentation Tool

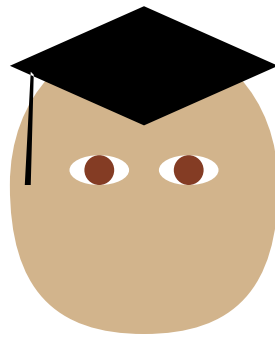
by Matthew Flatt and Eli Barzilay

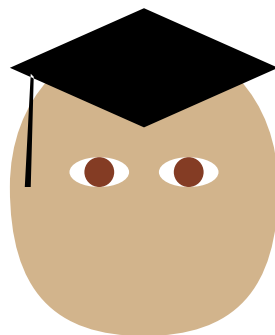
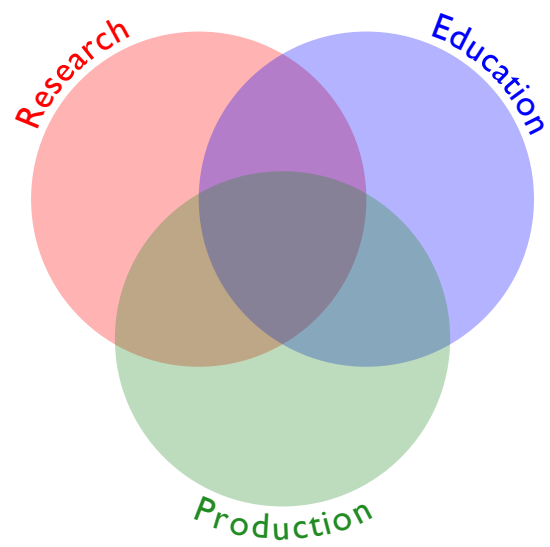
Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via LaTeX) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

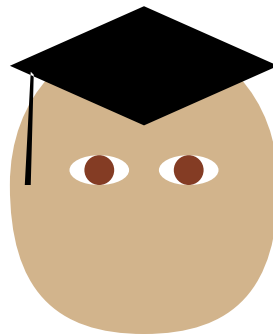
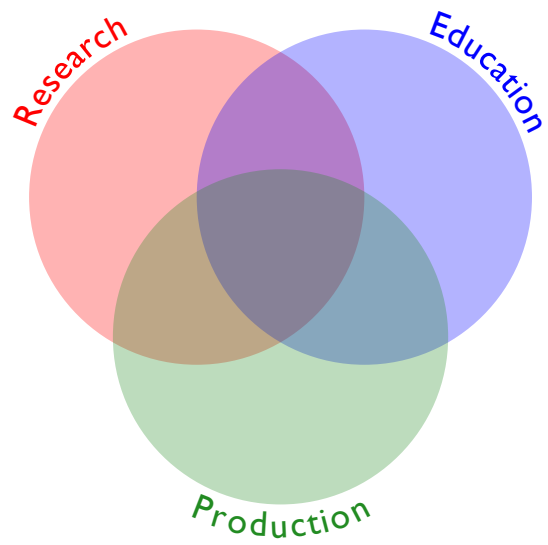
This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scribbings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

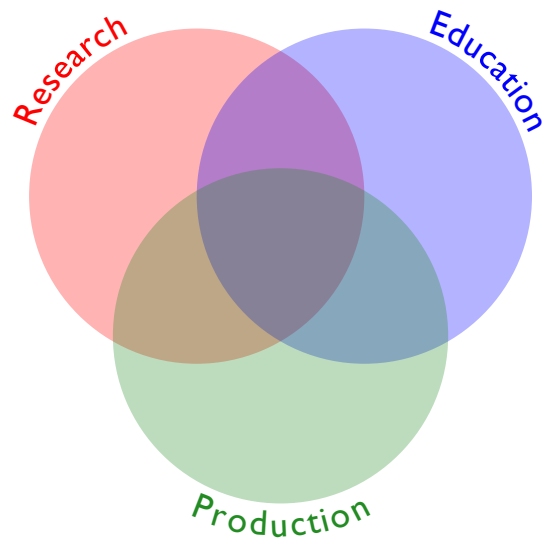
- 1.1 A First Example
- 1.2 Multiple Sections
- 1.3 Splitting the Document Source
- 1.4 Document Styles



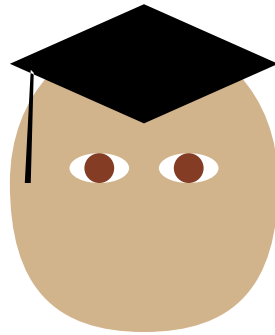




1

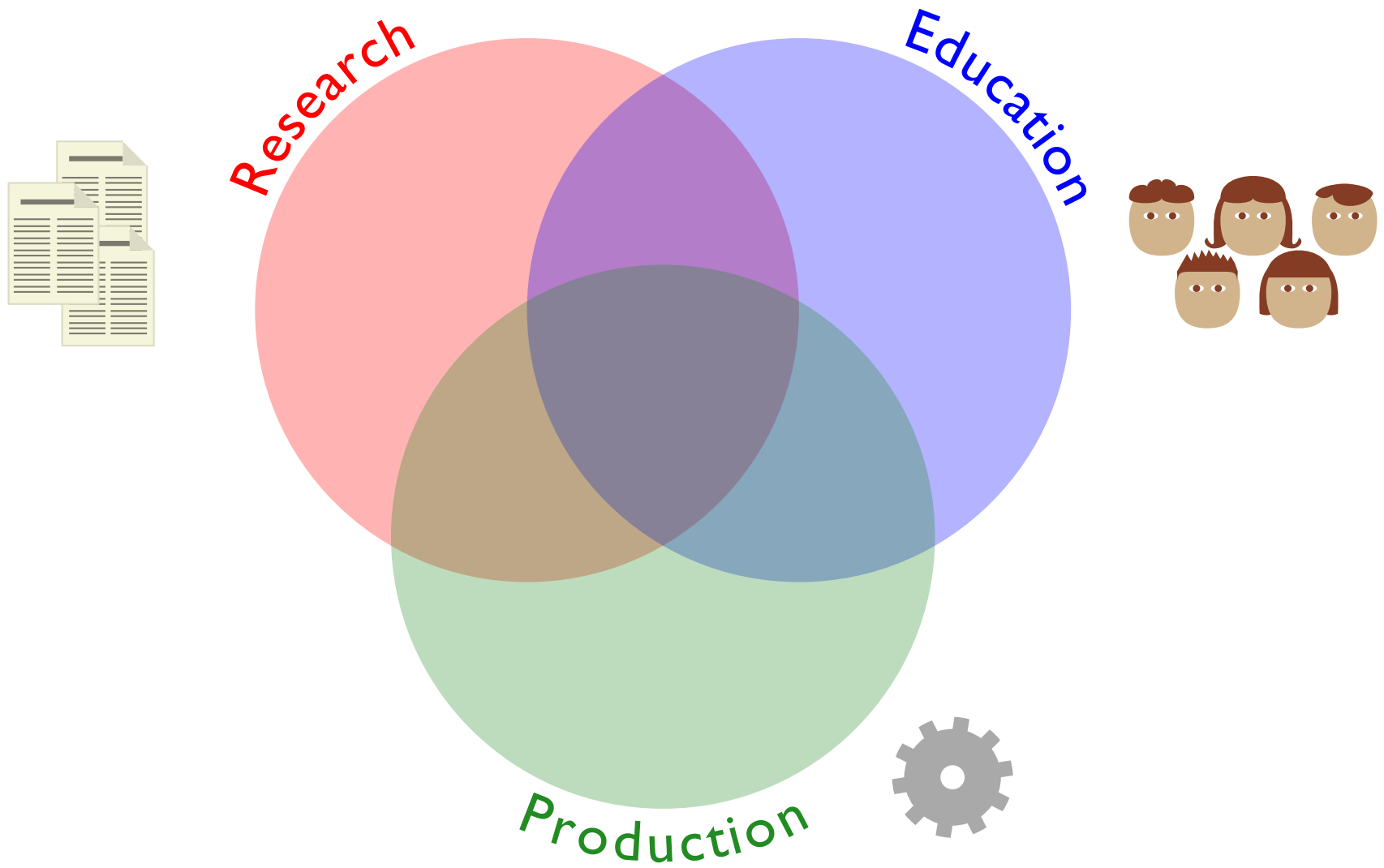


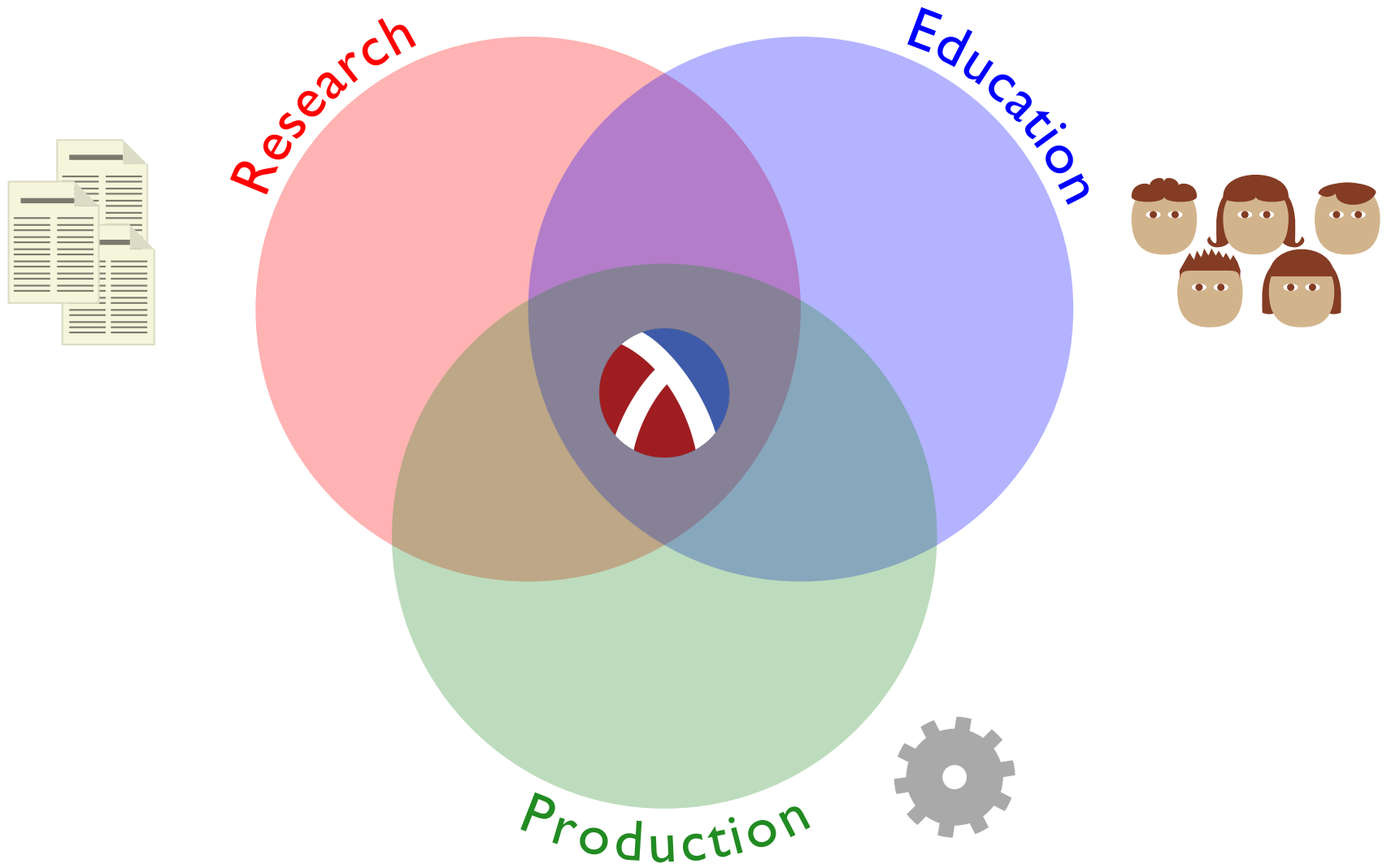
2

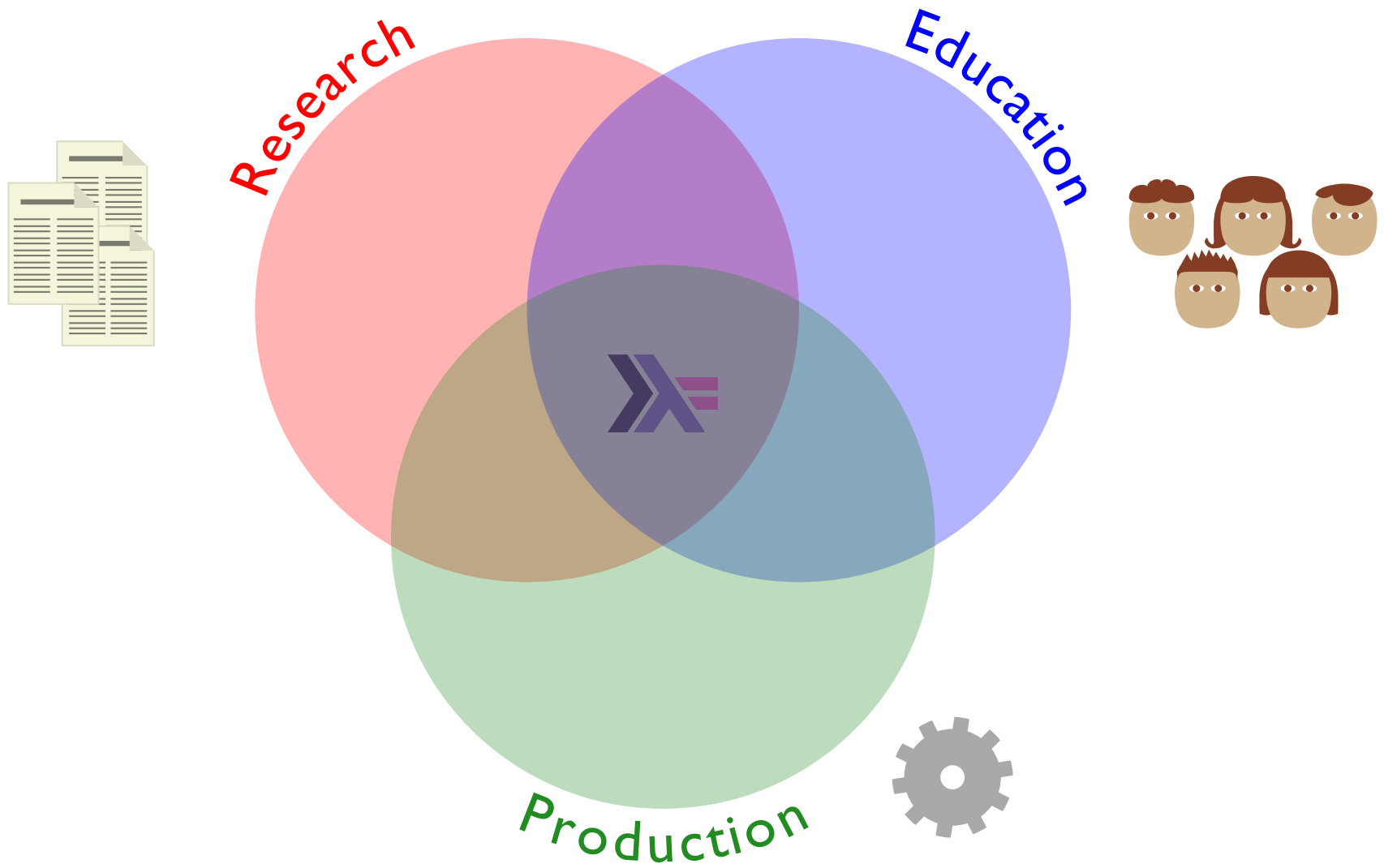


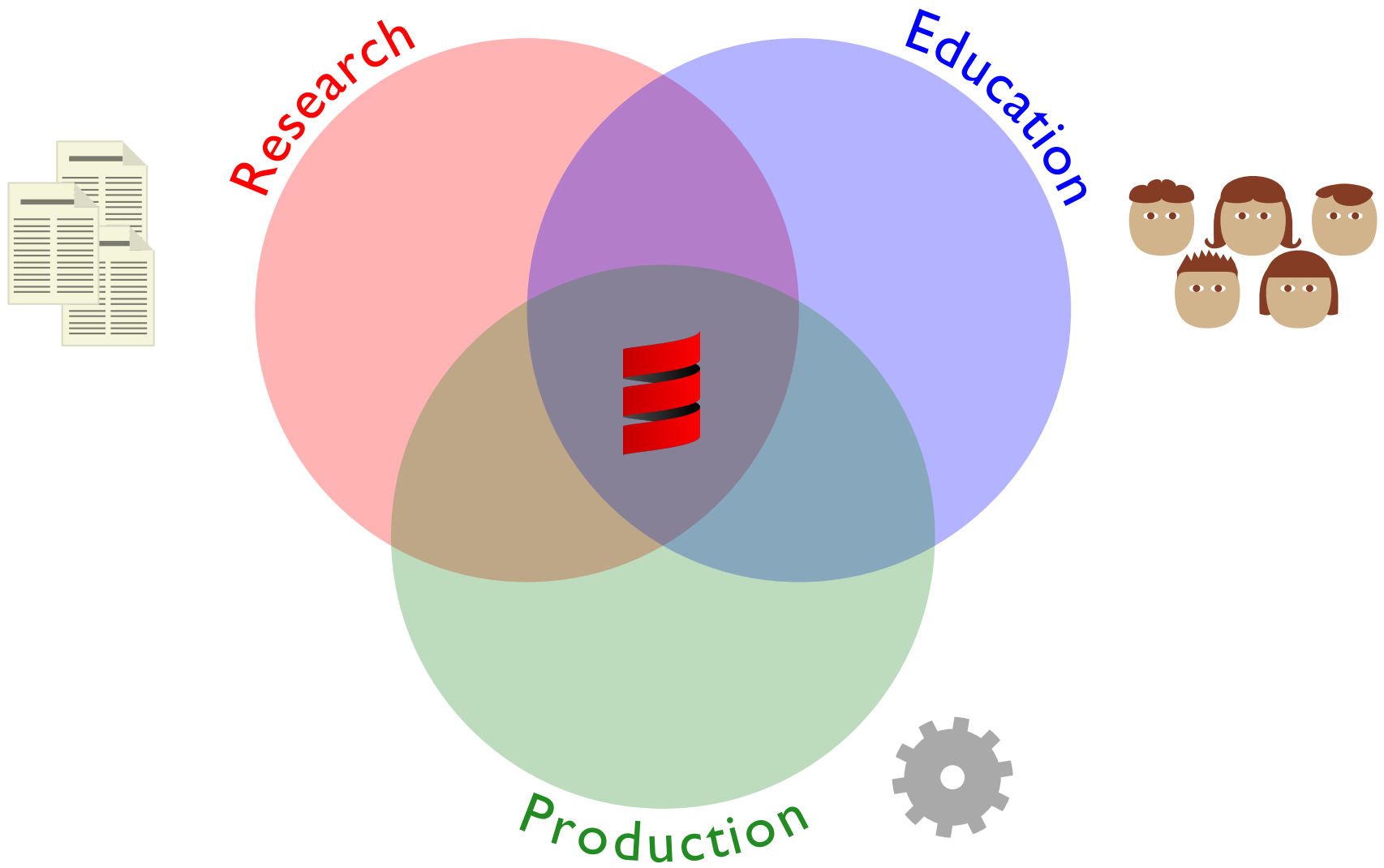
3

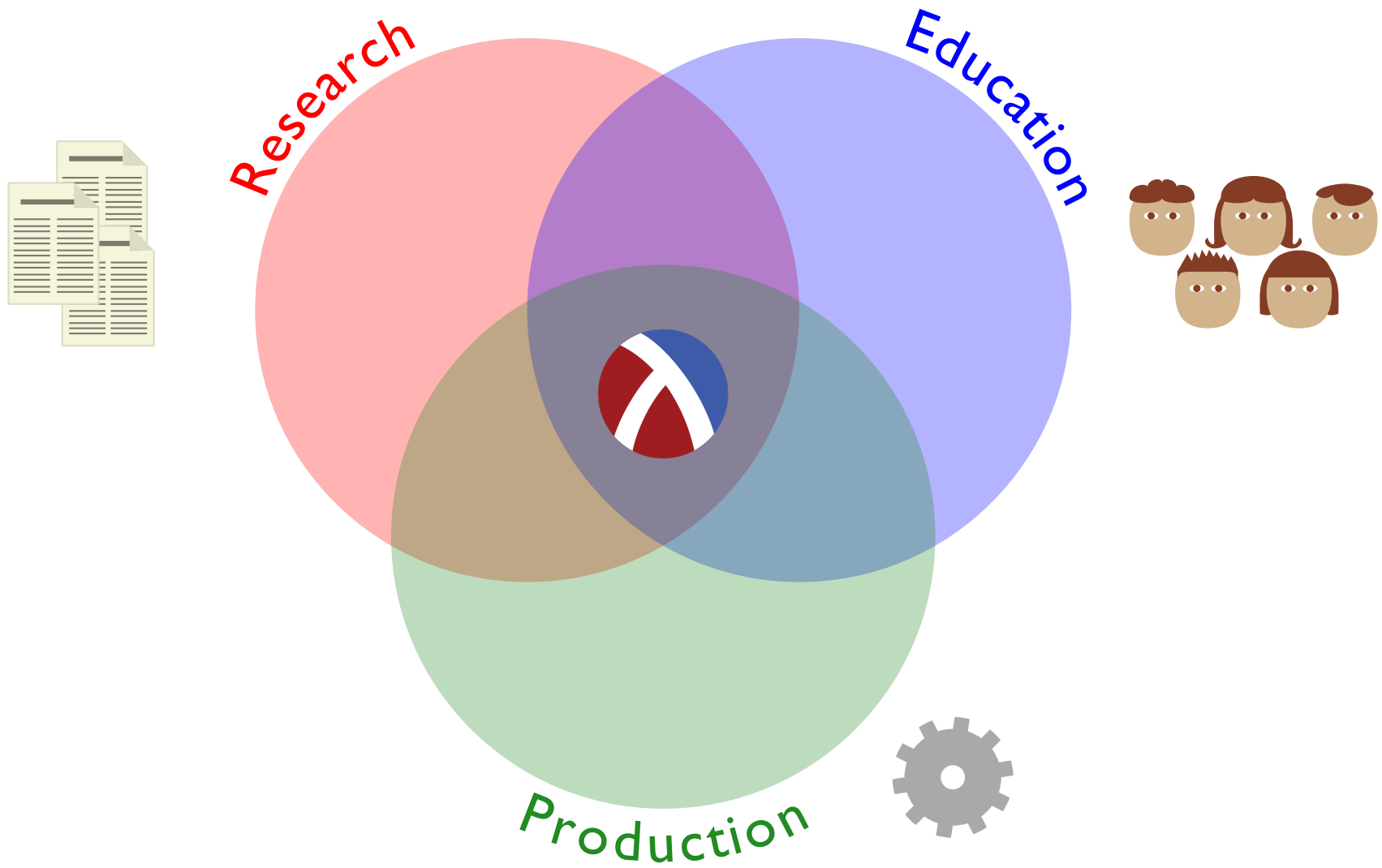


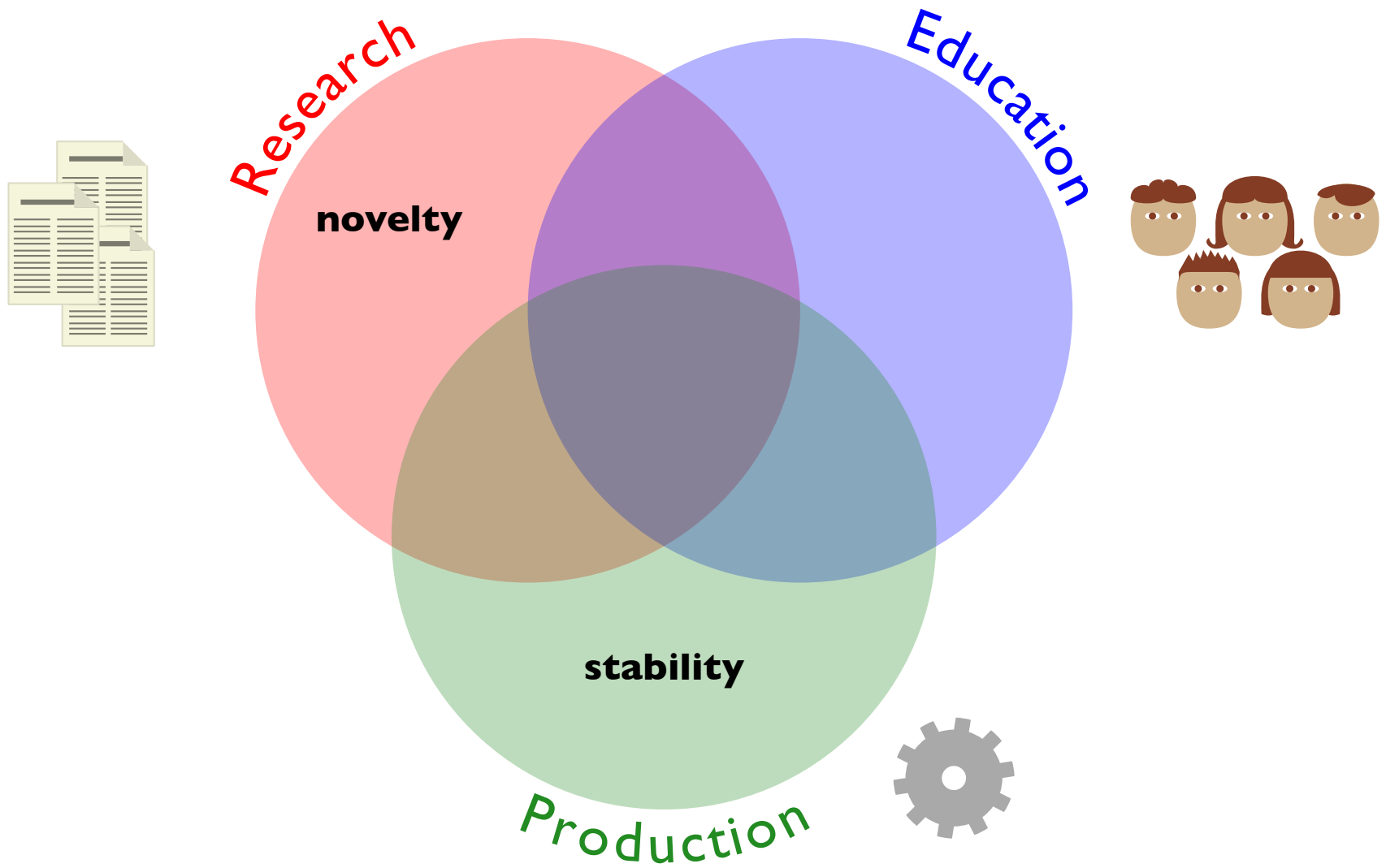


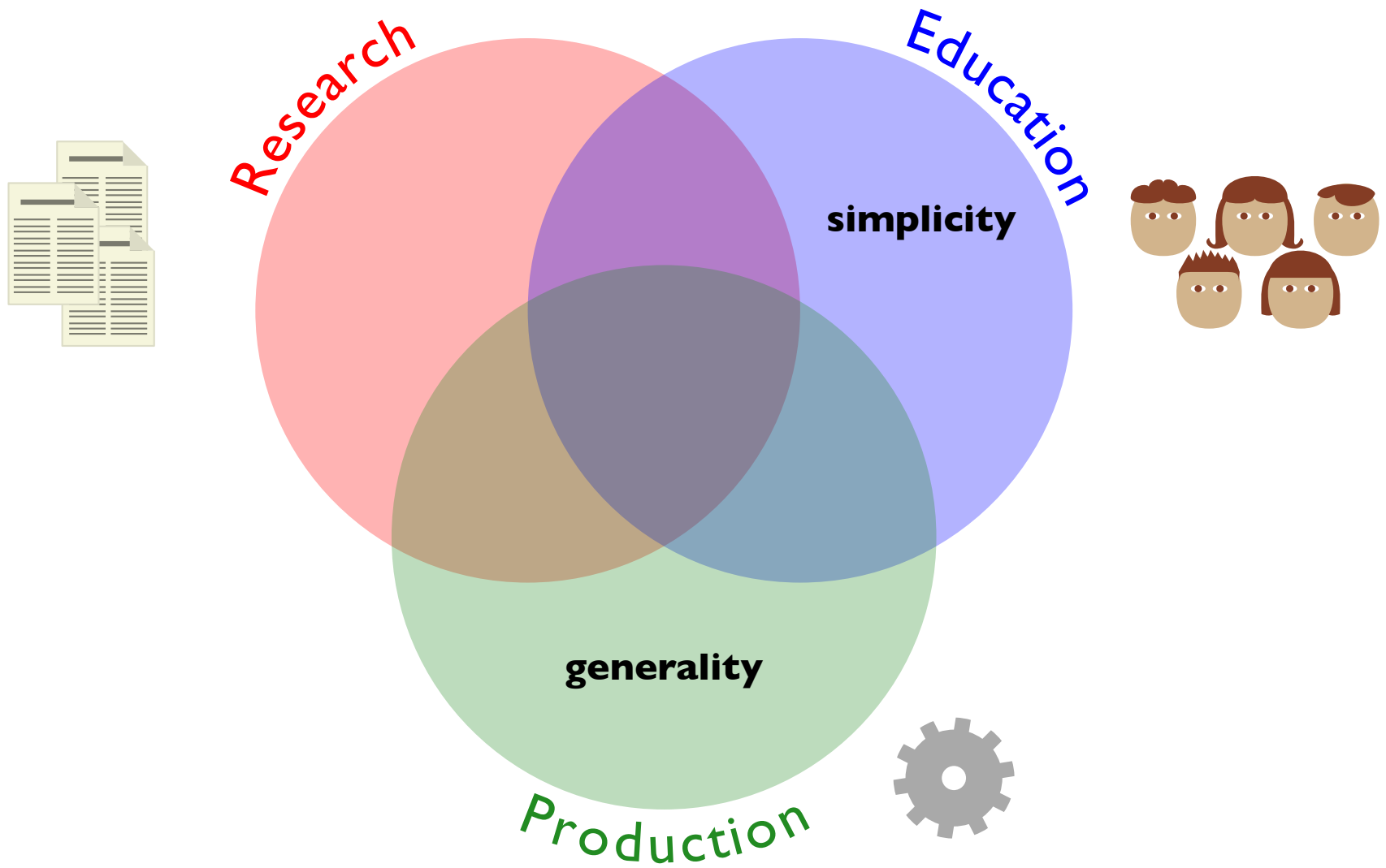


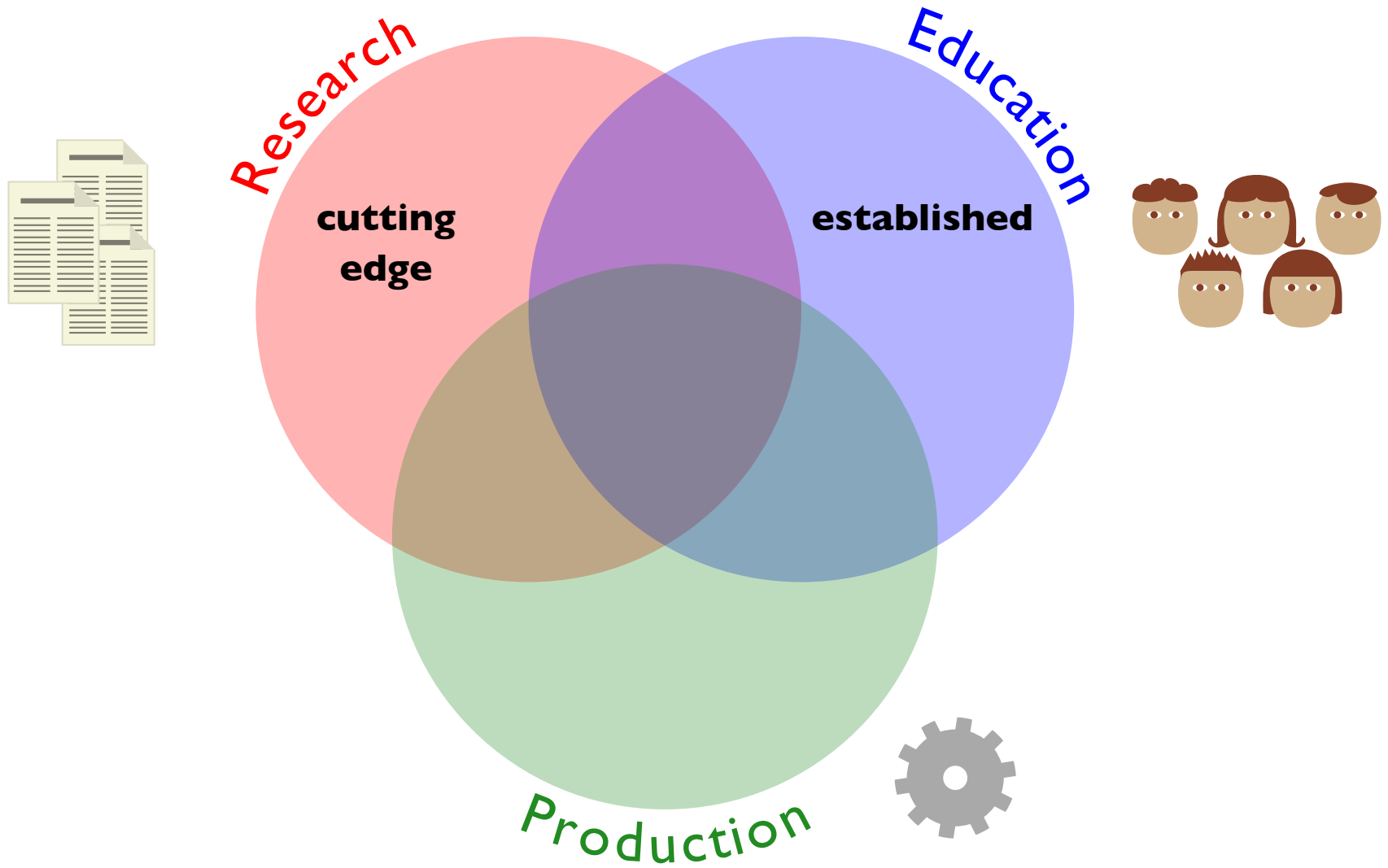


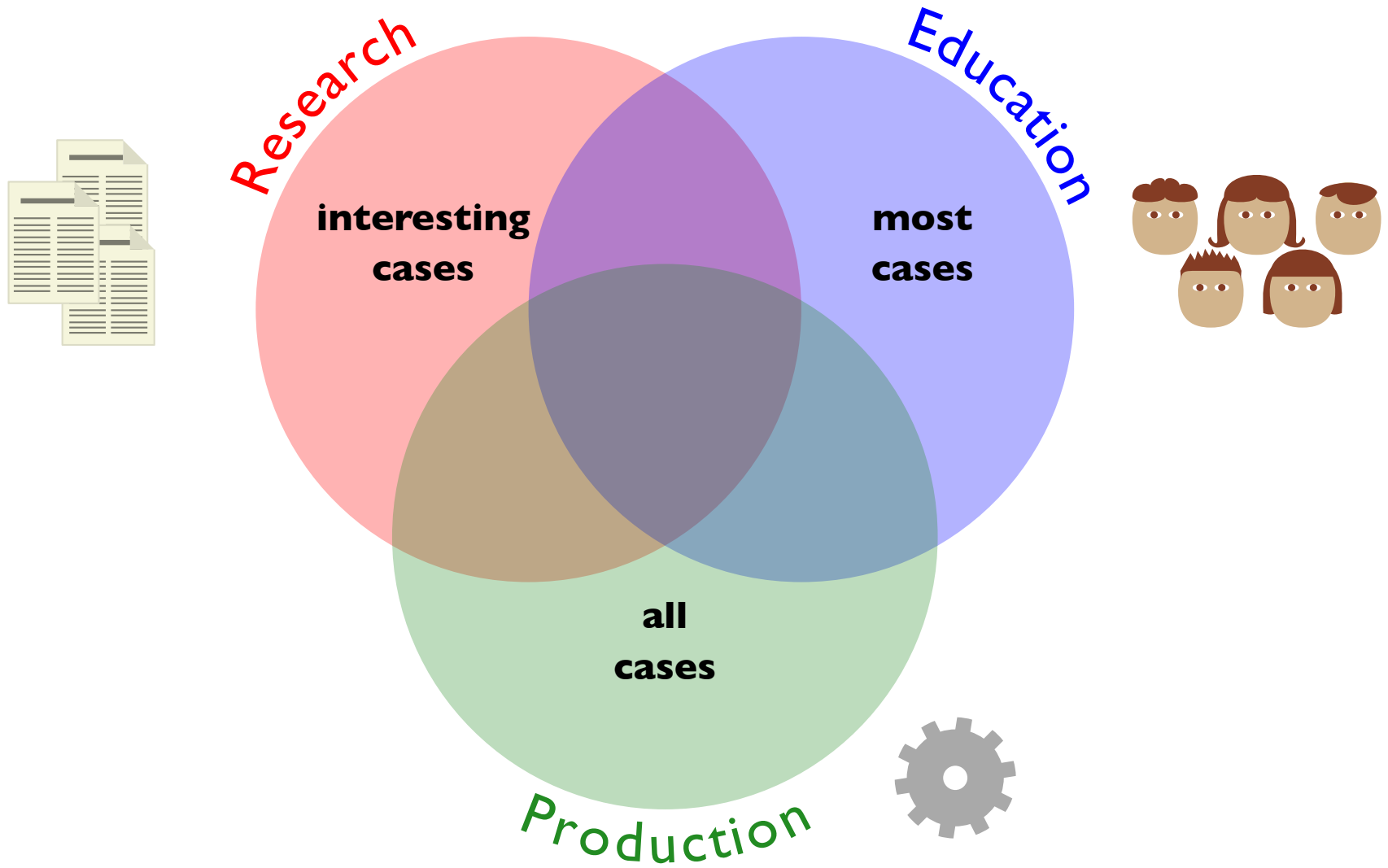


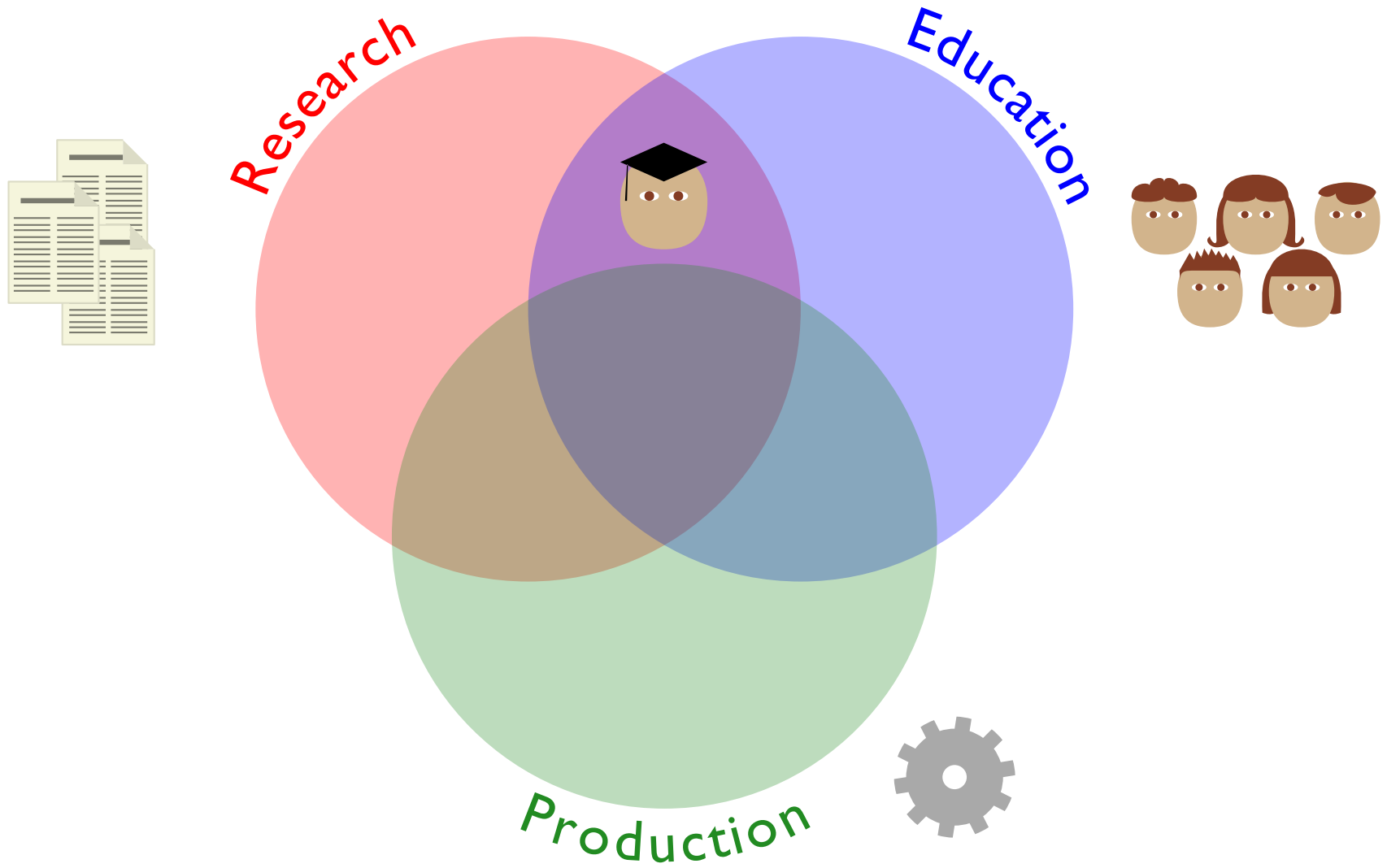


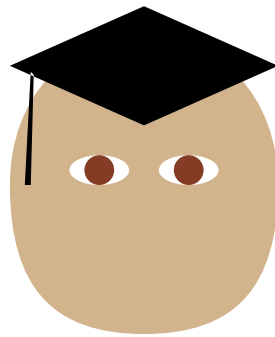




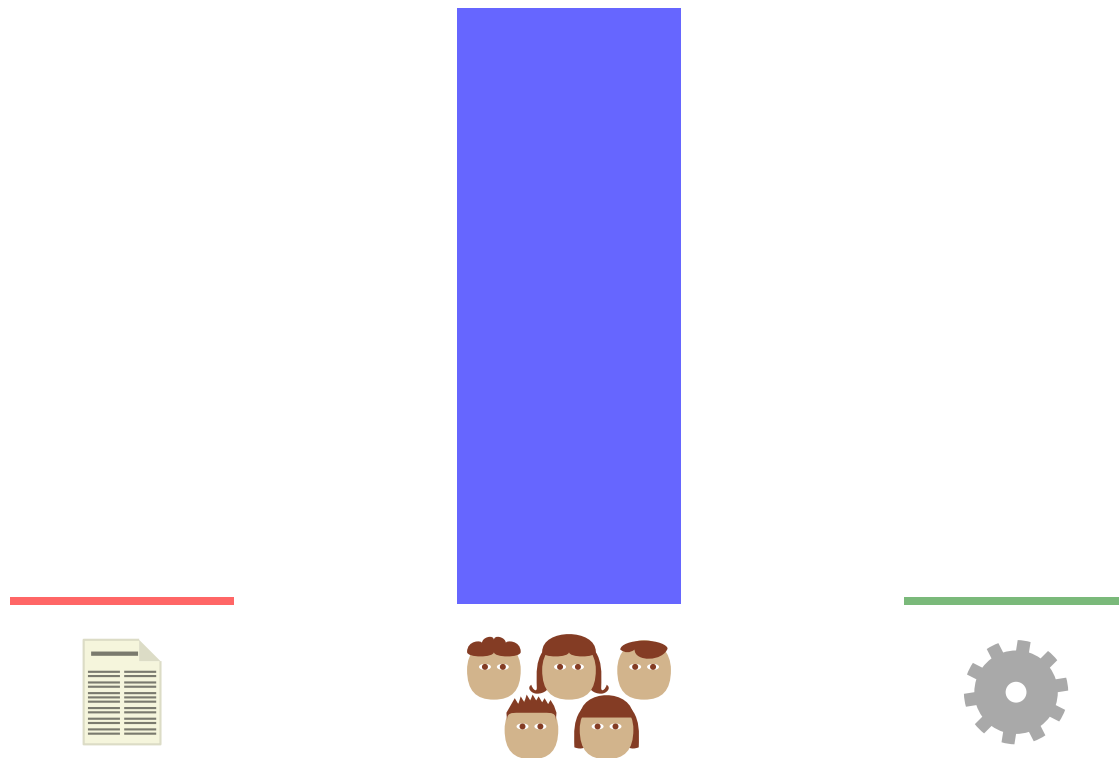




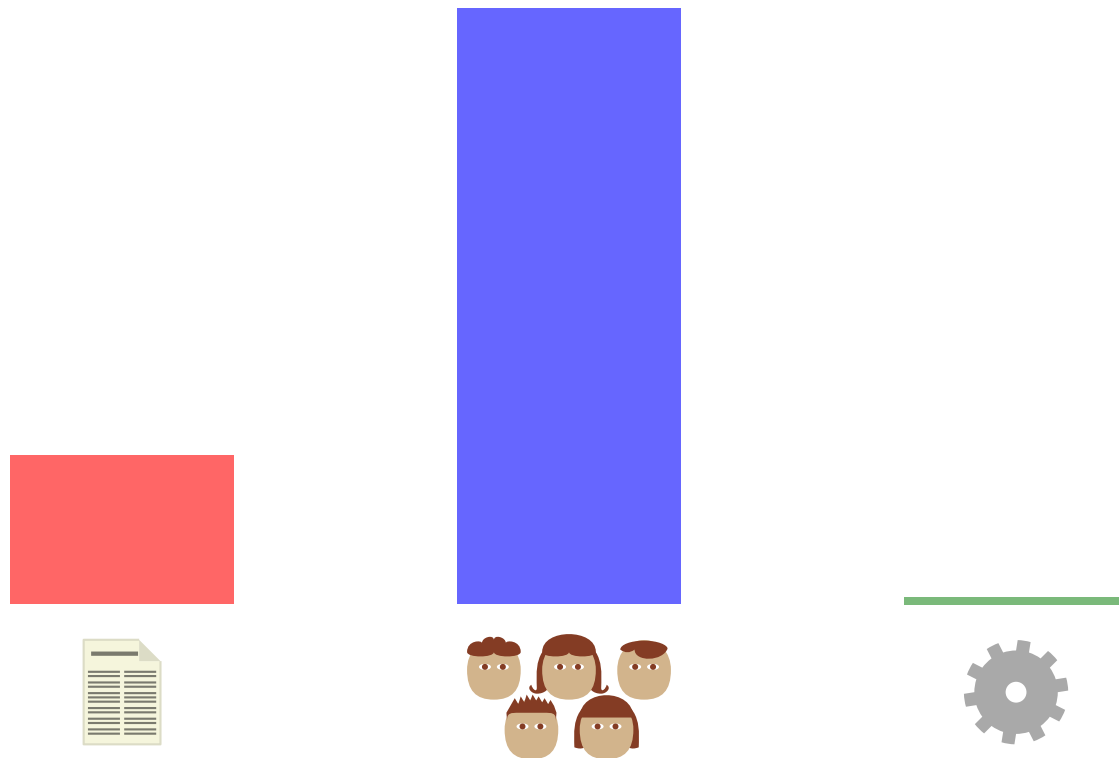




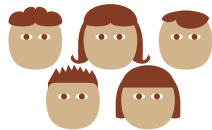
Citizen View of Professors



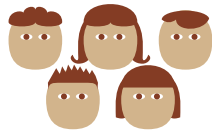
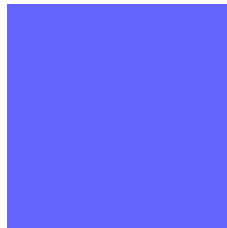
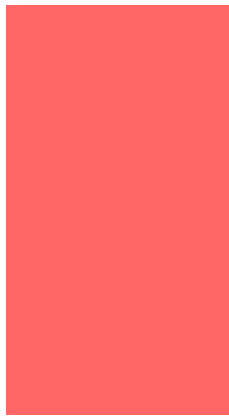
Citizen View of Professors



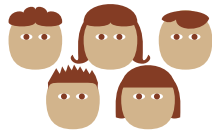
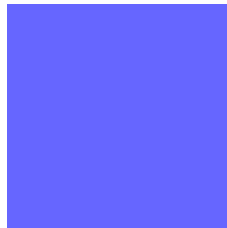
Citizen View of Professors



Department View of Professors

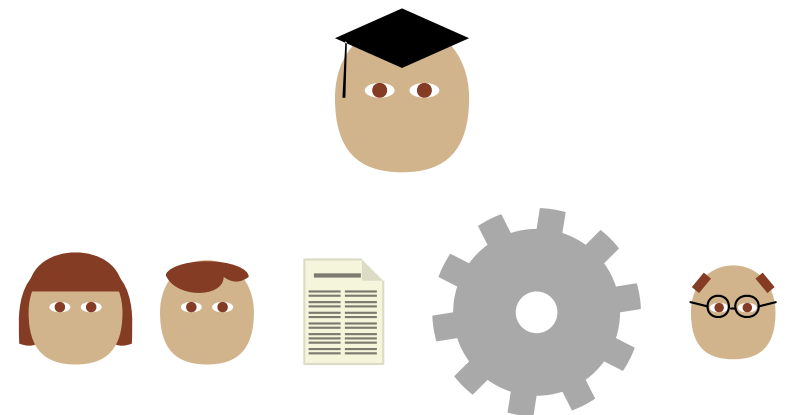
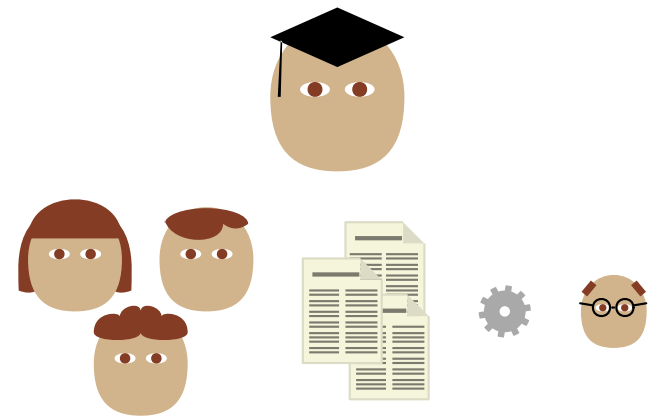
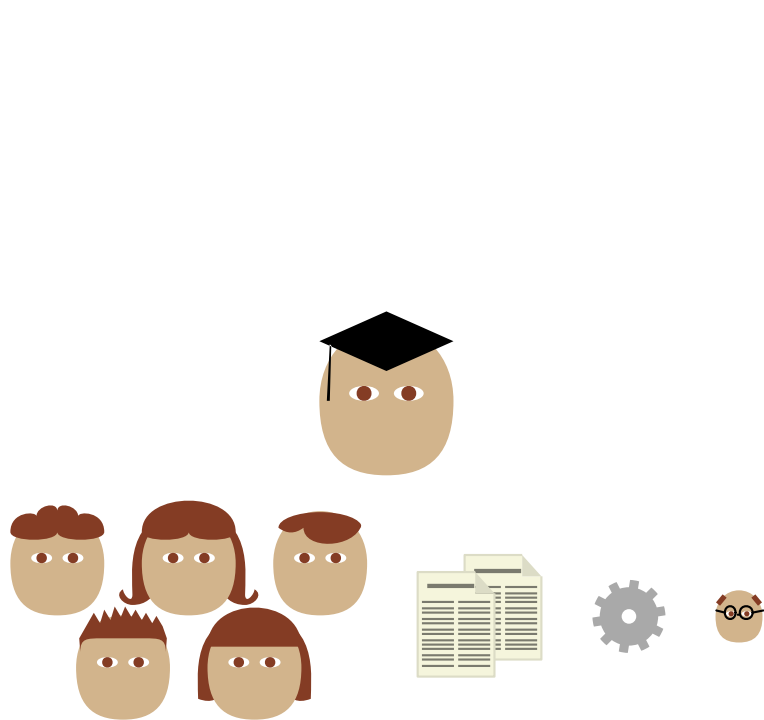


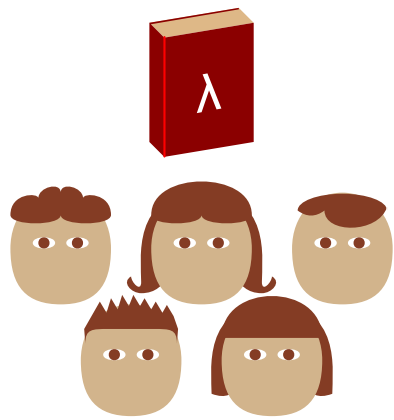
My Reality 🎓

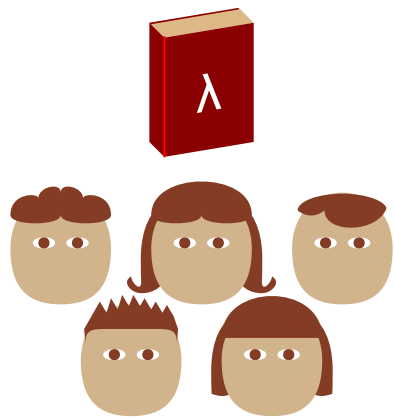


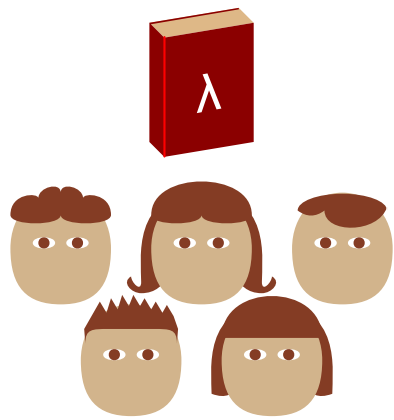
My Reality 🎓

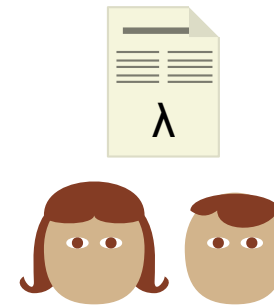
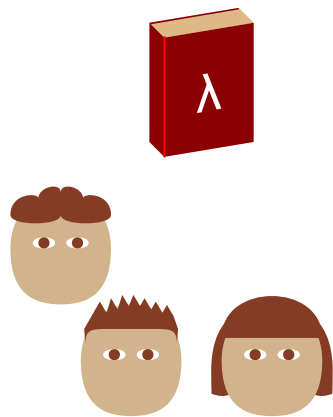


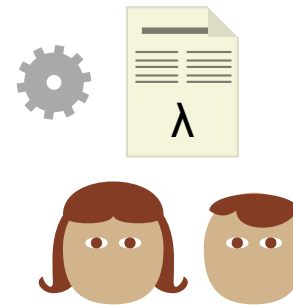
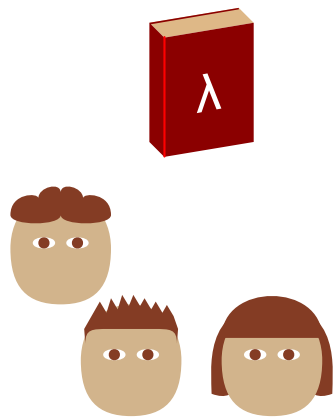


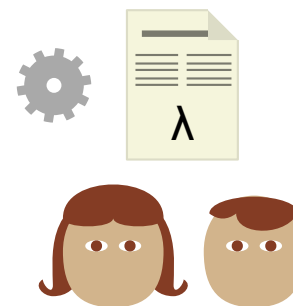
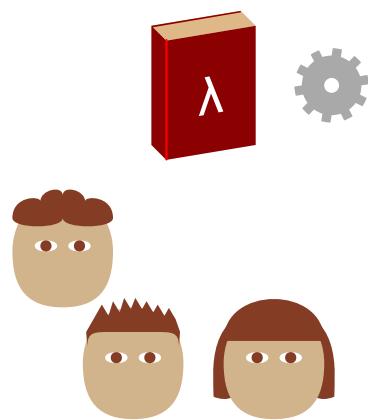




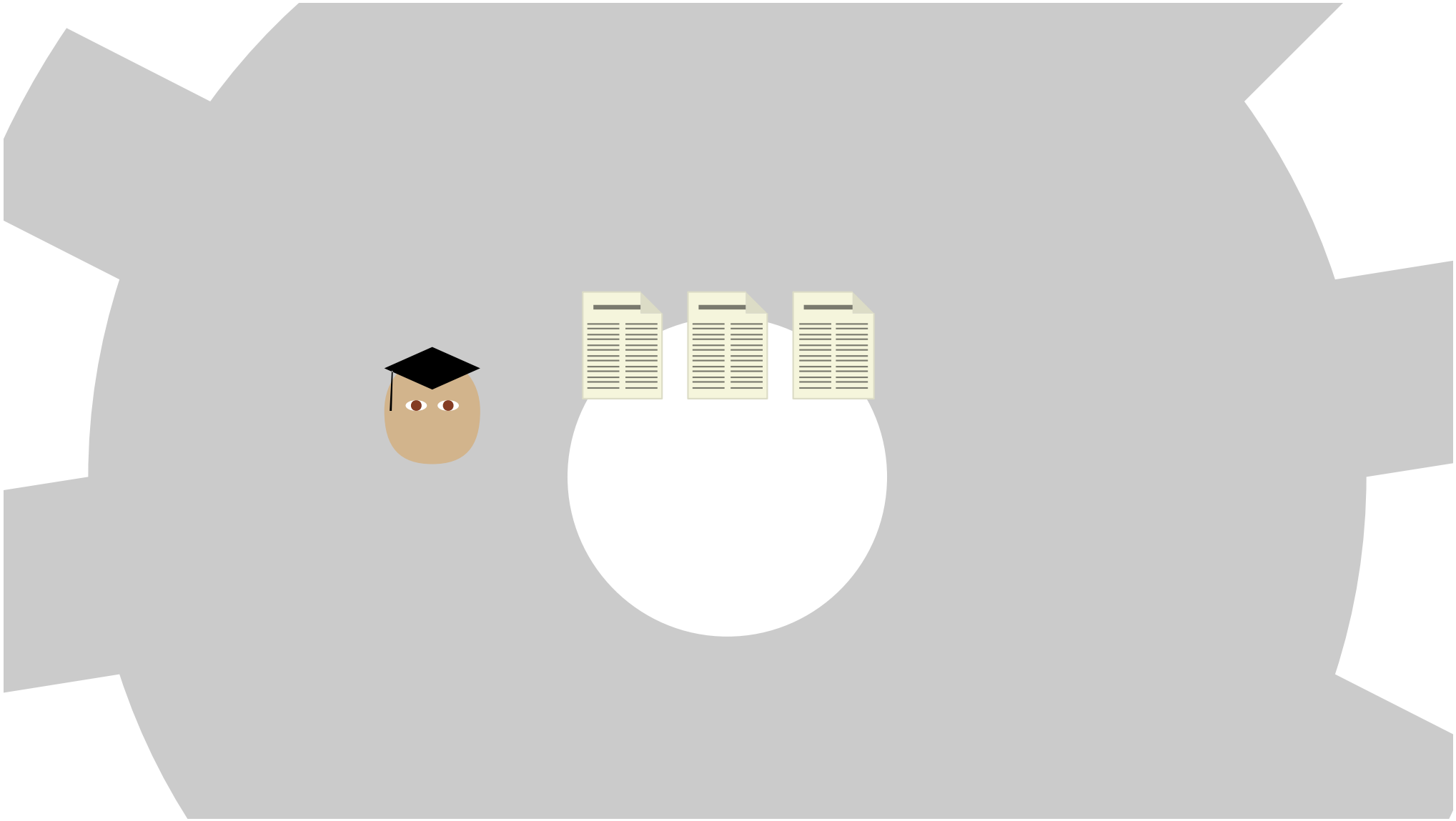


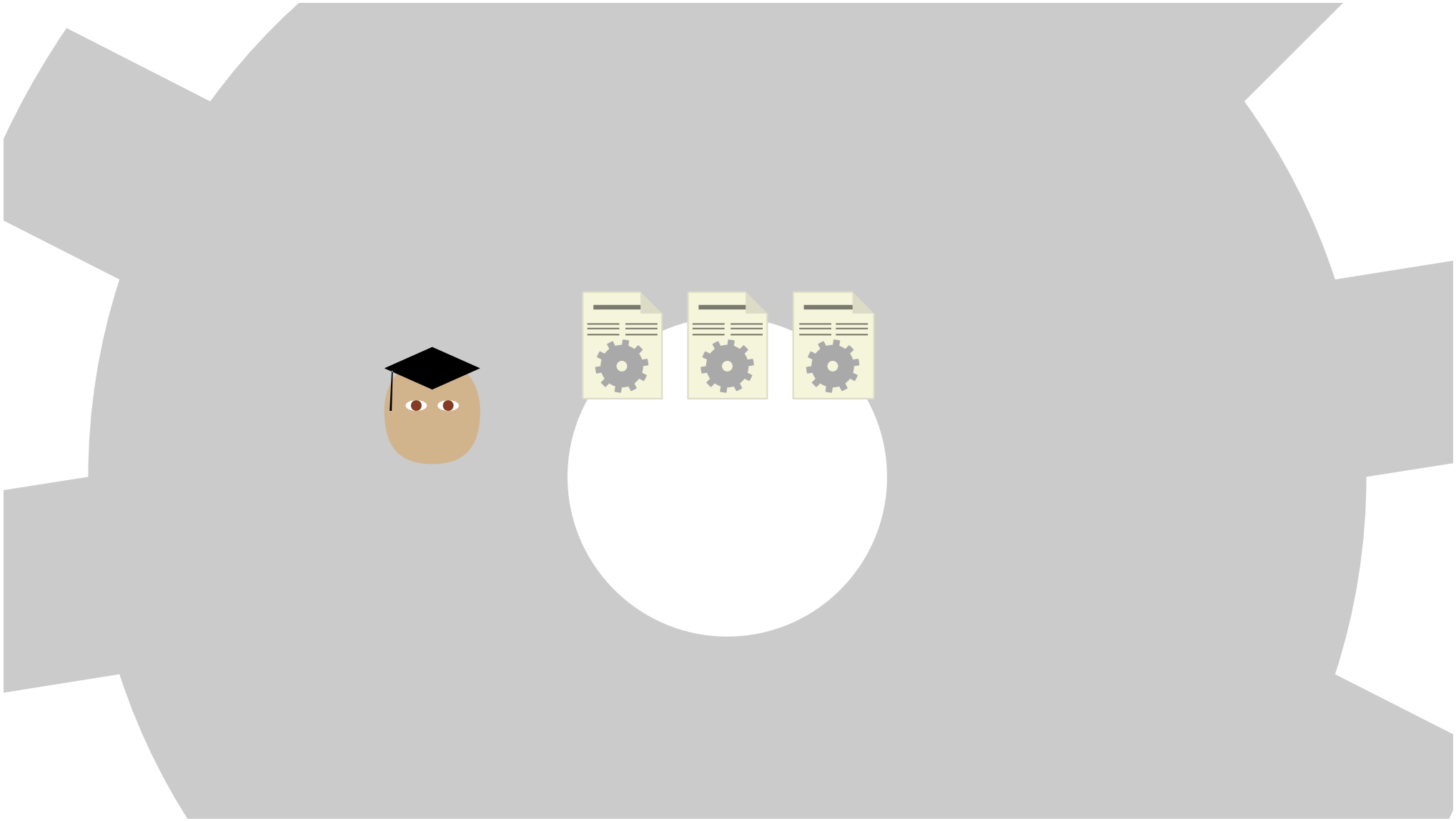




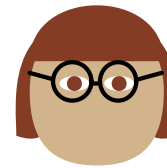


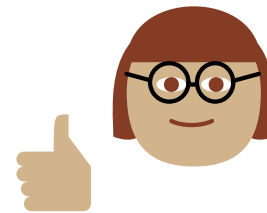


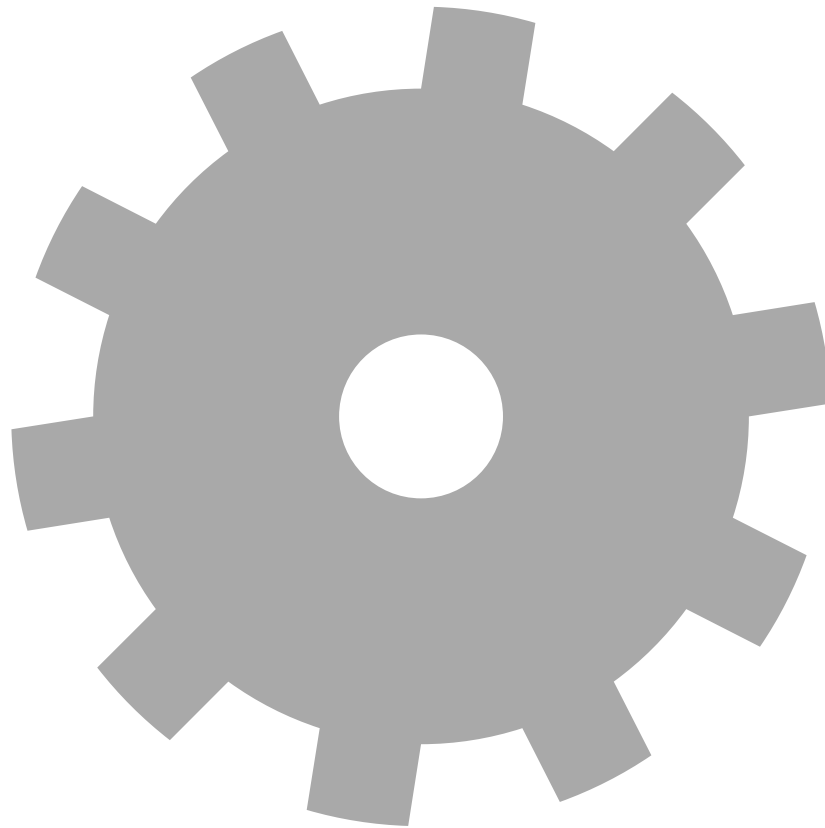
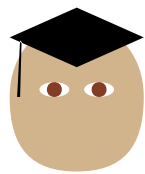


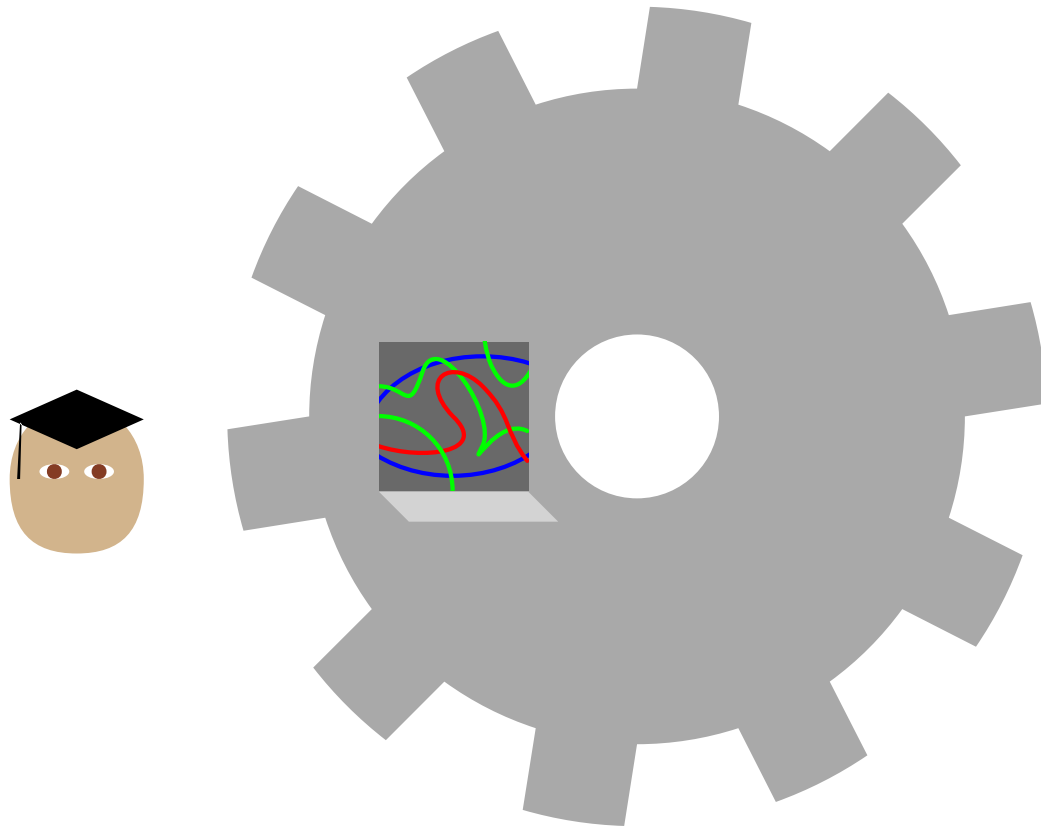


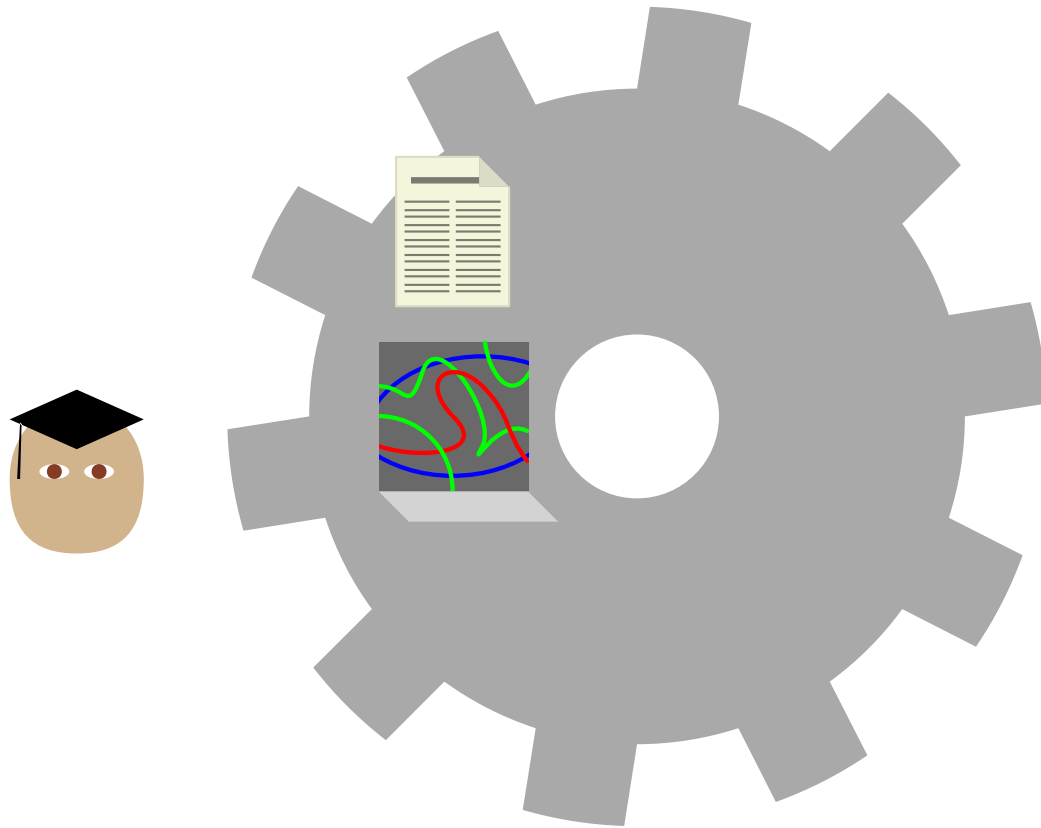


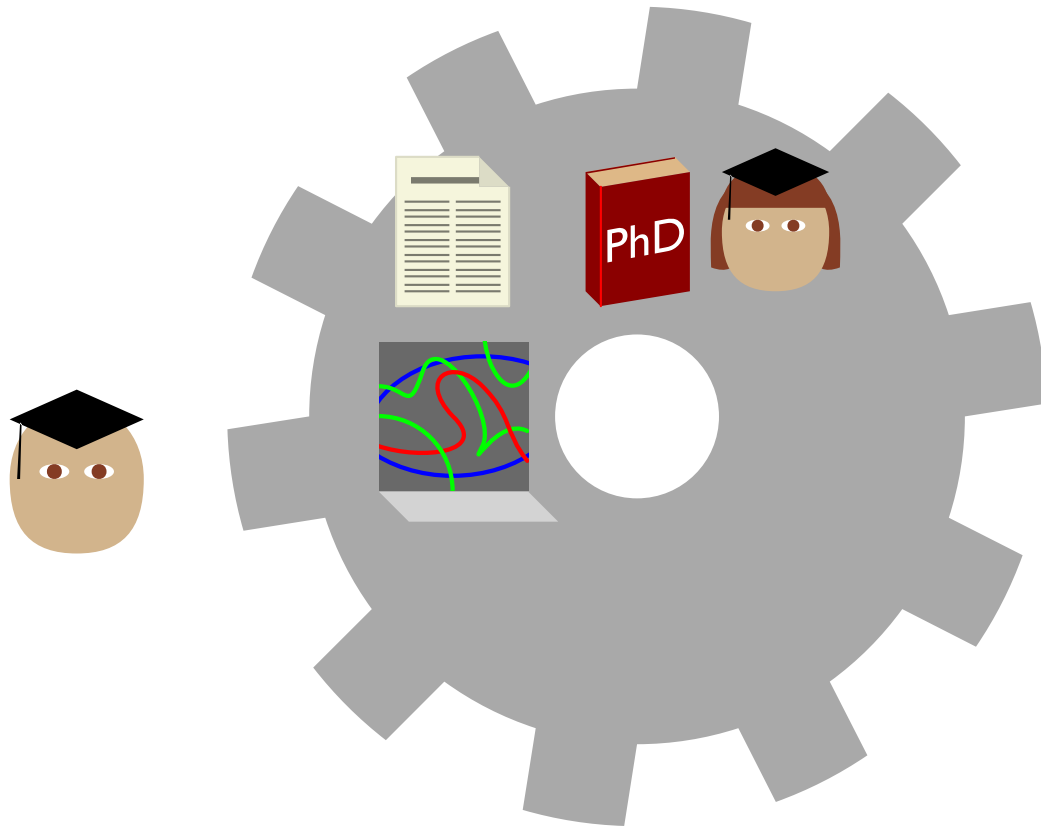


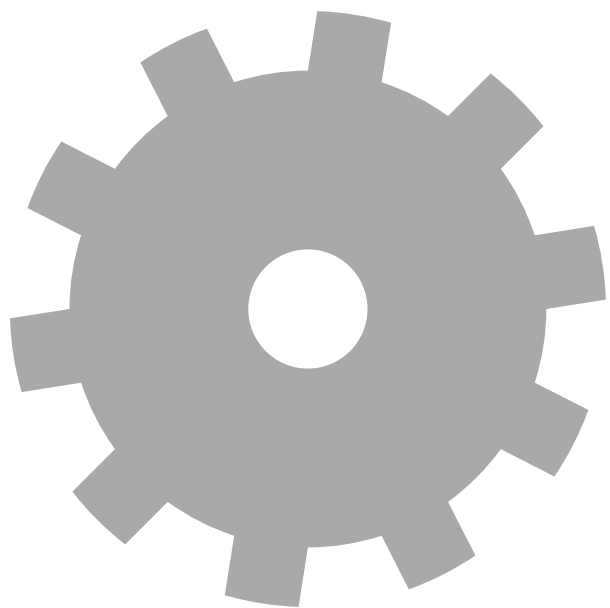


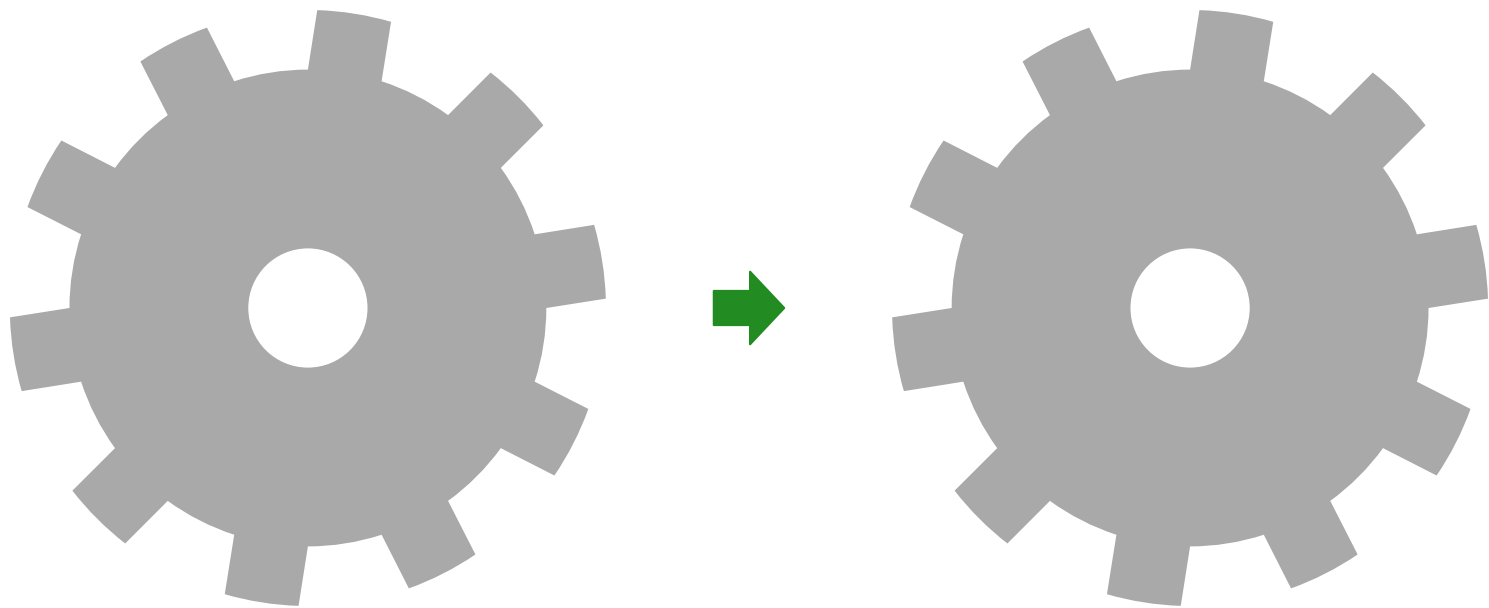


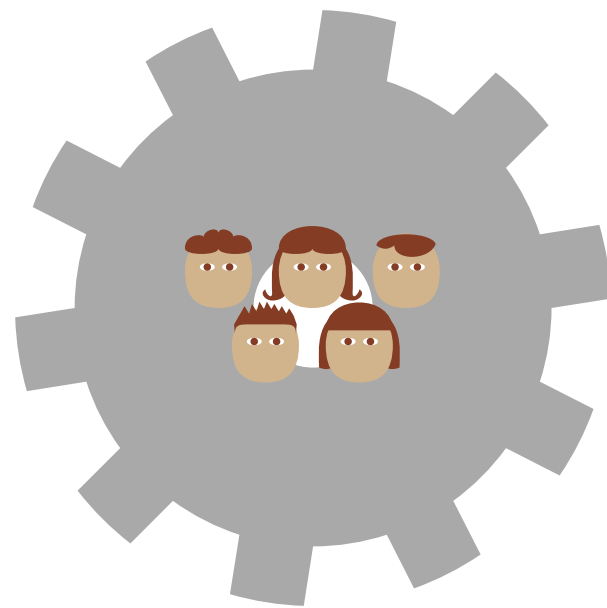
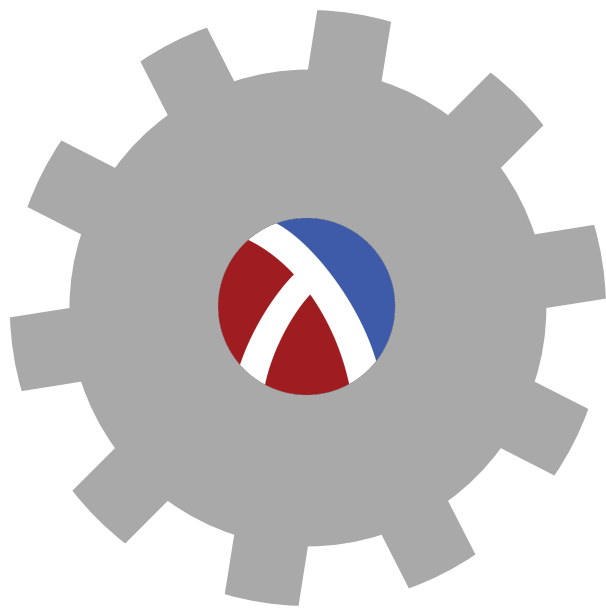


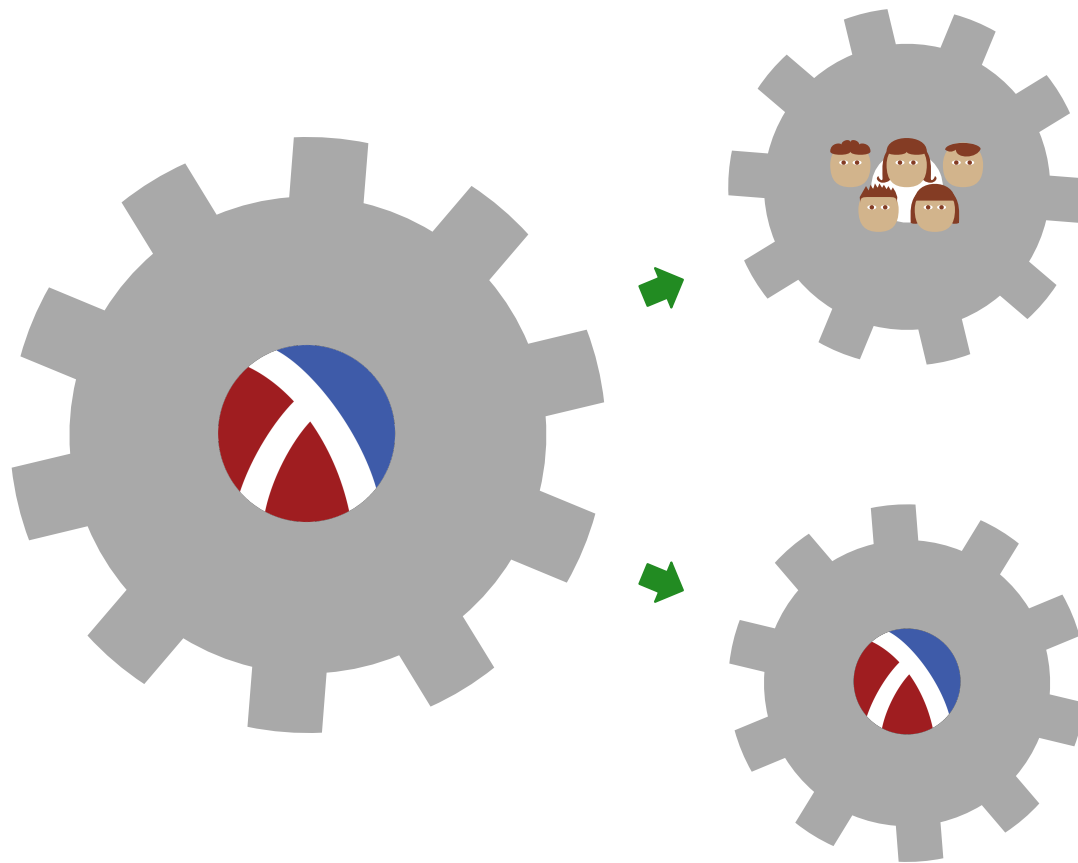


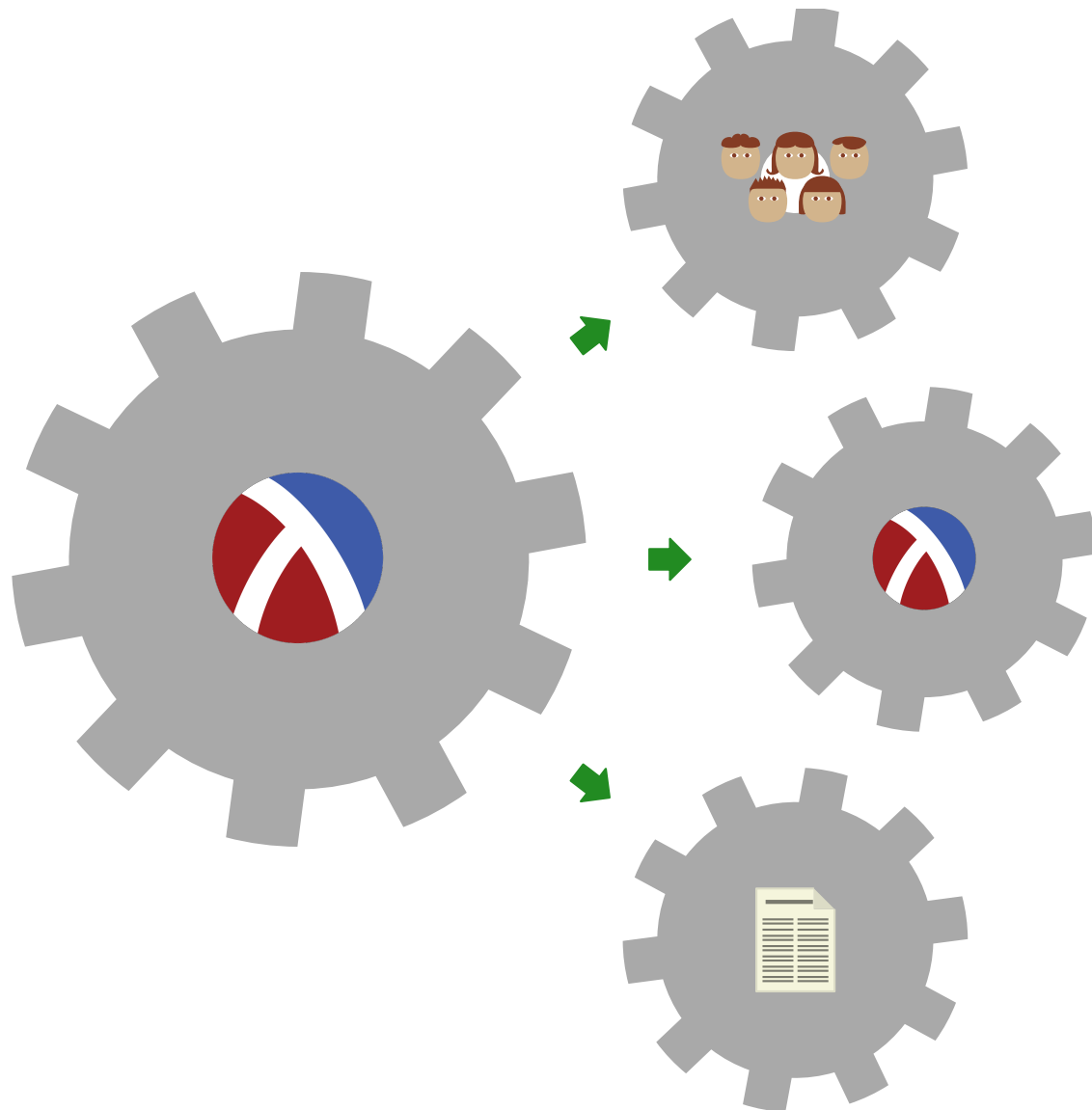


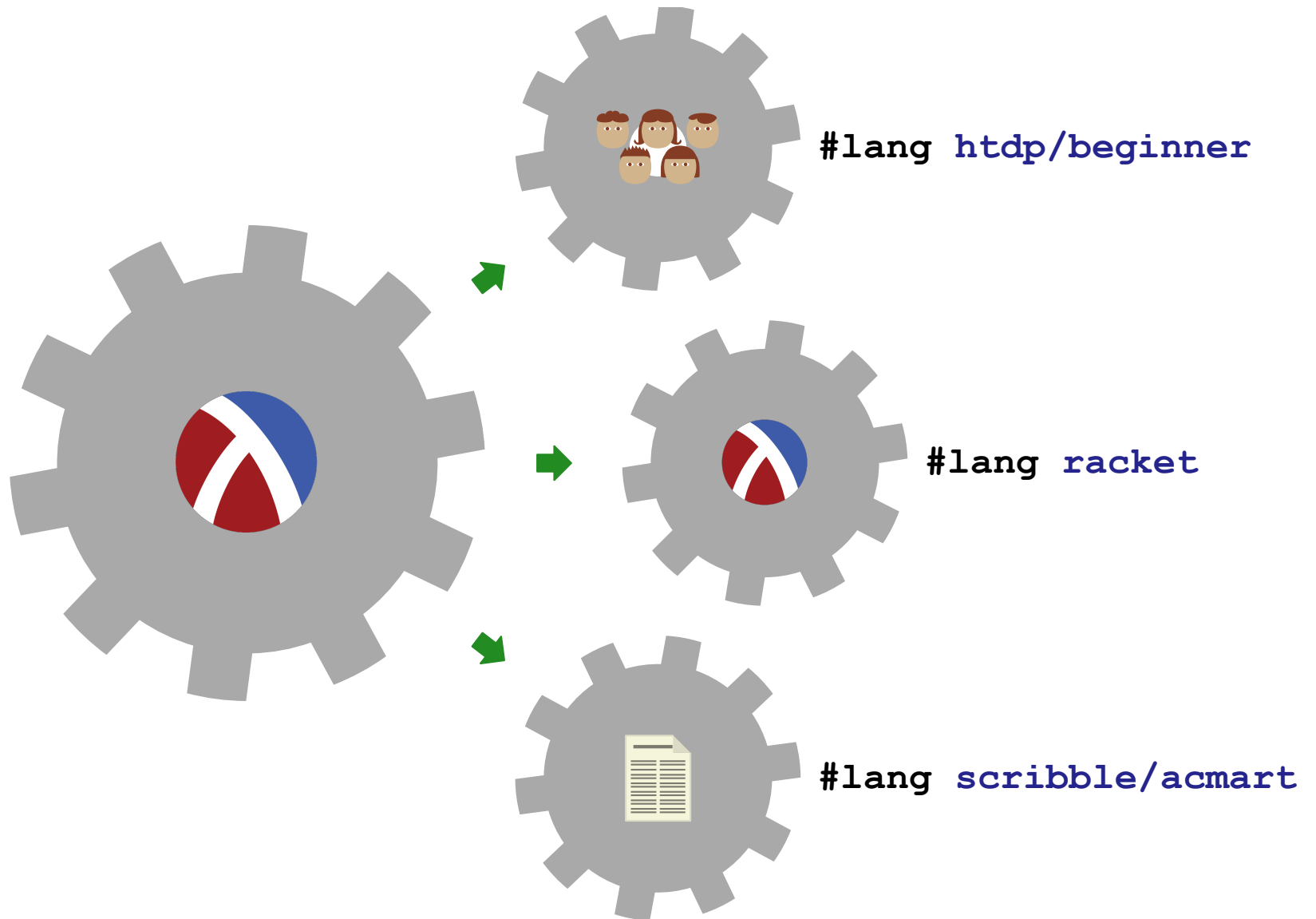


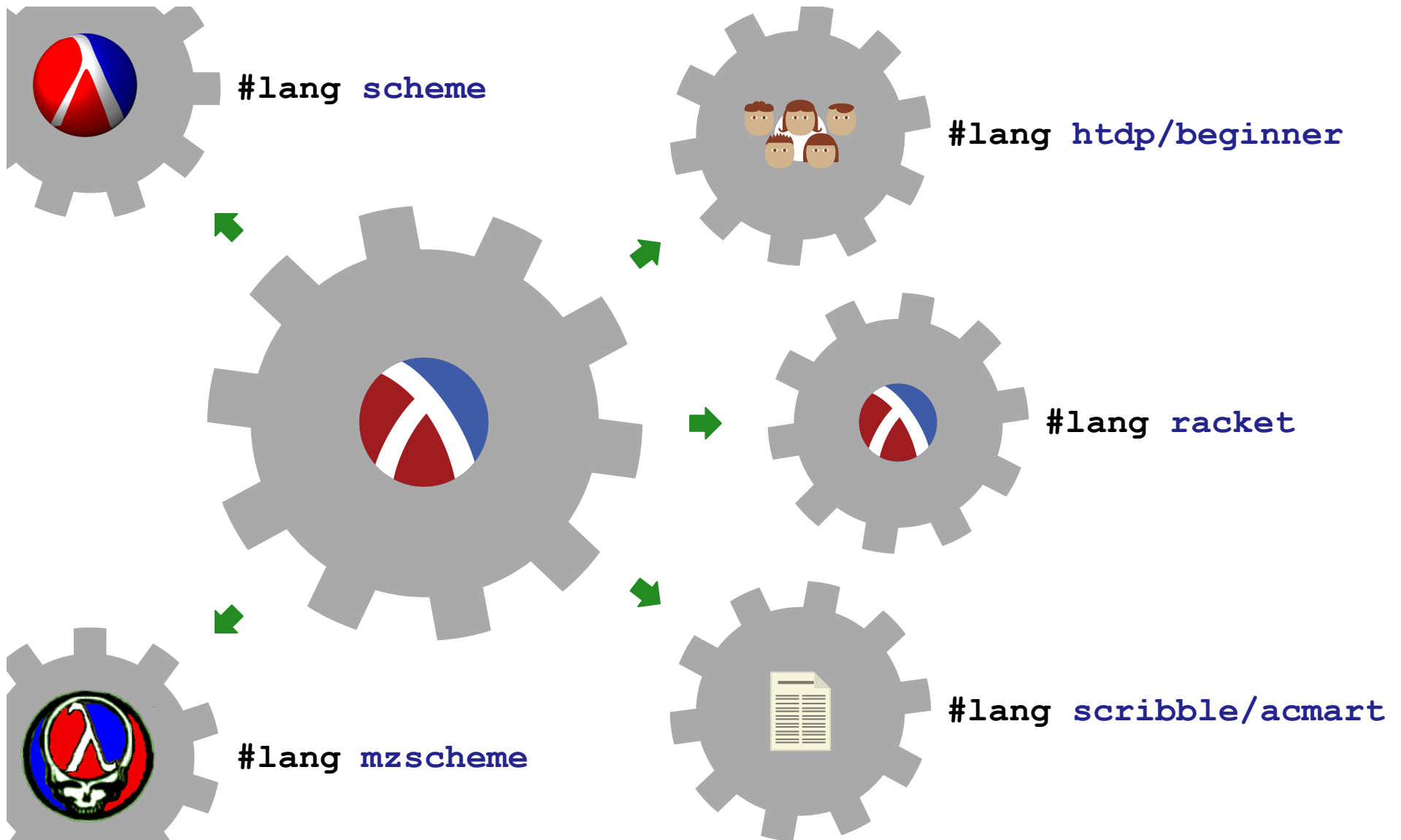


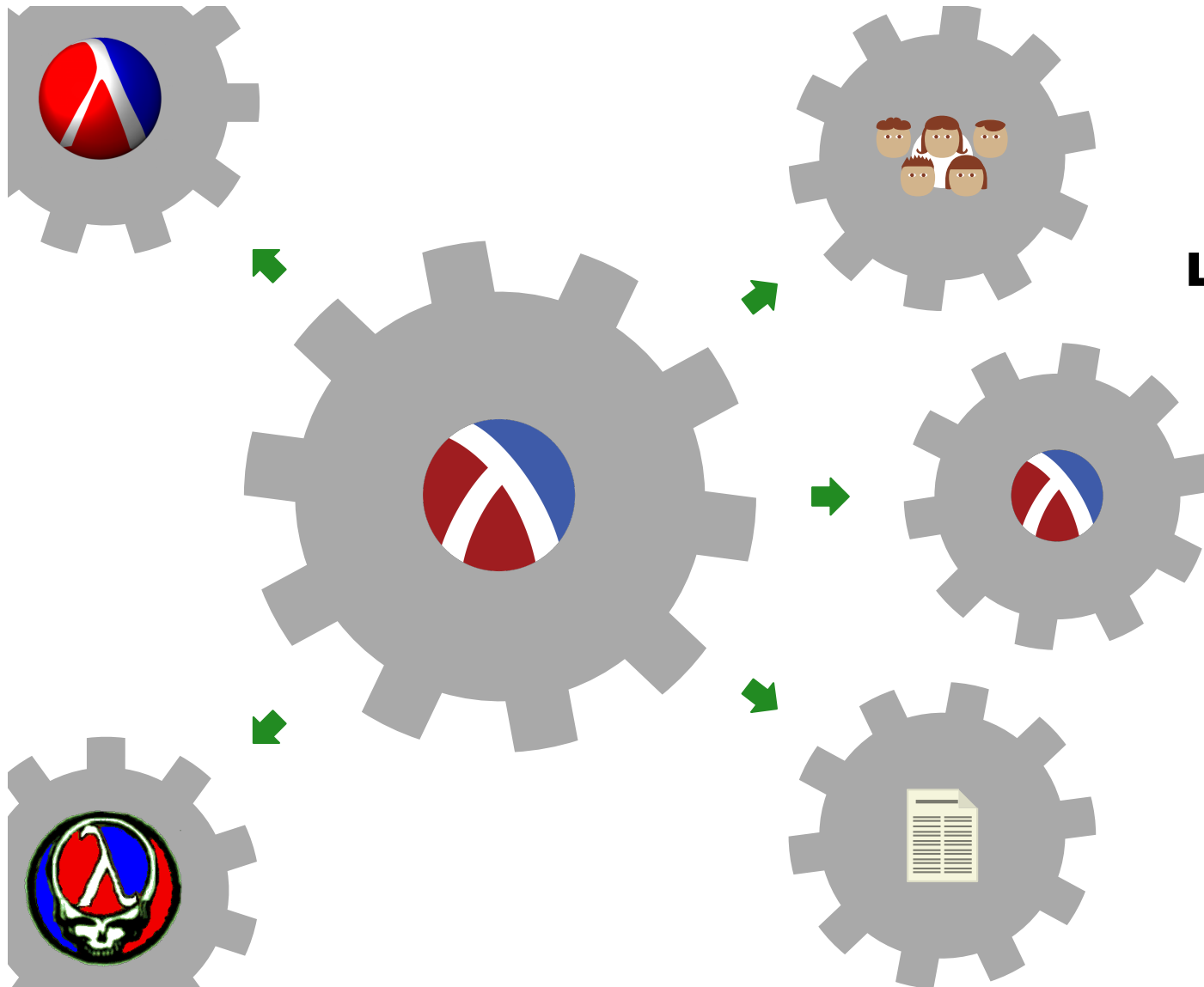




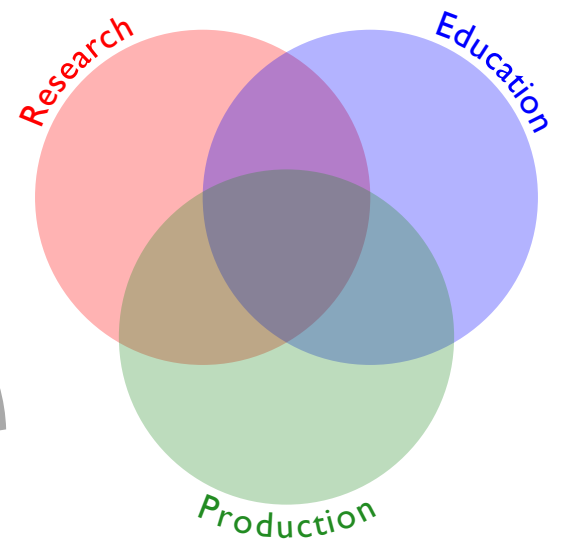
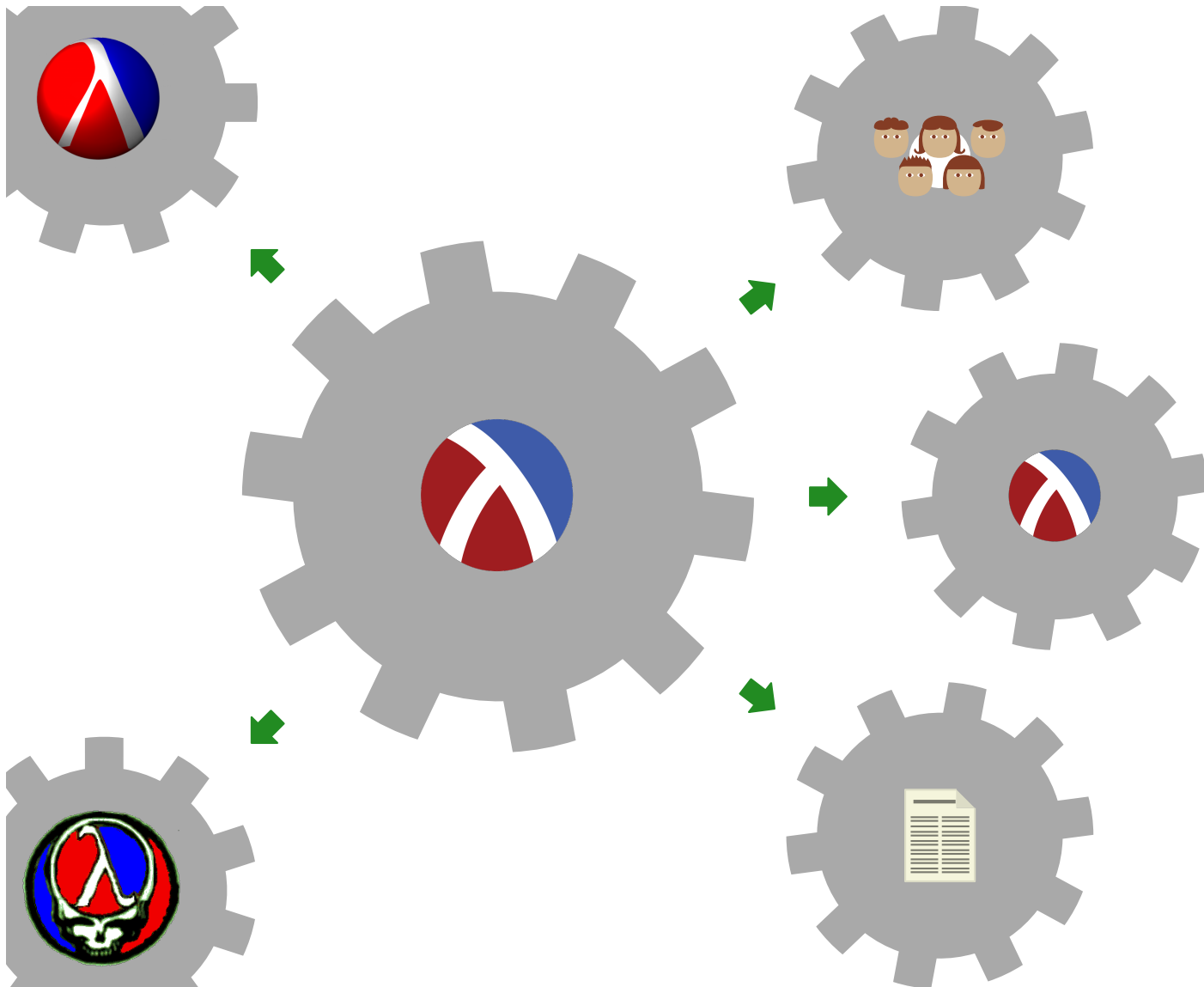


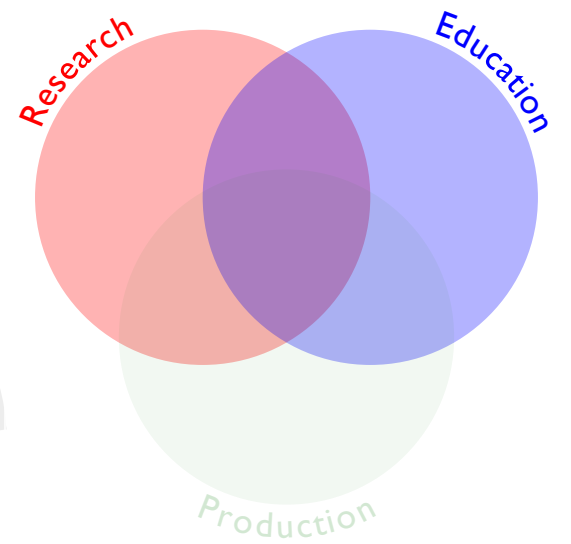
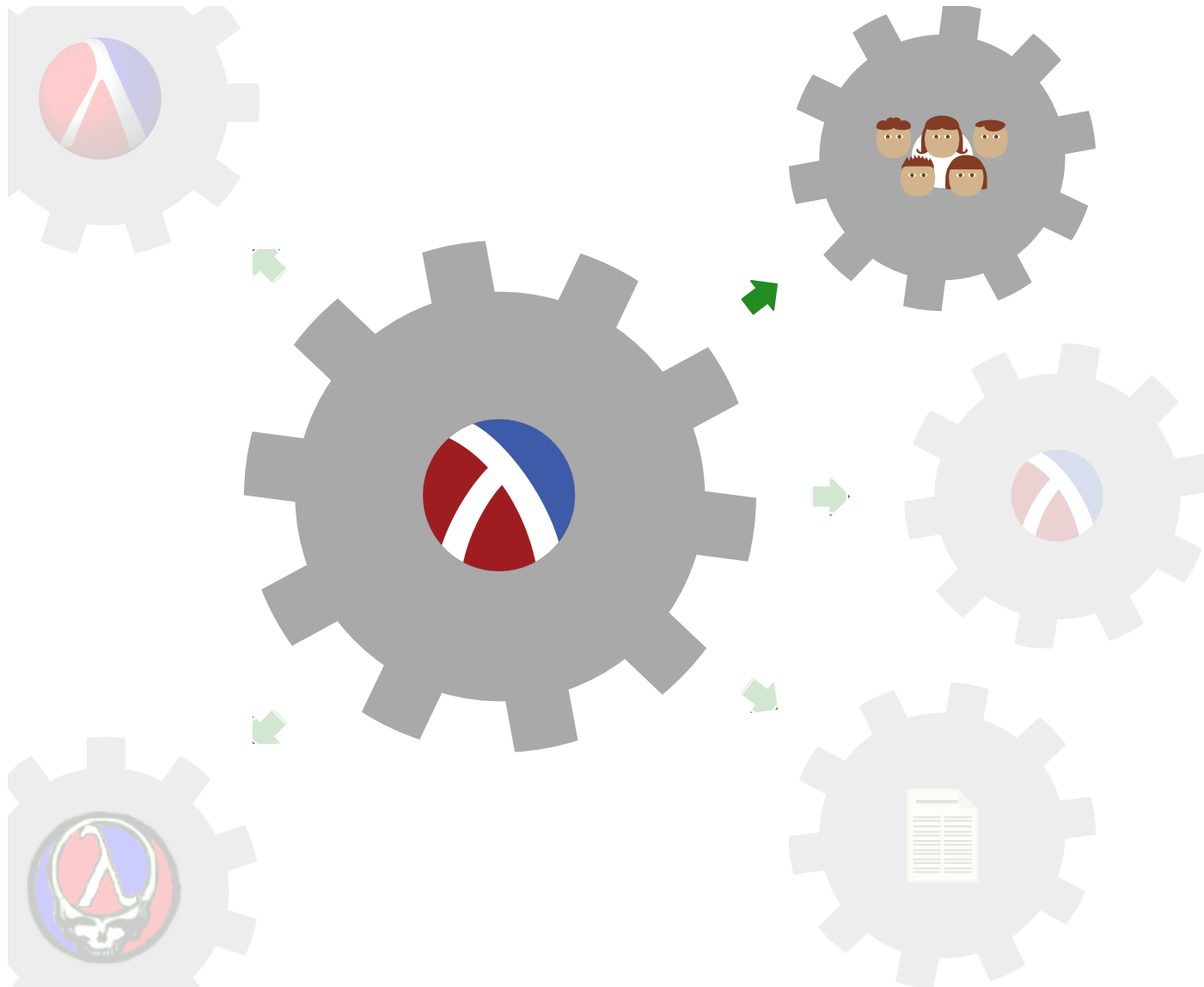


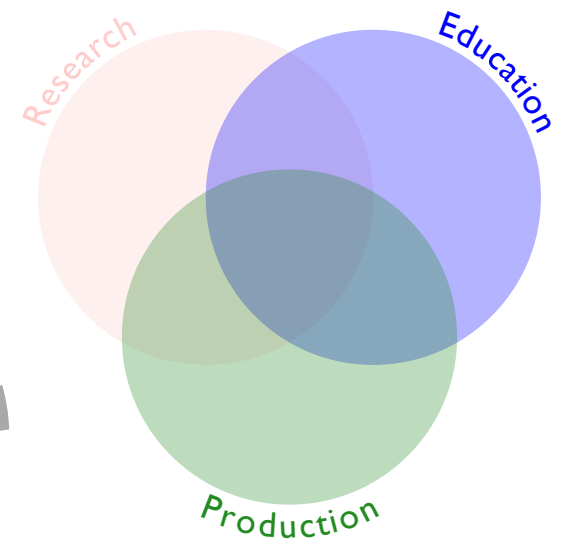
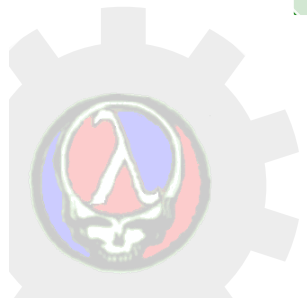
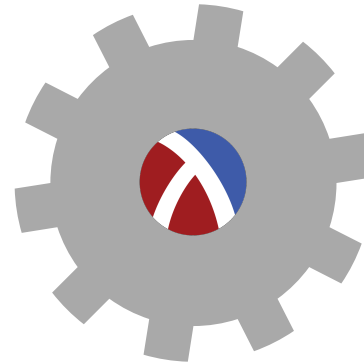
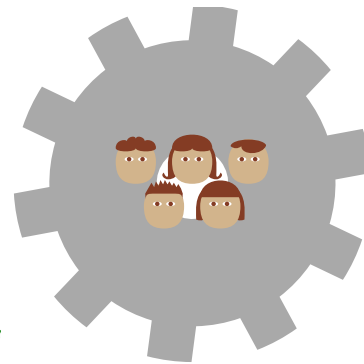
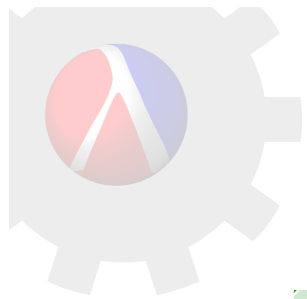


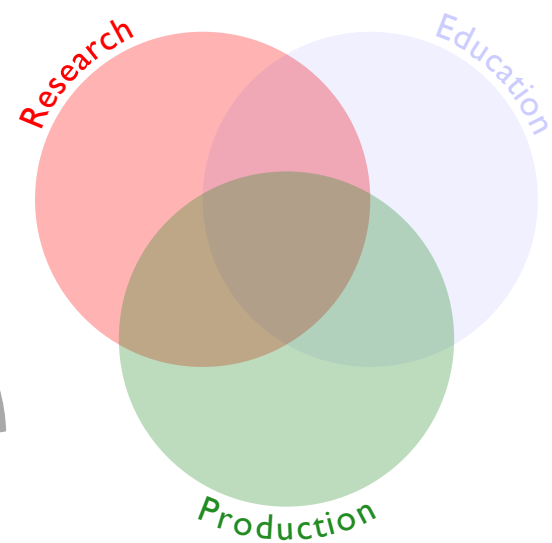
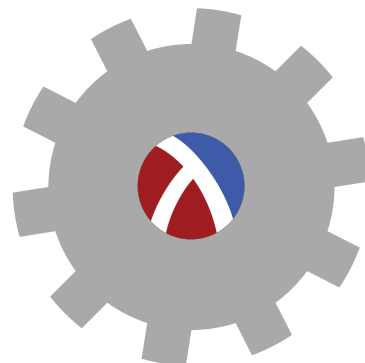
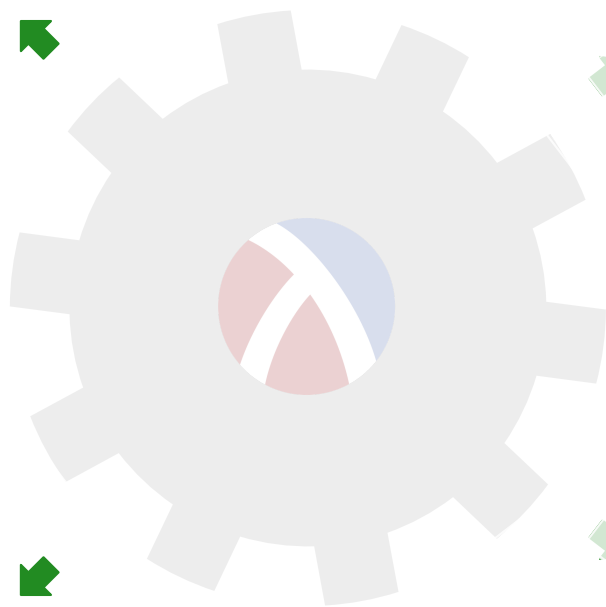
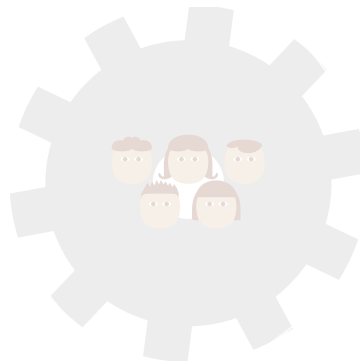
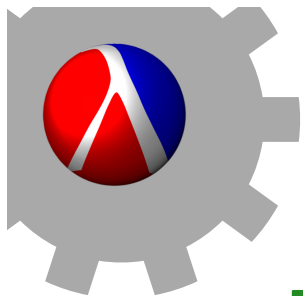


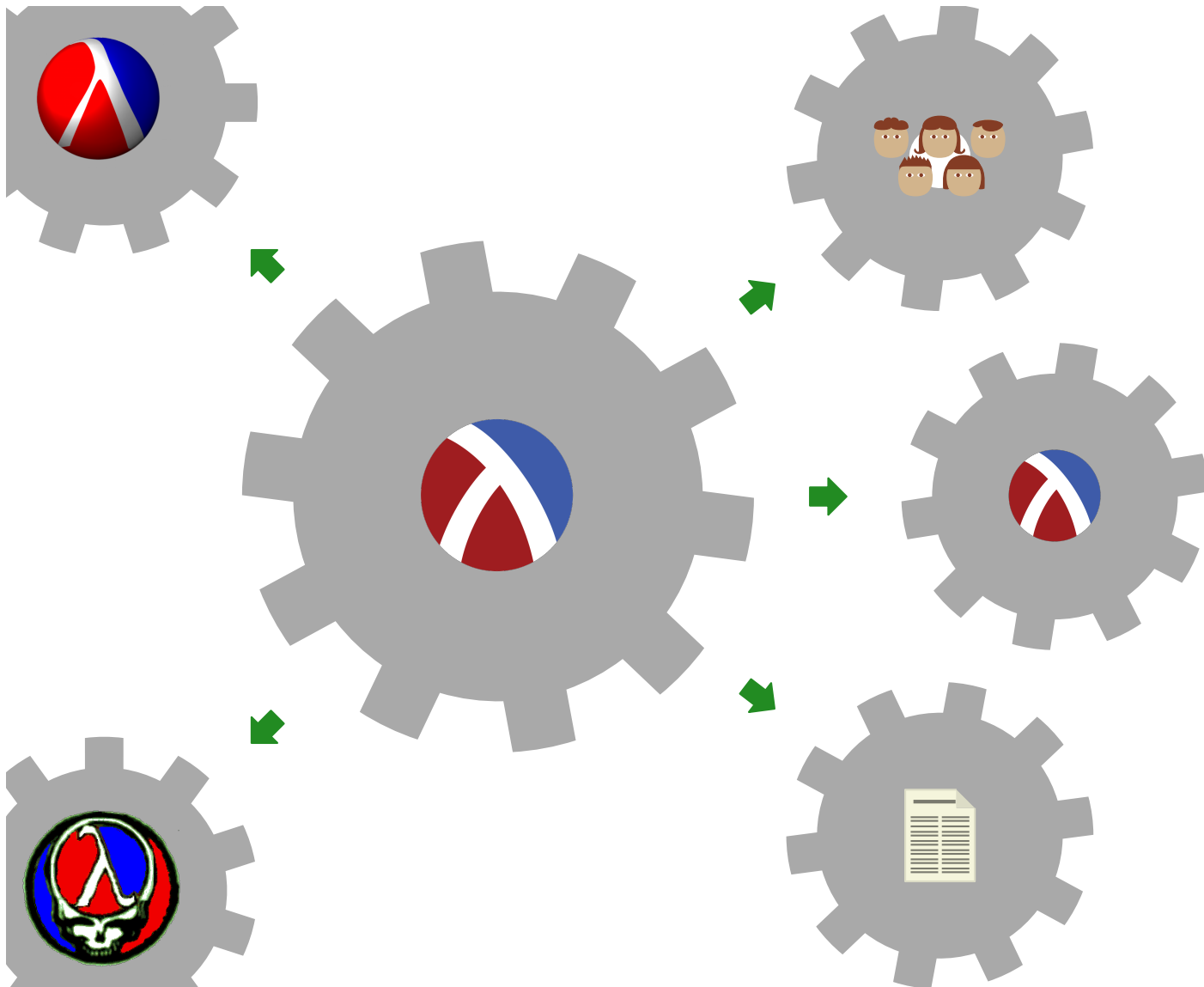
Language-Oriented Programming

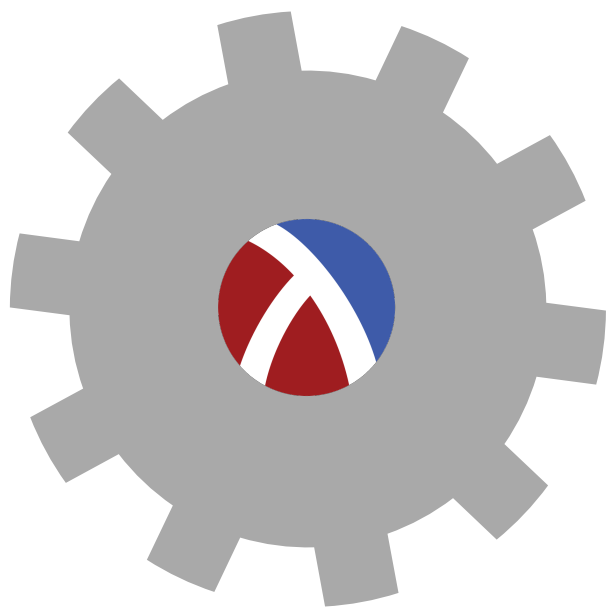














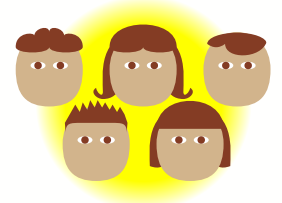
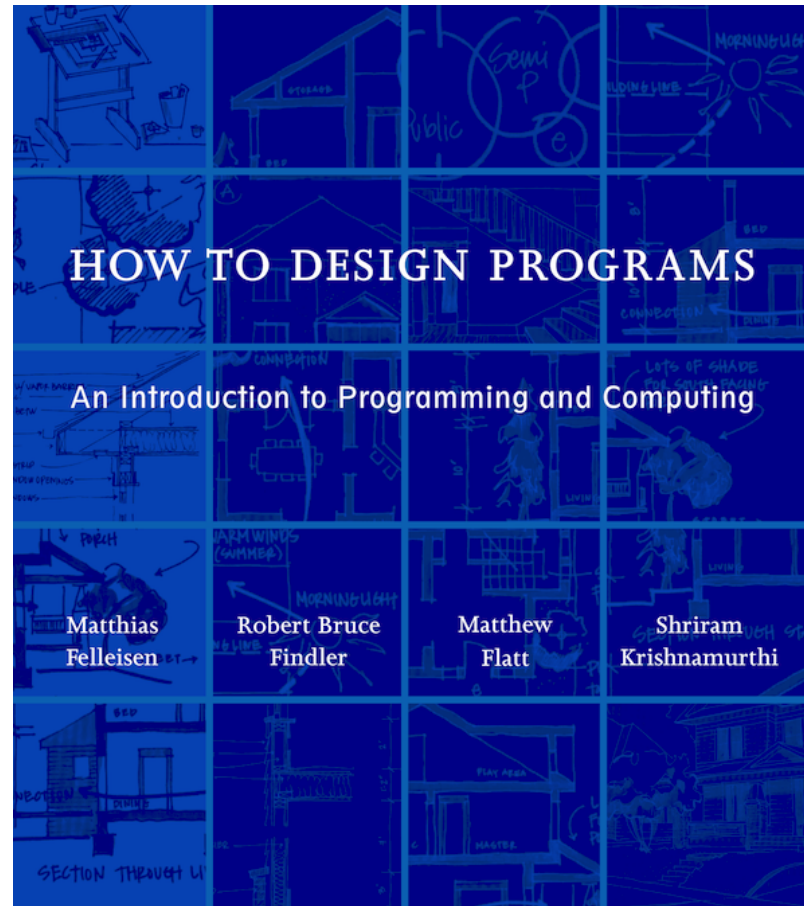


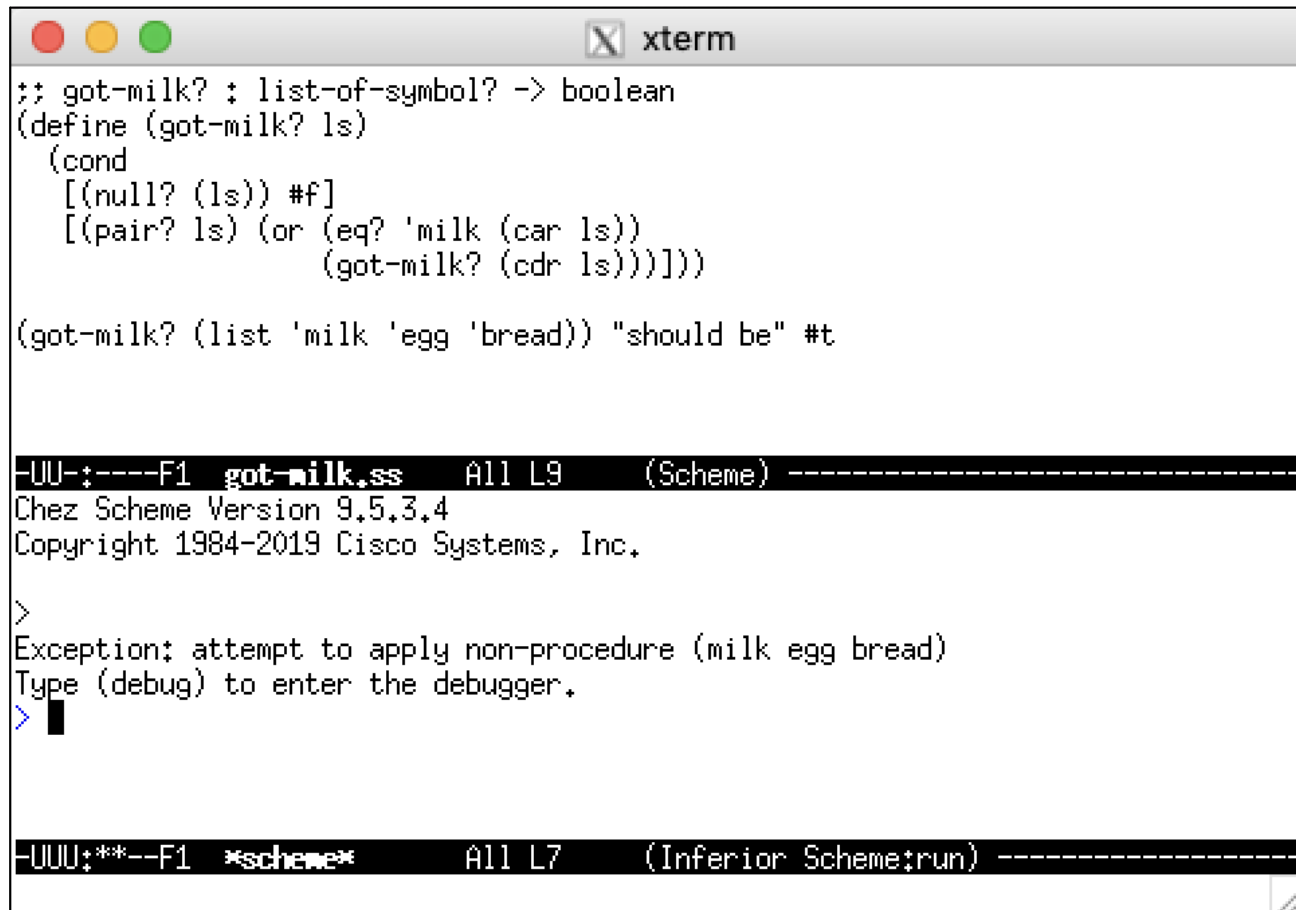
Back in 1995...

ca. 1995

HtDP and DrScheme

Felleisen et al.





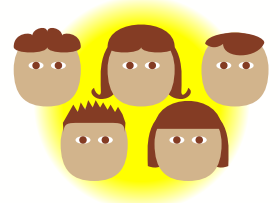
```
;; got-milk? : list-of-symbol? -> boolean
(define (got-milk? ls)
  (cond
    [(null? (ls)) #f]
    [(pair? ls) (or (eq? 'milk (car ls))
                    (got-milk? (cdr ls)))]))

(got-milk? (list 'milk 'egg 'bread)) "should be" #t

-UU-:----F1 got-milk.ss All L9 (Scheme) -----
Chez Scheme Version 9.5.3.4
Copyright 1984-2019 Cisco Systems, Inc.

>
Exception: attempt to apply non-procedure (milk egg bread)
Type (debug) to enter the debugger.
> █

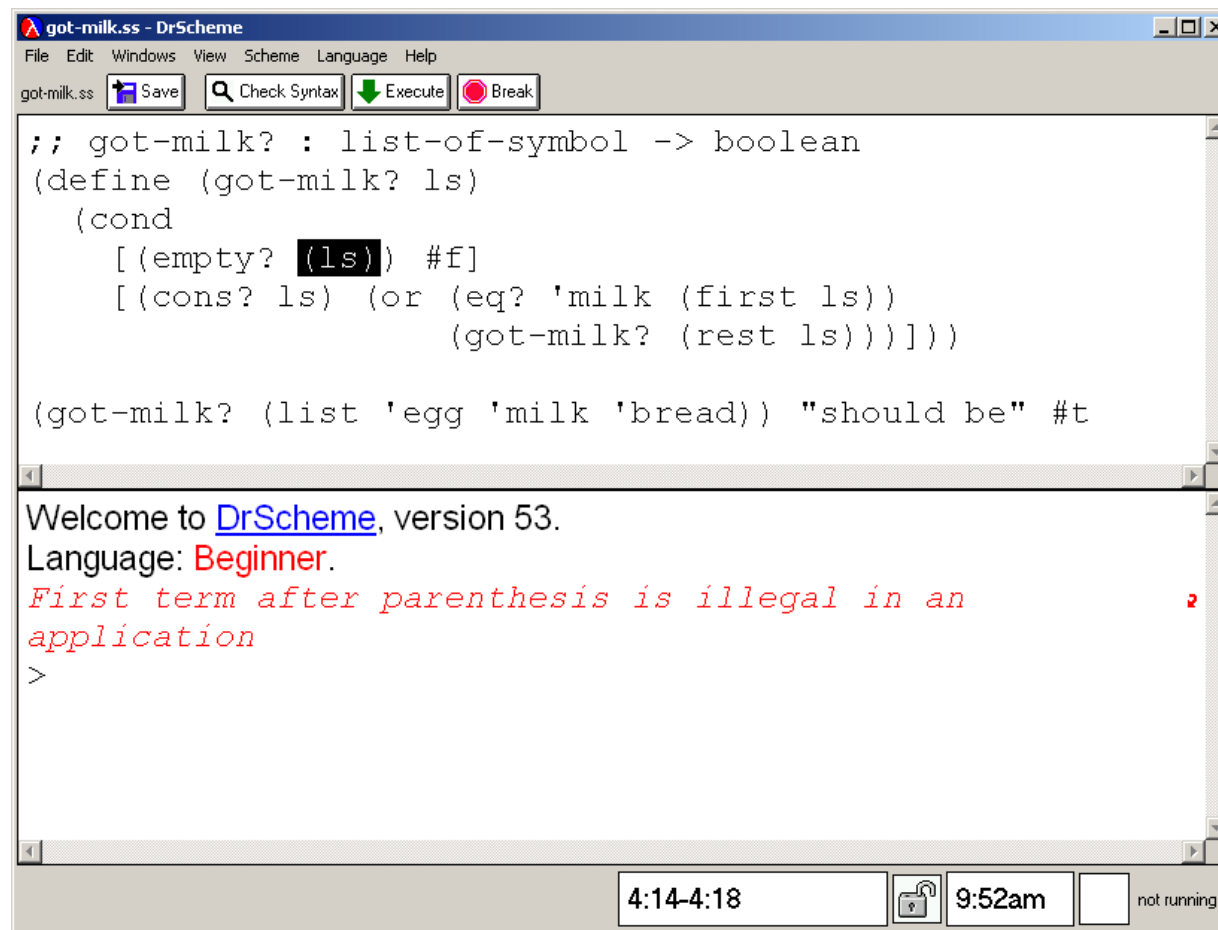
-UUU:***--F1 *scheme* All L7 (Inferior Scheme:run) -----
```



ca. 1995

HtDP and DrScheme

Felleisen et al.

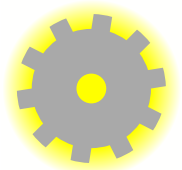
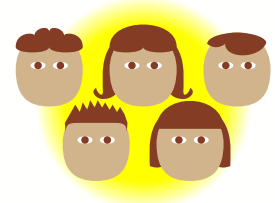


```
;; got-milk? : list-of-symbol -> boolean
(define (got-milk? ls)
  (cond
    [(empty? (ls)) #f]
    [(cons? ls) (or (eq? 'milk (first ls))
                    (got-milk? (rest ls)))]))

(got-milk? (list 'egg 'milk 'bread)) "should be" #t
```

Welcome to [DrScheme](#), version 53.
Language: **Beginner**.
First term after parenthesis is illegal in an application
>

4:14-4:18 9:52am not running



ca. 1995

HtDP and DrScheme

Felleisen et al.

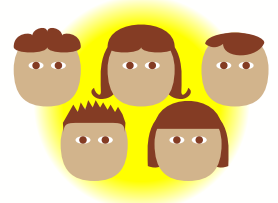


```
got-milk.ss - DrScheme
File Edit Windows View Scheme Language Help
got-milk.ss Save Check Syntax Execute Break

;; got-milk? : list-of-symbol -> boolean
(define (got-milk? ls)
  (cond
    [(empty? (ls)) #f]
    [(cons? ls) (or (eq? 'milk (first ls))
                    (got-milk? (rest ls)))]))

(got-milk? (list 'egg 'milk 'bread)) "should be" #t

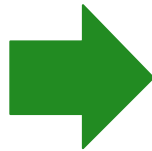
Welcome to DrScheme, version 53.
Language: Beginner.
First term after parenthesis is illegal in an
application
>
```



ca. 1995

MzScheme

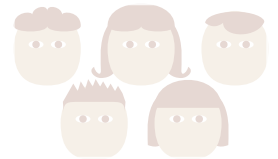
()



```
;; got-milk? : list-of-symbol -> boolean
(define (got-milk? ls)
  (cond
    [(empty? (ls)) #f]
    [(cons? ls) (or (eq? 'milk (first ls))
                    (got-milk? (rest ls)))]))

(got-milk? (list 'egg 'milk 'bread)) "should be" #t
```

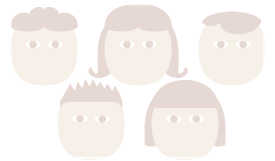
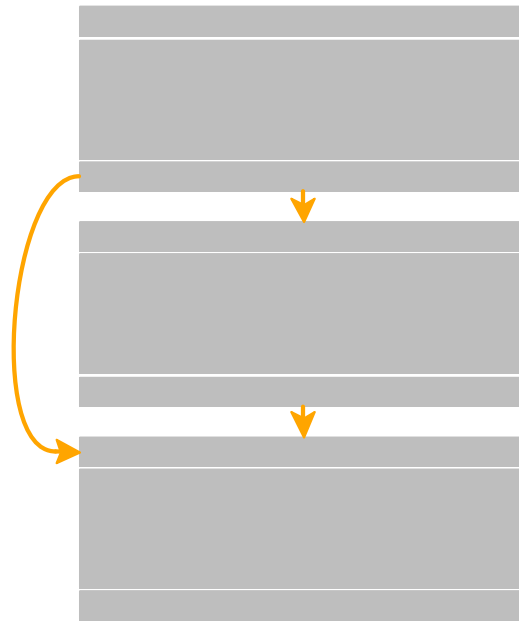
Welcome to [DrScheme](#), version 53.
Language: **Beginner**.
First term after parenthesis is illegal in an application
>



ca. 1998

Units and Mixins

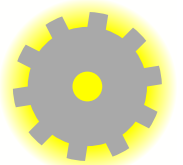
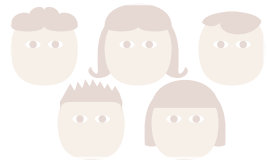
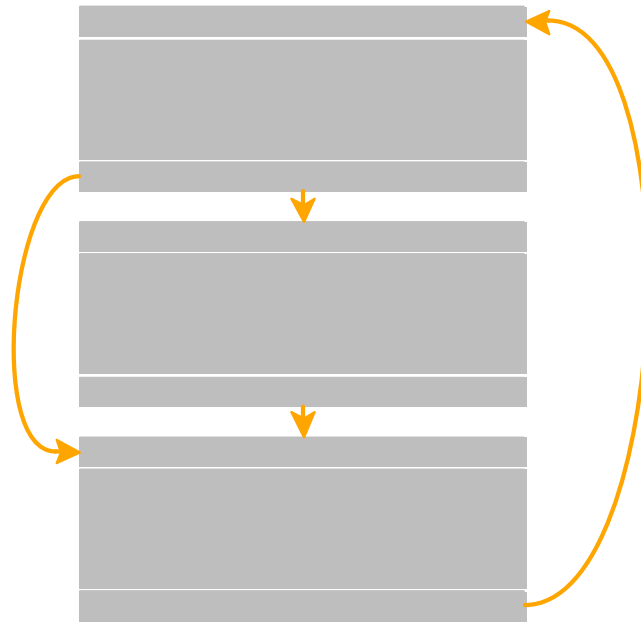
Flatt et al.



ca. 1998

Units and Mixins

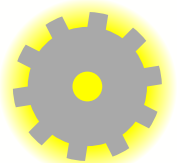
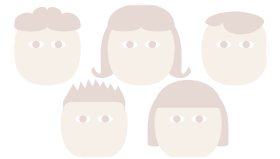
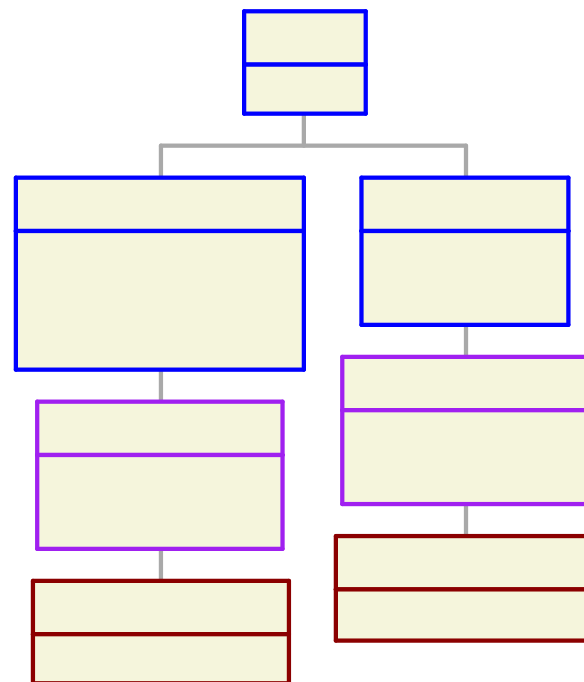
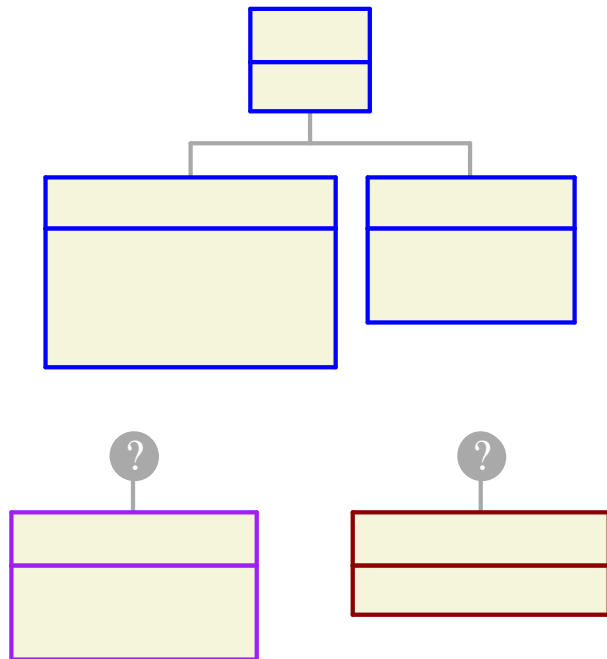
Flatt et al.



ca. 1998

Units and Mixins

Flatt et al.

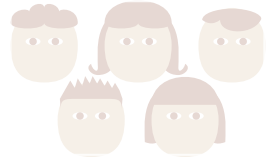


$$\frac{P, \Gamma \vdash_e e \Rightarrow e' : t' \quad \langle md, (t_1 \dots t_n \longrightarrow t), (var_1 \dots var_n), e_b \rangle \in_P t' \quad P, \Gamma \vdash_s e_j \Rightarrow e'_j : t_j \text{ for } j \in [1, n]}{P, \Gamma \vdash_e e.md(e_1 \dots e_n) \Rightarrow e'.md(e'_1 \dots e'_n) : t} \text{[call}^c\text{]}$$

$$\frac{\begin{array}{l} \overline{t_i} \cup \overline{t_{p1}} \cup \overline{t_{p2}} \cup \overline{x_i} \cup \overline{x_{p1}} \cup \overline{x_{p2}} \text{ distinct} \quad \overline{t_e} \cup \overline{x_e} \text{ distinct} \\ \overline{t_{w1} :: \kappa_{w1}} \cup \overline{x_{w1} :: \tau_{w1}} \subseteq \overline{t_i :: \kappa_i} \cup \overline{t_{p2} :: \kappa_{p2}} \cup \overline{x_i :: \tau_i} \cup \overline{x_{p2} :: \tau_{p2}} \\ \overline{t_{w2} :: \kappa_{w2}} \cup \overline{x_{w2} :: \tau_{w2}} \subseteq \overline{t_i :: \kappa_i} \cup \overline{t_{p1} :: \kappa_{p1}} \cup \overline{x_i :: \tau_i} \cup \overline{x_{p1} :: \tau_{p1}} \\ \overline{t_e :: \kappa_e} \cup \overline{x_e :: \tau_e} \subseteq \overline{t_{p1} :: \kappa_{p1}} \cup \overline{t_{p2} :: \kappa_{p2}} \cup \overline{x_{p1} :: \tau_{p1}} \cup \overline{x_{p2} :: \tau_{p2}} \\ \Gamma \vdash_{e1} : \text{sig}[i1, e1, b1] \quad \Gamma \vdash_{e2} : \text{sig}[i2, e2, b2] \\ \Gamma \vdash \text{sig}[w1, p1, b1] :: \Omega \quad \Gamma \vdash \text{sig}[w2, p2, b2] :: \Omega \\ \text{sig}[i1, e1, b1] \leq \text{sig}[w1, p1, b1] \quad \text{sig}[i2, e2, b2] \leq \text{sig}[w2, p2, b2] \\ \Gamma \vdash \text{sig}[i, e, b2] :: \Omega \end{array}}{\Gamma \vdash \text{compound import } \overline{t_i :: \kappa_i} \overline{x_i :: \tau_i} \text{ export } \overline{t_e :: \kappa_e} \overline{x_e :: \tau_e} \text{ link } e_1 \text{ with } \overline{t_{w1} :: \kappa_{w1}} \overline{x_{w1} :: \tau_{w1}} \text{ provides } \overline{t_{p1} :: \kappa_{p1}} \overline{x_{p1} :: \tau_{p1}} \text{ and } e_2 \text{ with } \overline{t_{w2} :: \kappa_{w2}} \overline{x_{w2} :: \tau_{w2}} \text{ provides } \overline{t_{p2} :: \kappa_{p2}} \overline{x_{p2} :: \tau_{p2}} : \text{sig}[i, e, b2]}$$

$$\frac{P, \Gamma \vdash_e e \Rightarrow e' : t' \quad \langle c.f.d, t \rangle \in_P t'}{P, \Gamma \vdash_e e.f.d \Rightarrow e' : \underline{c.f.d} : t} \text{[get}^c\text{]}$$

$$\frac{\begin{array}{l} \overline{t} \cup \overline{x} \text{ distinct} \quad \Gamma \vdash \tau_b :: \Omega \quad \Gamma \vdash \overline{\sigma} :: \overline{\kappa} \\ \Gamma \vdash \overline{\sigma} \overline{e} : \overline{\tau} \quad \Gamma \vdash e_u : \text{sig}[i, e, b] \\ \text{sig}[i, e, b] \leq \text{sig import } \overline{t :: \kappa} \overline{x :: \tau} \text{ export } \emptyset \tau_b \end{array}}{\Gamma \vdash \text{invoke } e_u \text{ with } \overline{t :: \kappa} = \overline{\sigma} \overline{x :: \tau} \equiv \overline{e} : \tau_b}$$



ca. 1998

Units and Mixins

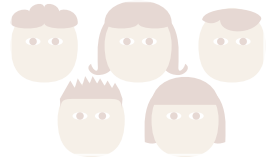
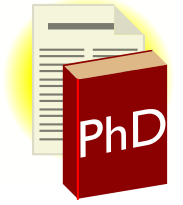
Flatt et al.

$$\frac{P, \Gamma \vdash_e e \Rightarrow e' : t' \quad \langle md, (t_1 \dots t_n \longrightarrow t), (var_1 \dots var_n), e_b \rangle \in_P t' \quad P, \Gamma \vdash_s e_j \Rightarrow e'_j : t_j \text{ for } j \in [1, n]}{P, \Gamma \vdash_e e.md(e_1 \dots e_n) \Rightarrow e'.md(e'_1 \dots e'_n) : t} \text{[call}^c\text{]}$$

$$\frac{\begin{array}{l} \overline{t_i} \cup \overline{t_{p1}} \cup \overline{t_{p2}} \cup \overline{x_i} \cup \overline{x_{p1}} \cup \overline{x_{p2}} \text{ distinct} \quad \overline{t_e} \cup \overline{x_e} \text{ distinct} \\ \overline{t_{w1} :: \kappa_{w1}} \cup \overline{x_{w1} :: \tau_{w1}} \subseteq \overline{t_i :: \kappa_i} \cup \overline{t_{p2} :: \kappa_{p2}} \cup \overline{x_i :: \tau_i} \cup \overline{x_{p2} :: \tau_{p2}} \\ \overline{t_{w2} :: \kappa_{w2}} \cup \overline{x_{w2} :: \tau_{w2}} \subseteq \overline{t_i :: \kappa_i} \cup \overline{t_{p1} :: \kappa_{p1}} \cup \overline{x_i :: \tau_i} \cup \overline{x_{p1} :: \tau_{p1}} \\ \overline{t_e :: \kappa_e} \cup \overline{x_e :: \tau_e} \subseteq \overline{t_{p1} :: \kappa_{p1}} \cup \overline{t_{p2} :: \kappa_{p2}} \cup \overline{x_{p1} :: \tau_{p1}} \cup \overline{x_{p2} :: \tau_{p2}} \\ \Gamma \vdash_{e1} : \text{sig}[i1, e1, b1] \quad \Gamma \vdash_{e2} : \text{sig}[i2, e2, b2] \\ \Gamma \vdash \text{sig}[w1, p1, b1] :: \Omega \quad \Gamma \vdash \text{sig}[w2, p2, b2] :: \Omega \\ \text{sig}[i1, e1, b1] \leq \text{sig}[w1, p1, b1] \quad \text{sig}[i2, e2, b2] \leq \text{sig}[w2, p2, b2] \\ \Gamma \vdash \text{sig}[i, e, b2] :: \Omega \end{array}}{\Gamma \vdash \text{compound import } \overline{t_i :: \kappa_i} \overline{x_i :: \tau_i} \text{ export } \overline{t_e :: \kappa_e} \overline{x_e :: \tau_e} \text{ link } e_1 \text{ with } \overline{t_{w1} :: \kappa_{w1}} \overline{x_{w1} :: \tau_{w1}} \text{ provides } \overline{t_{p1} :: \kappa_{p1}} \overline{x_{p1} :: \tau_{p1}} \text{ and } e_2 \text{ with } \overline{t_{w2} :: \kappa_{w2}} \overline{x_{w2} :: \tau_{w2}} \text{ provides } \overline{t_{p2} :: \kappa_{p2}} \overline{x_{p2} :: \tau_{p2}} : \text{sig}[i, e, b2]}$$

$$\frac{P, \Gamma \vdash_e e \Rightarrow e' : t' \quad \langle c.f.d, t \rangle \in_P t'}{P, \Gamma \vdash_e e.f.d \Rightarrow e' : \underline{c}.fd : t} \text{[get}^c\text{]}$$

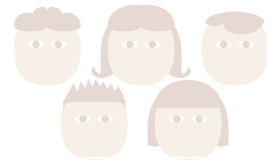
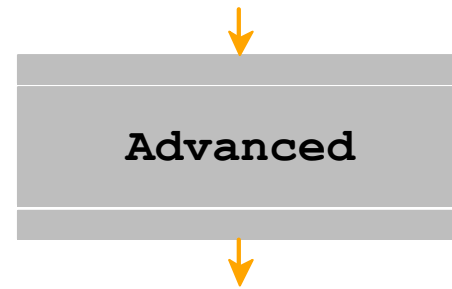
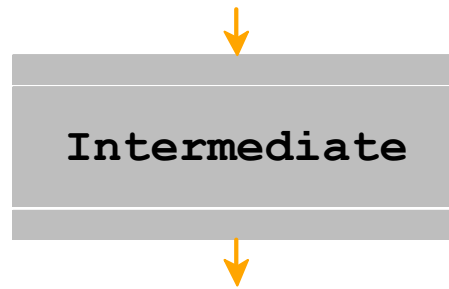
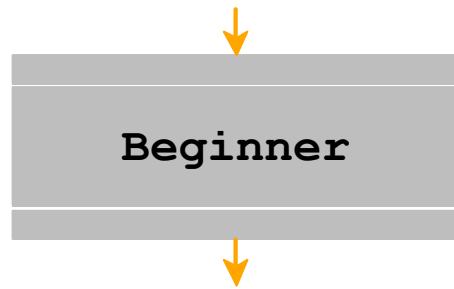
$$\frac{\begin{array}{l} \overline{t} \cup \overline{x} \text{ distinct} \quad \Gamma \vdash \tau_b :: \Omega \quad \Gamma \vdash \overline{\sigma} :: \overline{\kappa} \\ \Gamma \vdash \overline{s} \overline{e} : \overline{\tau} \quad \Gamma \vdash e_u : \text{sig}[i, e, b] \\ \text{sig}[i, e, b] \leq \text{sig import } \overline{t :: \kappa} \overline{x :: \tau} \text{ export } \emptyset \tau_b \end{array}}{\Gamma \vdash \text{invoke } e_u \text{ with } \overline{t :: \kappa} = \overline{\sigma} \overline{x :: \tau} \equiv \overline{e} : \tau_b}$$



ca. 2000

Unit/lang

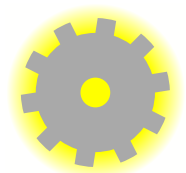
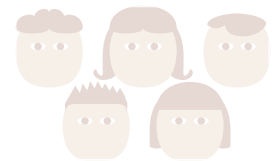
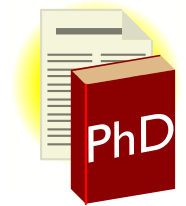
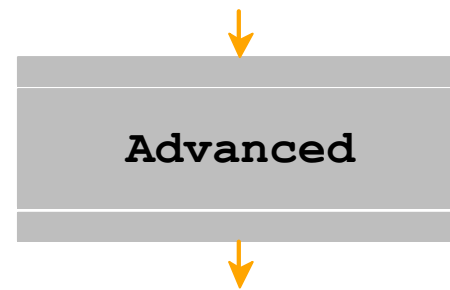
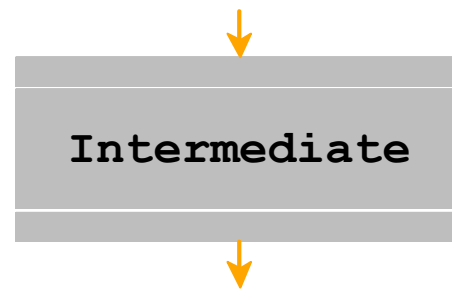
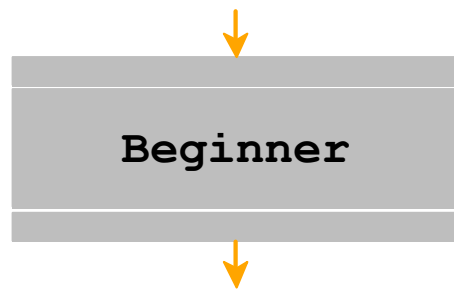
Krishnamurthi



ca. 2000

Unit/lang

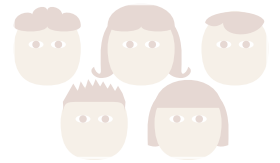
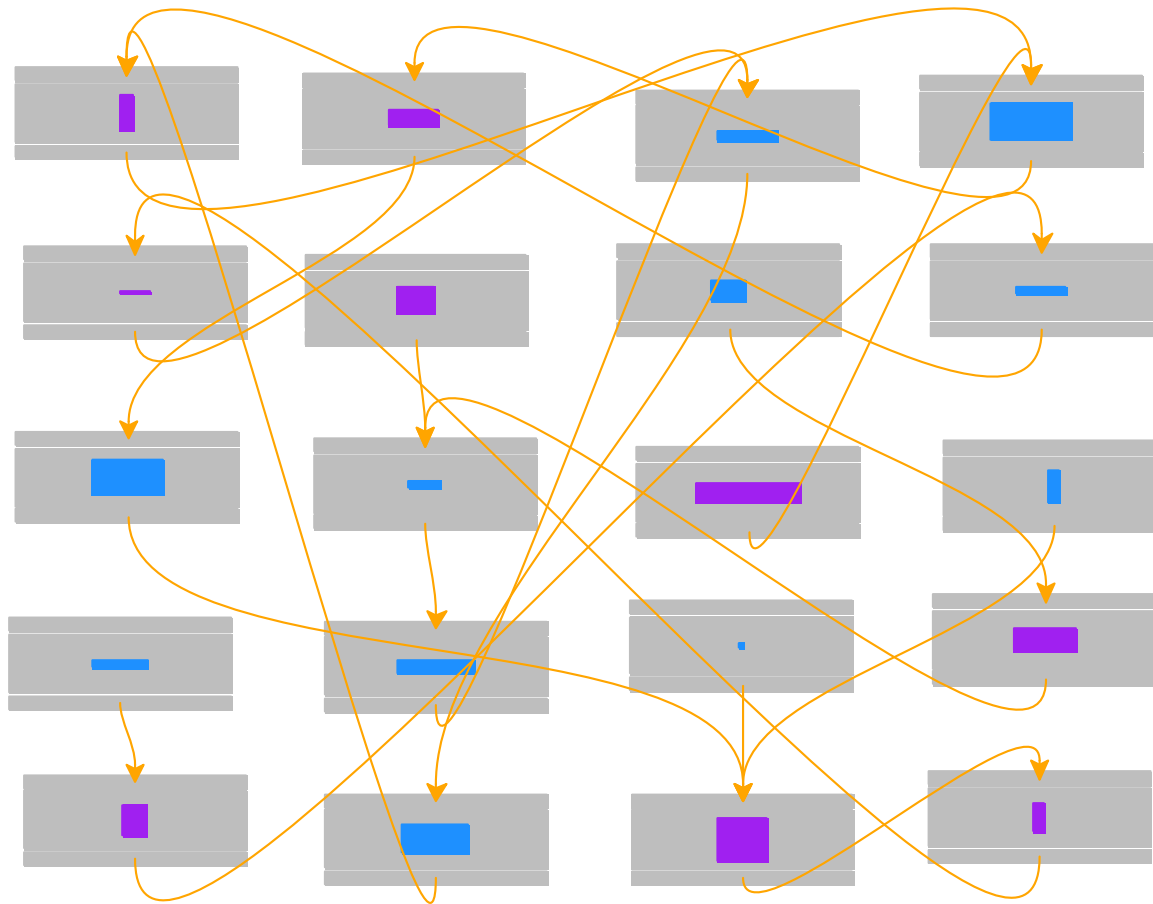
Krishnamurthi



ca. 2000

Units Experience

■ run time
■ compile time

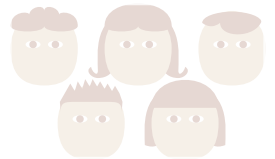


ca. 2002

Modules and Macros

```
#lang racket
(require slideshow/base
         (for-syntax syntax/parse))

(define-syntax (slides stx)
  (syntax-case stx ()
    [(_ id) #'(void)]
    [(_ id e es ...)
     #'(let ([id e])
         (slide id)
         (slides id es ...)))]))
```

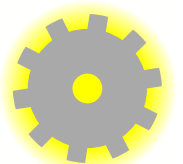
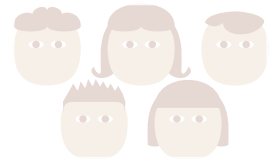


ca. 2002

Modules and Macros

```
#lang racket
(require slideshow/base
         (for-syntax syntax/parse))

(define-syntax (slides stx)
  (syntax-case stx ()
    [(_ id) #'(void)]
    [(_ id e es ...)
     #'(let ([id e])
         (slide id)
         (slides id es ...)))]))
```



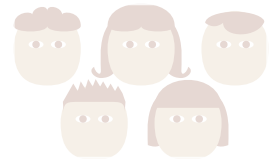
ca. 2002

Experience with Modules

```
(help ☠)
```

```
(define (help x)
```

```
.... (+ x 1) ....)
```



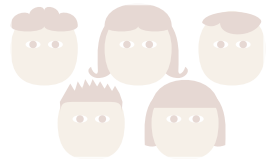
ca. 2002

Experience with Modules

```
(help ☠)
```

```
(define (help x)
```

```
..... (+ x 1) .....
```

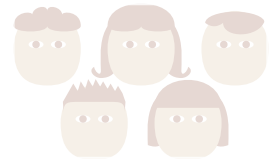


ca. 2002

Experience with Modules

```
(help ☠)
```

```
(define (help x)
  (unless (number? x)
    (error ....))
  .... (+ x 1) ....)
```



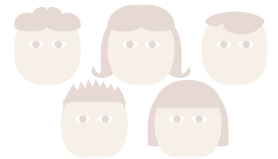
ca. 2002

Experience with Modules

```
(help (λ (x)  
        (+ x 1)))
```

```
(define (help proc)
```

```
.... (proc ☠) ....)
```



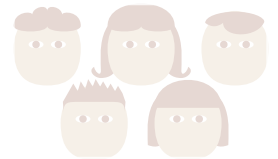
ca. 2002

Experience with Modules

```
(help (λ (x)  
      (+ x 1)))
```

```
(define (help proc)
```

```
.... (proc ☠) ....)
```



ca. 2002

Contracts

Findler

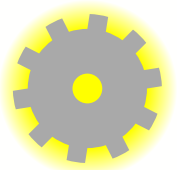
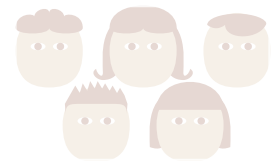
```
(help (λ (x)
      (+ x 1)))
```



```
(number? . -> . number?)
```

```
(define (help proc)
```

```
.... (proc ) ....)
```



ca. 2008

Typed Racket

Tobin-Hochstadt

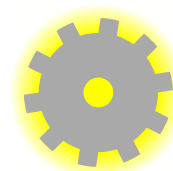
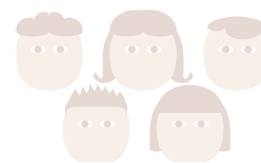
```
(help (λ (x)
        (+ x 1)))
```

$\Gamma \vdash \tau$

(Number -> Number)

```
(define (help proc)
```

```
.... (proc ☠) ....)
```



ca. 1992

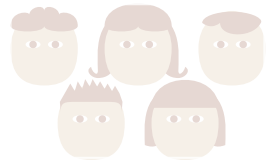
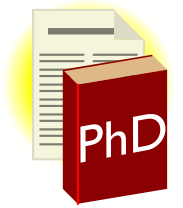
Soft Typing

Fagan

`(help (λ (x)
 (+ x 1)))`



$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}$$



ca. 1994

Soft Typing

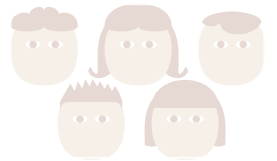
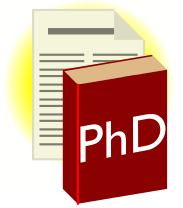
Wright

`(help (λ (x)
 (+ x 1)))`



$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}$$

$$\frac{\vdash T_1 \leftrightarrow T_2[A \leftarrow \mu A.T_2]}{\vdash T_1 \leftrightarrow \mu A.T_2}$$



ca. 1997

Set-Based Analysis

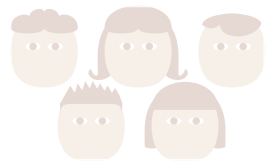
Flanagan

```
(help (λ (x)  
  (+ x 1)))
```



```
(define (help proc)
```

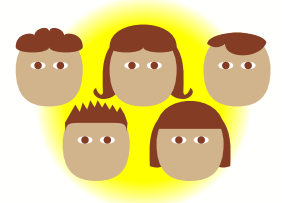
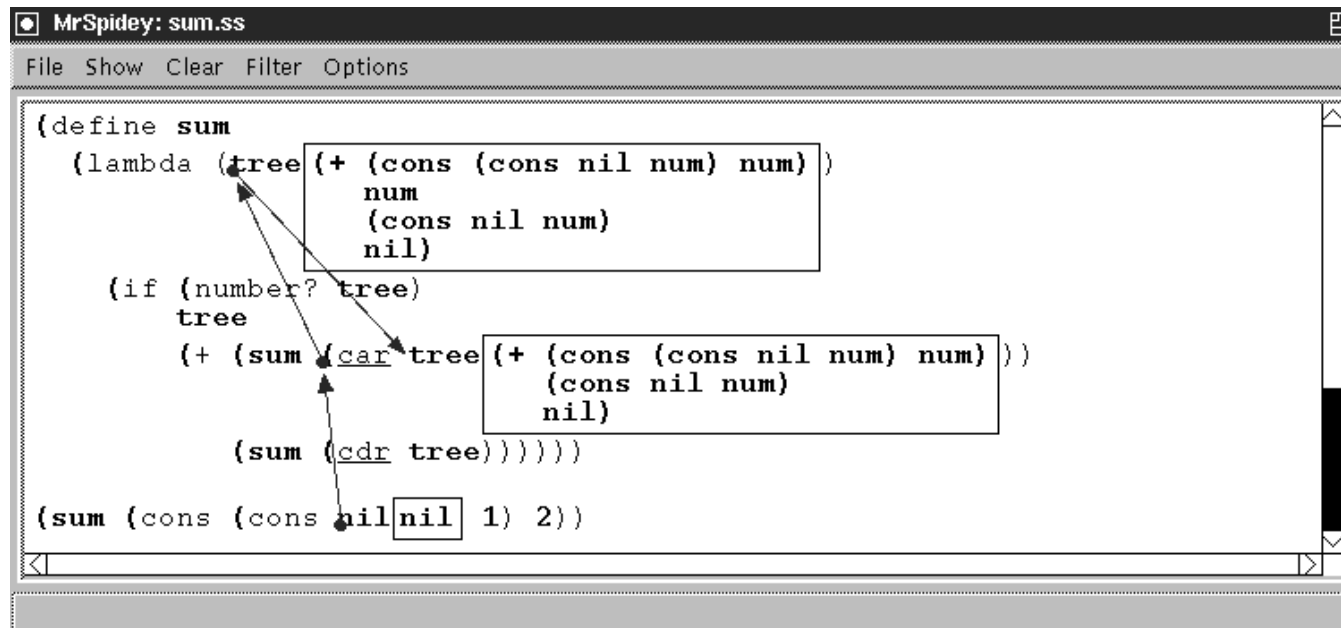
```
.... (proc  ) ....)
```



ca. 1997

Set-Based Analysis

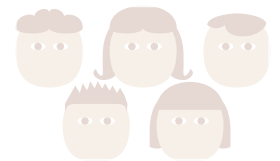
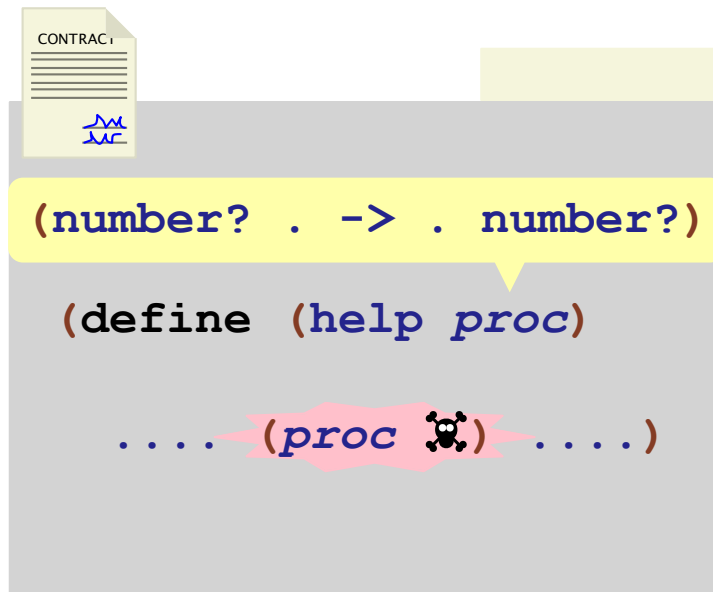
Flanagan



ca. 2006

Modular Set-Based Analysis

Meunier



ca. 2008

Typed Racket

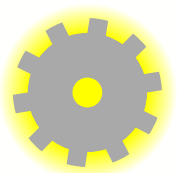
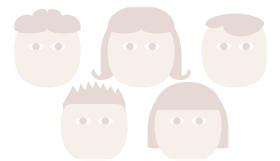
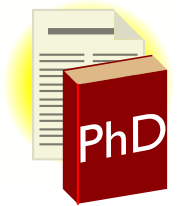
Tobin-Hochstadt

$\Gamma \vdash \tau$

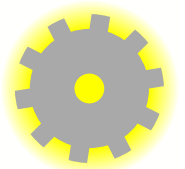
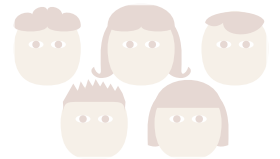
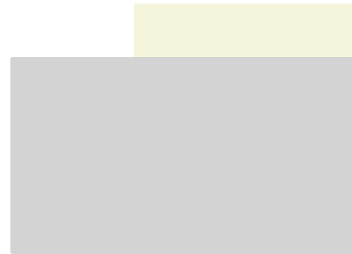
`(Number -> Number)`

`(define (help proc)`

`.... (proc ☠))`

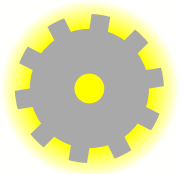
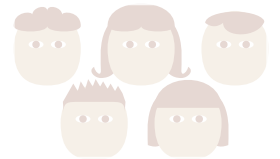


Modules and Macros and Types



Modules and Macros and Types

$$\Gamma \vdash \tau$$

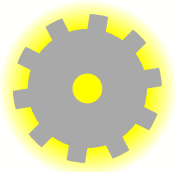
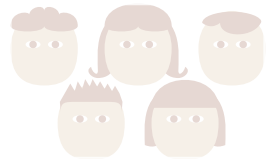
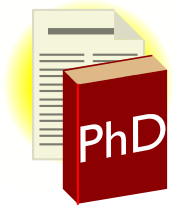
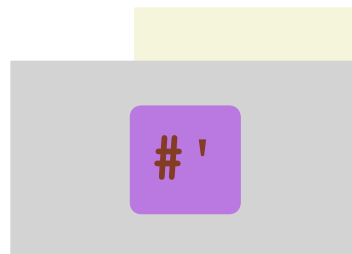


ca. 2008

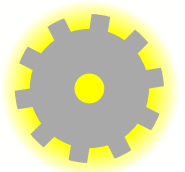
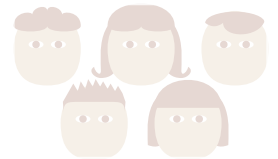
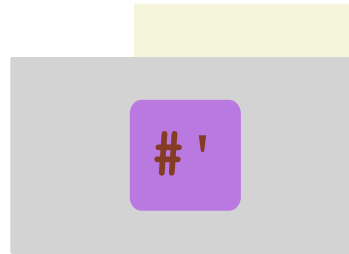
syntax-parse

Culpepper

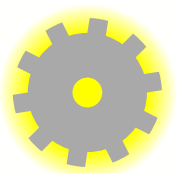
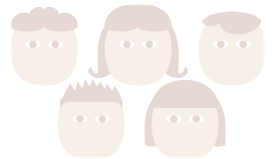
$\Gamma \vdash \tau$



Modules and Macros and Contracts and Types

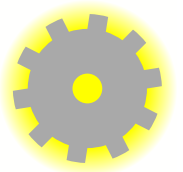
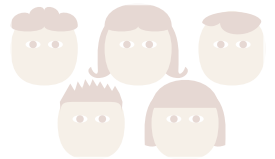
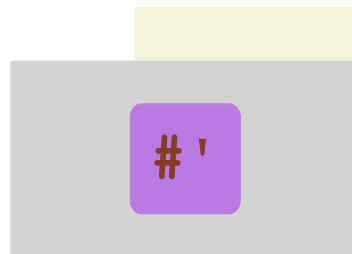


Modules and Macros and Contracts and Types

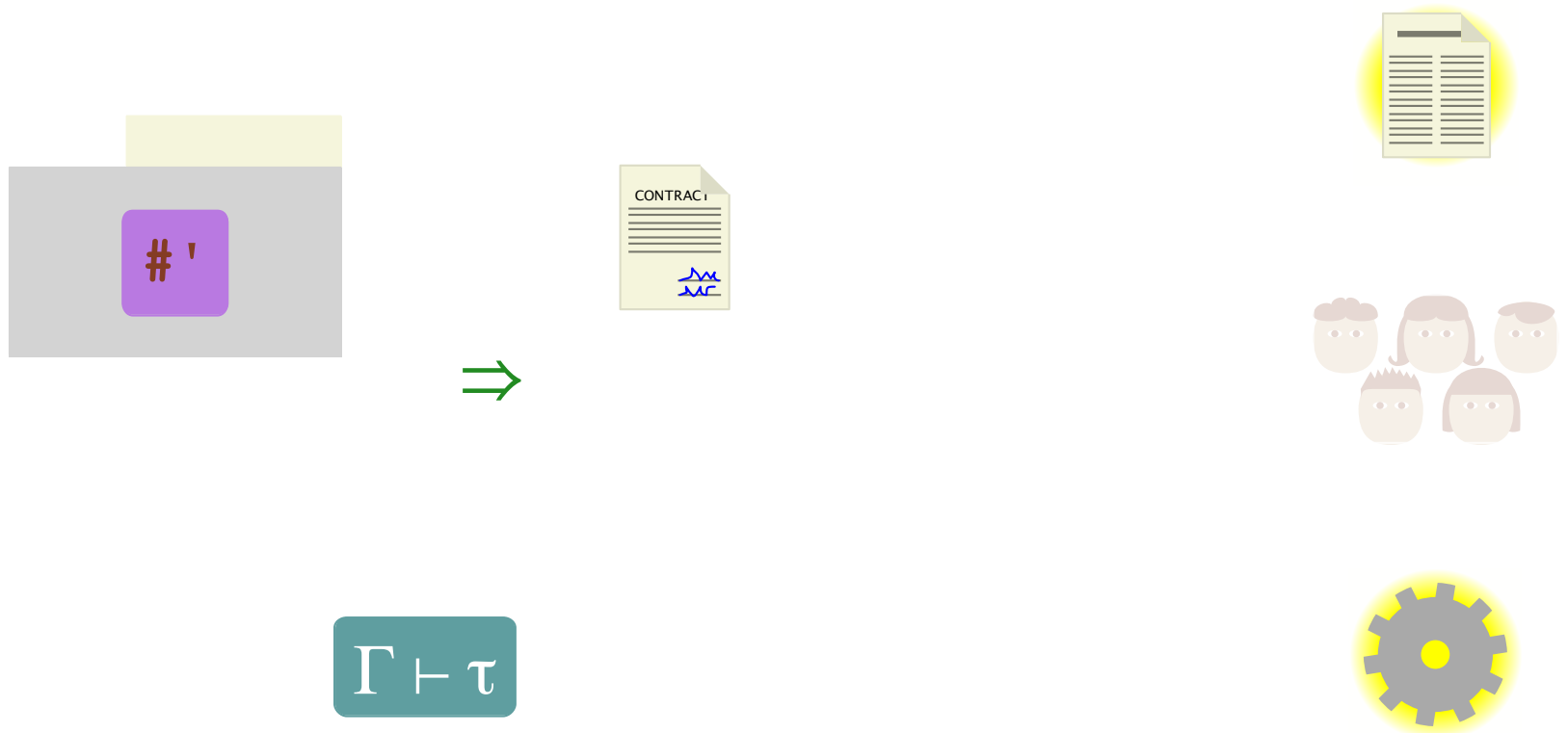


Modules and Macros and Contracts and Types

$\Gamma \vdash \tau$

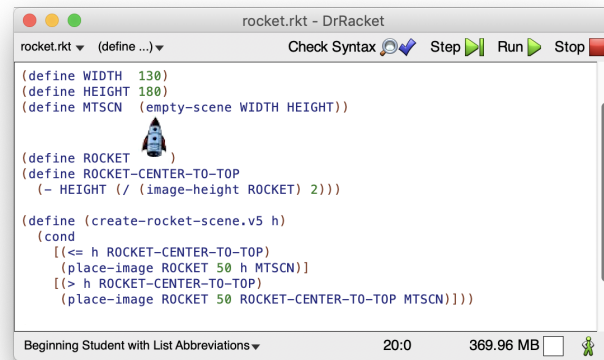


Modules and Macros and Contracts and Types




ca. 2004

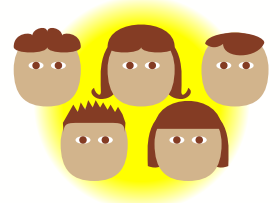
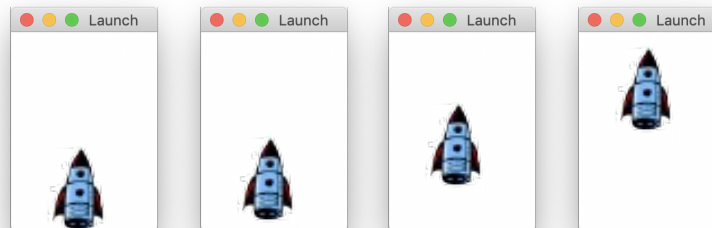
Big Bang



```
(define WIDTH 130)
(define HEIGHT 180)
(define MTSCN (empty-scene WIDTH HEIGHT))

(define ROCKET )
(define ROCKET-CENTER-TO-TOP
  (- HEIGHT (/ (image-height ROCKET) 2)))

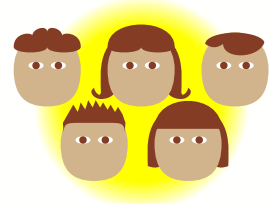
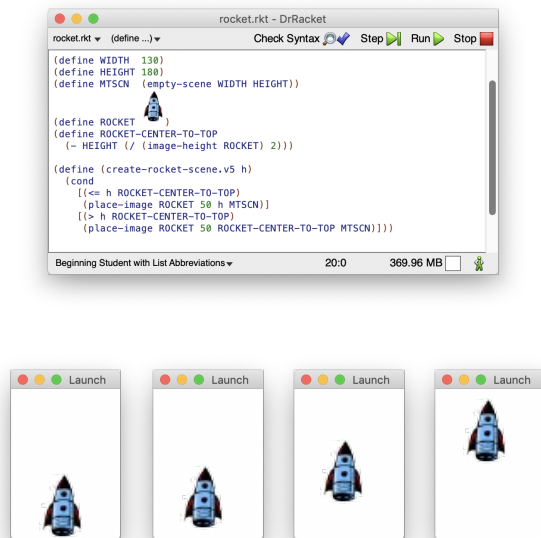
(define (create-rocket-scene.v5 h)
  (cond
    [(<= h ROCKET-CENTER-TO-TOP)
     (place-image ROCKET 50 h MTSCN)]
    [(> h ROCKET-CENTER-TO-TOP)
     (place-image ROCKET 50 ROCKET-CENTER-TO-TOP MTSCN)]))
```



ca. 2006

Bootstrap

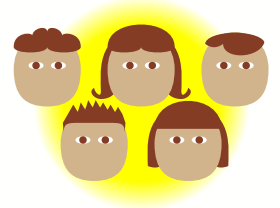
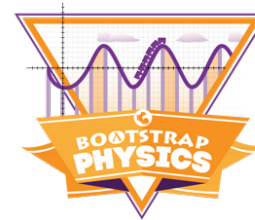
Schanzer et al.



ca. 2016

Bootstrap

Schanzer et al.



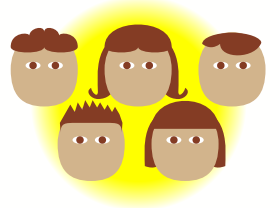
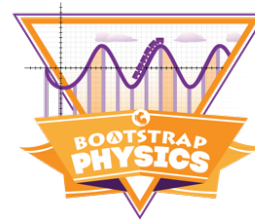
ca. 2016

Bootstrap

Schanzer et al.



bootstrapworld.org



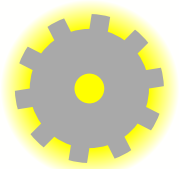
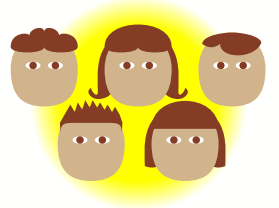
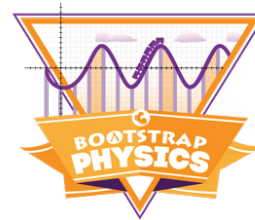
ca. 2016

Bootstrap

Schanzer et al.



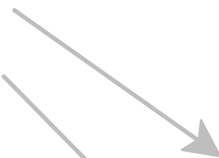
bootstrapworld.org



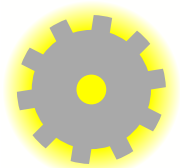
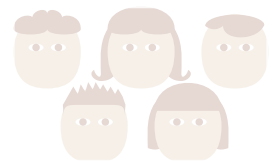
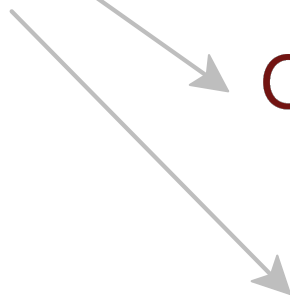
ca. 2004

Foreign Function Interface

Barzilay



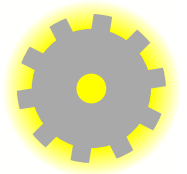
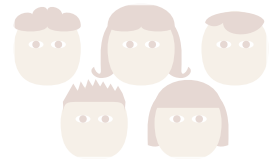
OpenSSL
Cryptography and SSL/TLS Toolkit



ca. 2006

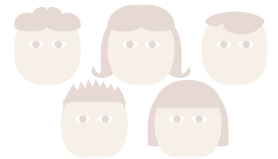
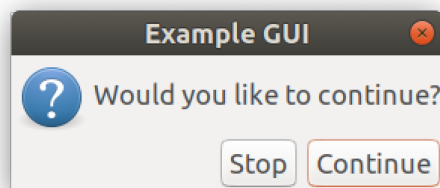
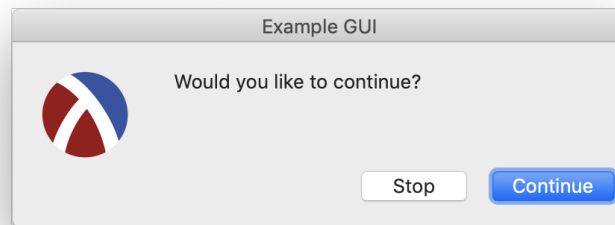
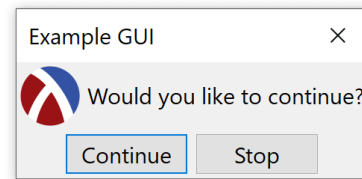
Native-Code Compiler

```
subi    (imm 16), %sfp
mov     %ac0, %rcx
mov     (disp 8 %sfp), %ac0
cmpi    (imm 6), %rcx
```



ca. 2010

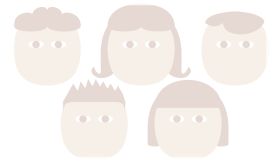
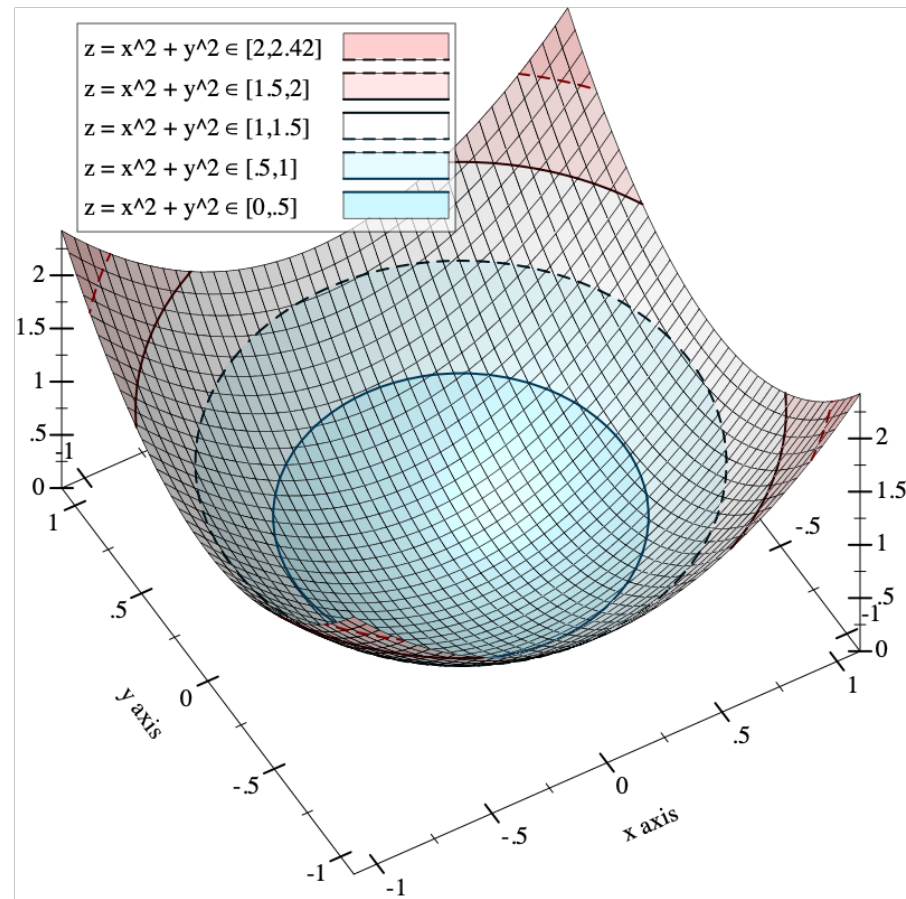
GUI Library Rewrite



ca. 2012

Math & Plot

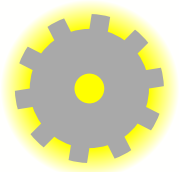
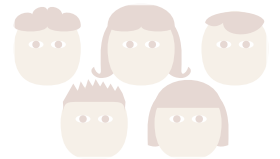
Toronto



ca. 2014

Packaging, Distribution, and Build

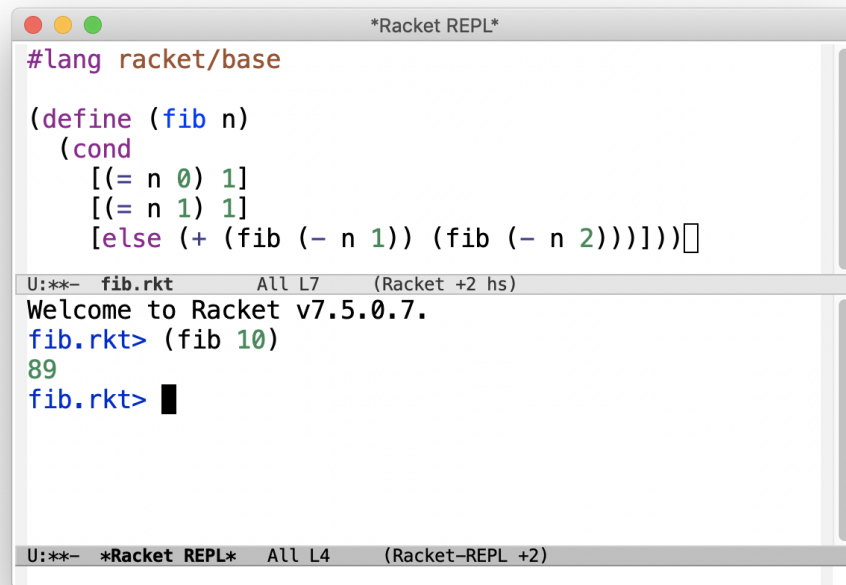
McCarthy & Flatt



ca. 2014

Racket Mode

Hendershott



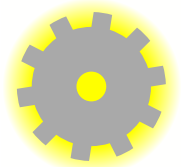
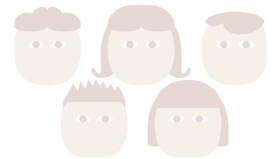
```
*Racket REPL*

#lang racket/base

(define (fib n)
  (cond
    [(= n 0) 1]
    [(= n 1) 1]
    [else (+ (fib (- n 1)) (fib (- n 2)))])])

U:**~ fib.rkt All L7 (Racket +2 hs)
Welcome to Racket v7.5.0.7.
fib.rkt> (fib 10)
89
fib.rkt> █

U:**~ *Racket REPL* All L4 (Racket-REPL +2)
```



ca. 2008

Documentation

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

1 Getting Started

2 @ Syntax

3 High-Level Scribble API

4 Scribbling Documentation

5 Literate Programming

6 Low-Level Scribble API

7 Running scribble

Index

ON THIS PAGE:

Scribble: The Racket Documentation Tool

v7.5.0.6

Scribble: The Racket Documentation Tool

by Matthew Flatt
and Eli Barzilay

Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via Latex) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scriblings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

1.1 A First Example

1.2 Multiple Sections

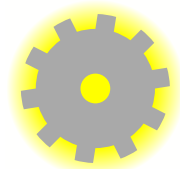
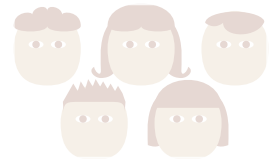
1.3 Splitting the Document Source

1.4 Document Styles

1.5 More Functions

1.5.1 Centering

1.5.2 Margin Notes



ca. 2008

Documentation

...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

1 Getting Started

2 @ Syntax

3 High-Level Scribble API

4 Scribbling Documentation

5 Literate Programming

6 Low-Level Scribble API

7 Running scribble

Index

ON THIS PAGE:

Scribble: The Racket Documentation Tool

v7.5.0.6

Scribble: The Racket Documentation Tool

by Matthew Flatt
and Eli Barzilay

Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via Latex) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scribblings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

1.1 A First Example

1.2 Multiple Sections

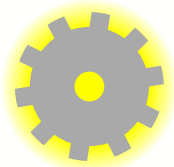
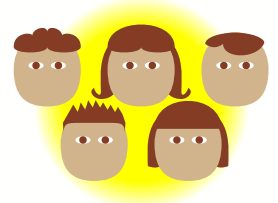
1.3 Splitting the Document Source

1.4 Document Styles

1.5 More Functions

1.5.1 Centering

1.5.2 Margin Notes



ca. 2008

Documentation

...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

1 Getting Started

2 @ Syntax

3 High-Level Scribble API

4 Scribbling Documentation

5 Literate Programming

6 Low-Level Scribble API

7 Running scribble

Index

ON THIS PAGE:

Scribble: The Racket Documentation Tool

v7.5.0.6

Scribble: The Racket Documentation Tool

by Matthew Flatt
and Eli Barzilay

Scribble is a collection of tools for creating prose documents—papers, books, library documentation, etc.—in HTML or PDF (via Latex) form. More generally, Scribble helps you write programs that are rich in textual content, whether the content is prose to be typeset or any other form of text to be generated programmatically.

This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scriblings/scribble>, starting with the "scribble.scrbl" file.

1 Getting Started

1.1 A First Example

1.2 Multiple Sections

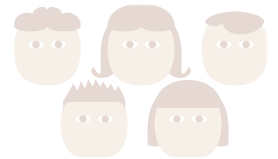
1.3 Splitting the Document Source

1.4 Document Styles

1.5 More Functions

1.5.1 Centering

1.5.2 Margin Notes



ca. 2008

Documentation

...search manuals...

top ← prev up next →

▼ Scribble: The Racket Documentation Tool

- 1 Getting Started
- 2 @ Syntax
- 3 High-Level Scribble API
- 4 Scribbling Documentation
- 5 Literate Programming
- 6 Low-Level Scribble API
- 7 Running scribble
- Index

ON THIS PAGE:

Scribble: The Racket Documentation Tool

v7.5.0.6

Scribble: The Racket Documentation Tool

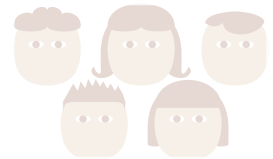
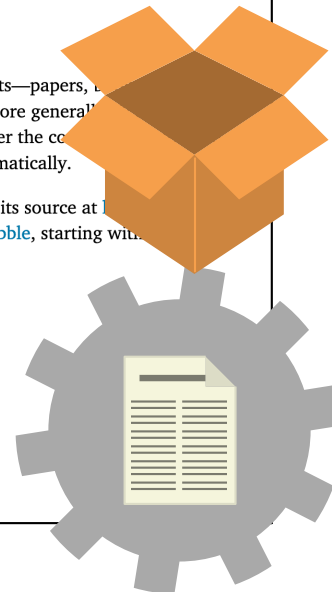
by Matthew Flatt
and Eli Barzilay

Scribble is a collection of tools for creating prose documents—papers, books, documentation, etc.—in HTML or PDF (via Latex) form. More generally, it lets you write programs that are rich in textual content, whether the content is typeset or any other form of text to be generated programmatically.

This document is itself written using Scribble. You can see its source at <https://github.com/racket/scribble/tree/master/scribble-doc/scribblings/scribble>, starting with the "scribble.scrbl" file.

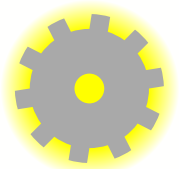
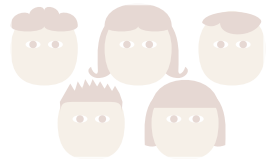
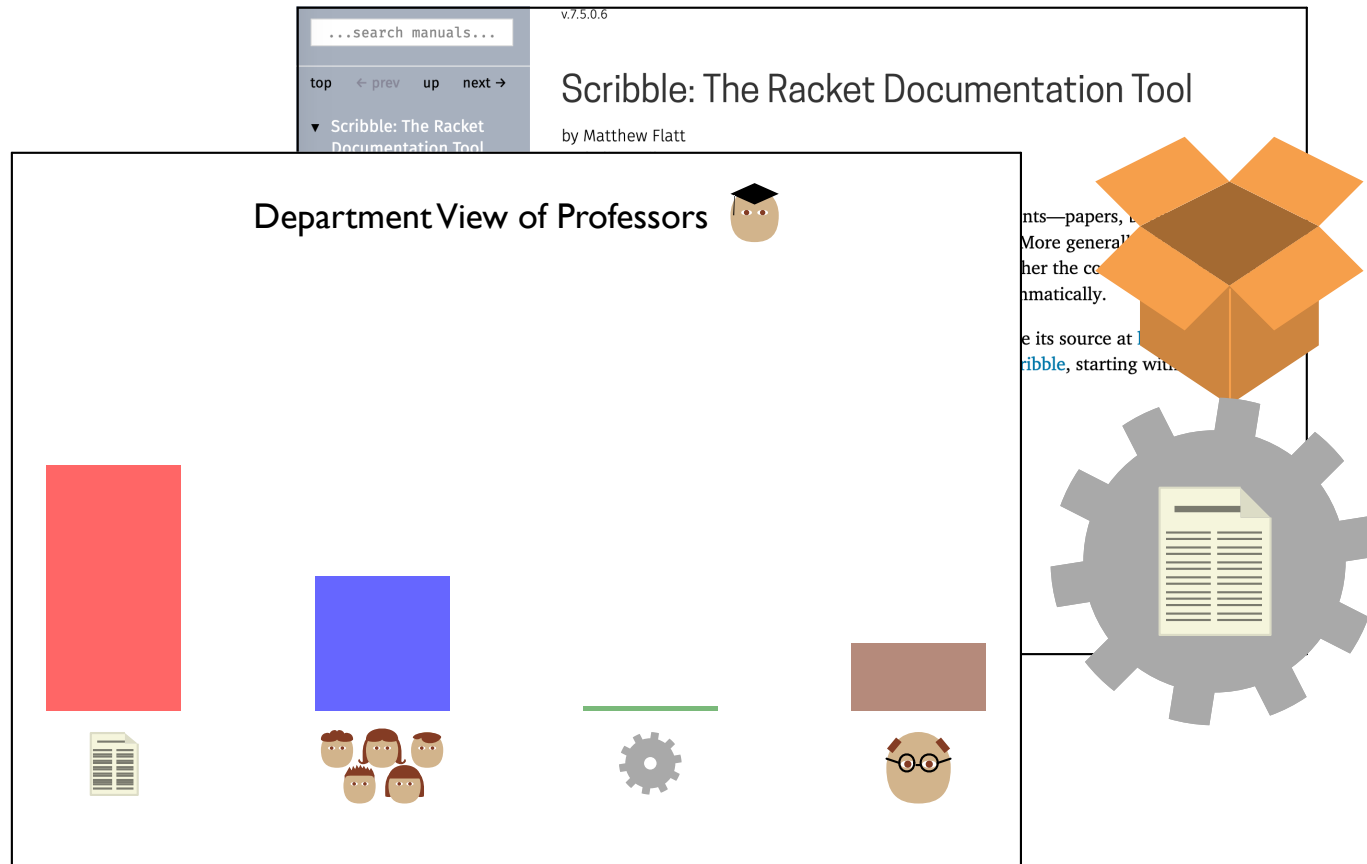
1 Getting Started

- 1.1 A First Example
- 1.2 Multiple Sections
- 1.3 Splitting the Document Source
- 1.4 Document Styles
- 1.5 More Functions
 - 1.5.1 Centering
 - 1.5.2 Margin Notes



ca. 2008

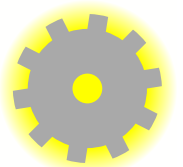
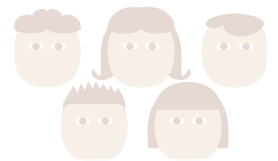
Documentation



ca. 2016

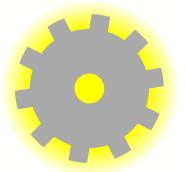
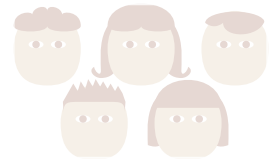
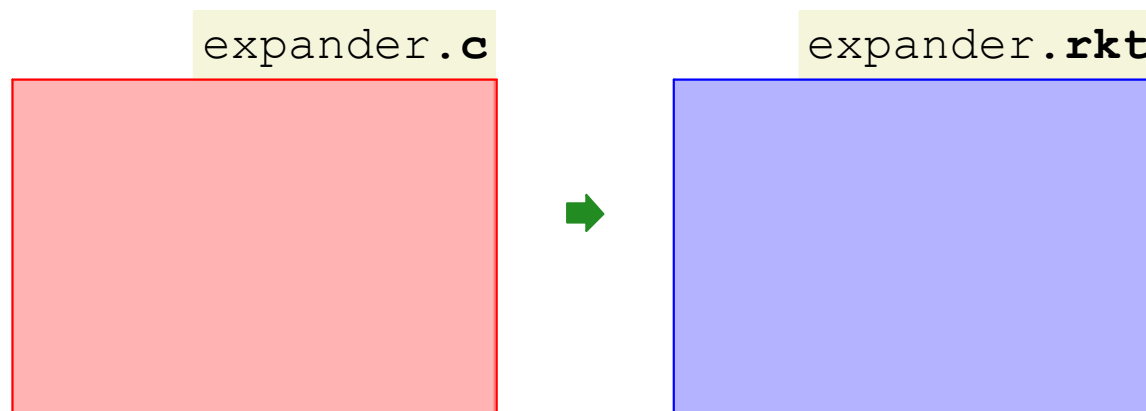
Macros and Scope

```
(define x 1)
(let ([x x])
  (λ (y)
    (let ([x y])
      (if x x x)))))
```



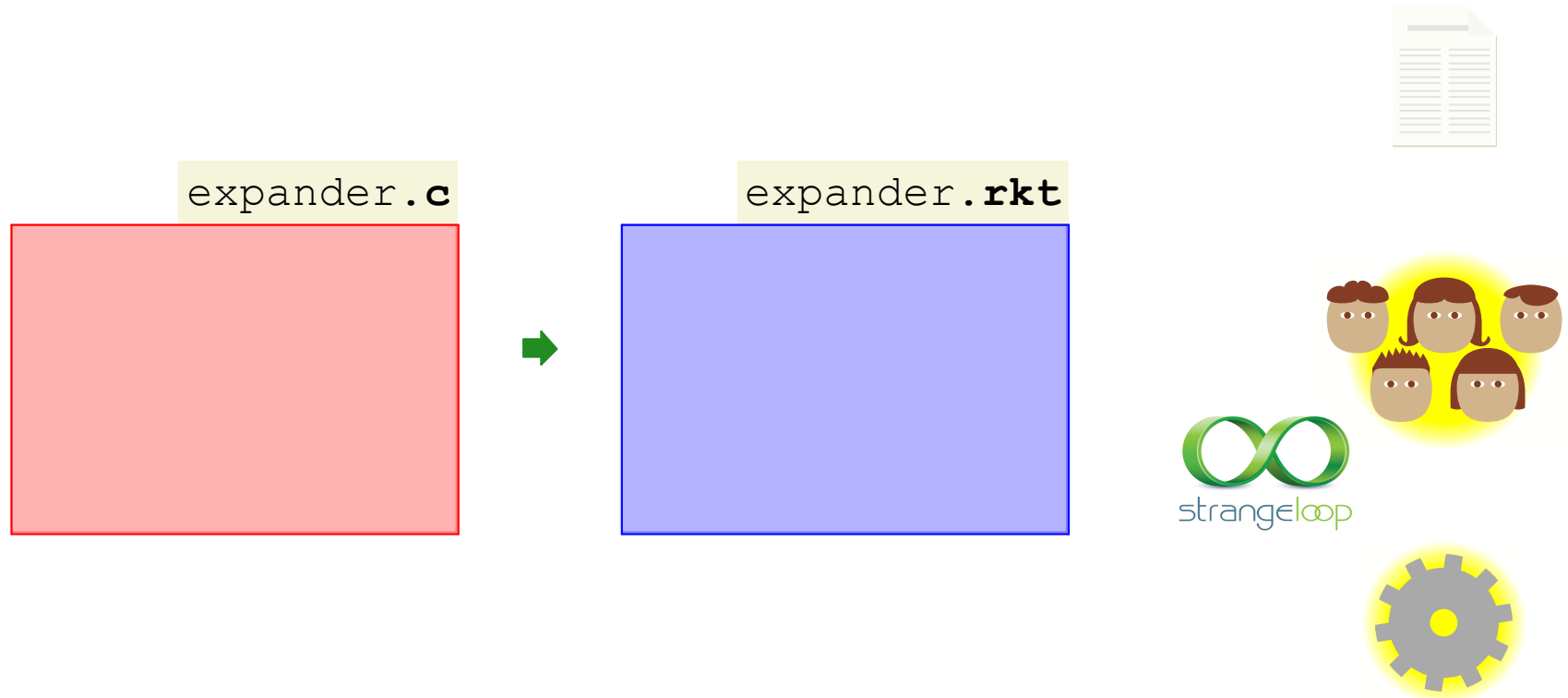
ca. 2016

Macros and Scope



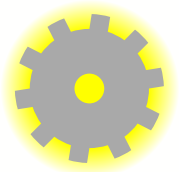
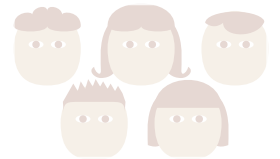
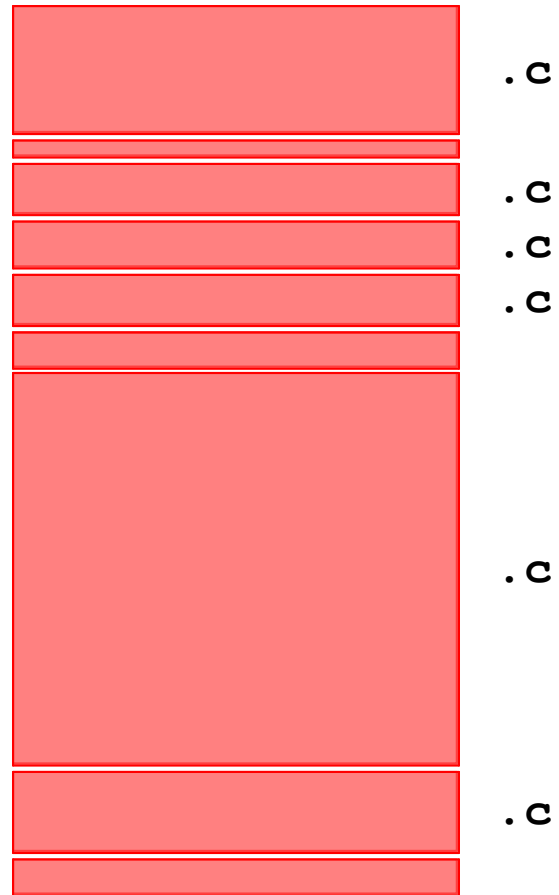
ca. 2016

Macros and Scope



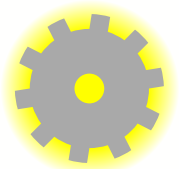
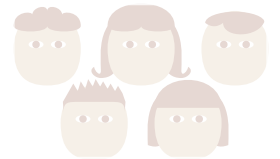
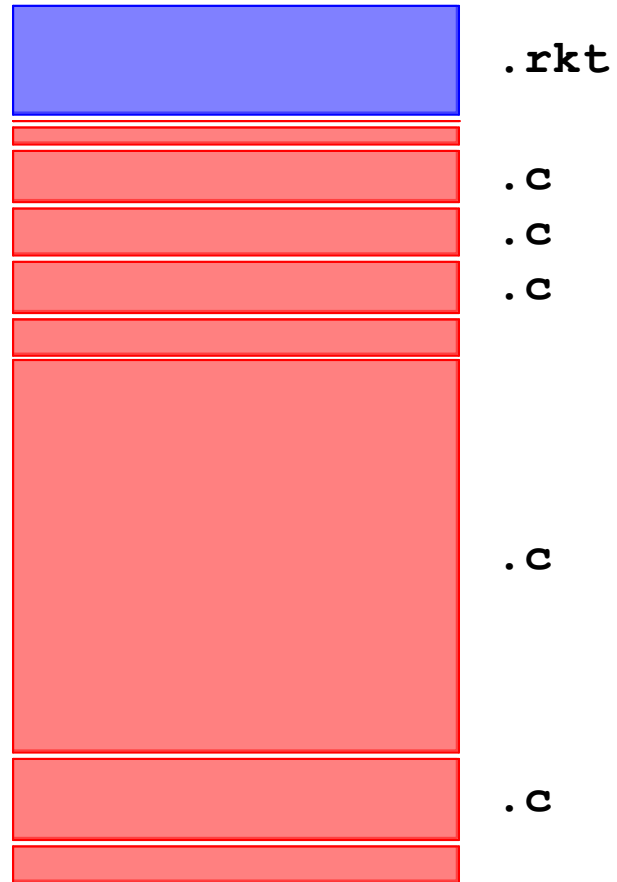
ca. 2019

Racket on Chez Scheme



ca. 2019

Racket on Chez Scheme



ca. 2019

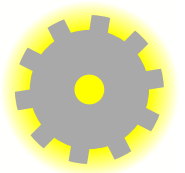
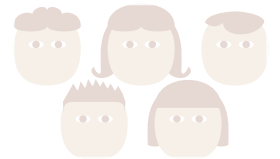
Racket on Chez Scheme



`.rkt`



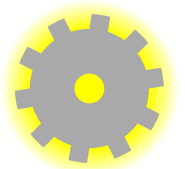
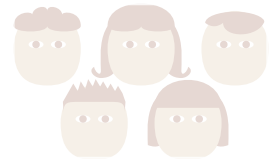
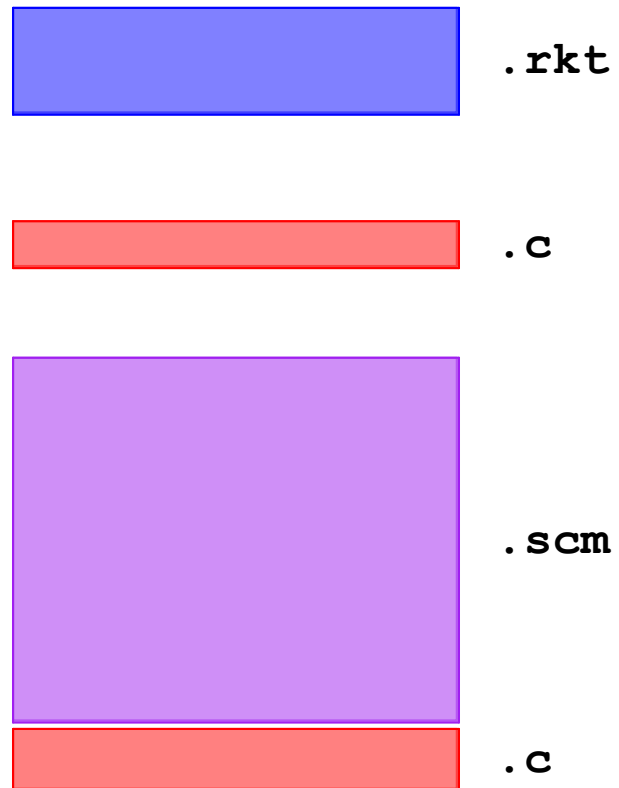
`.c`



ca. 2019

Racket on Chez Scheme

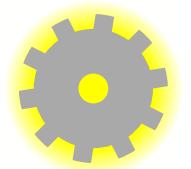
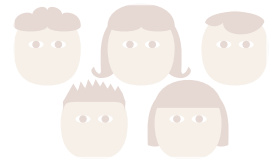
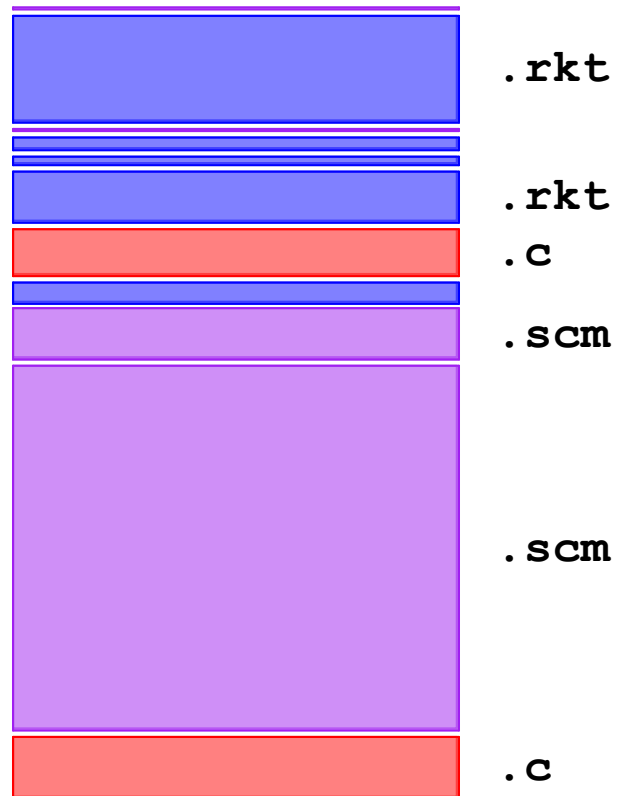
Chez Scheme



ca. 2019

Racket on Chez Scheme

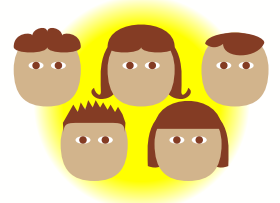
Racket on Chez Scheme



ca. 1984-2019

Chez Scheme

Dybvig et al.

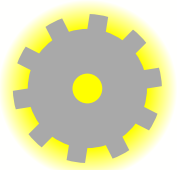
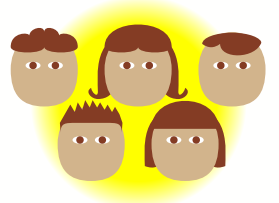
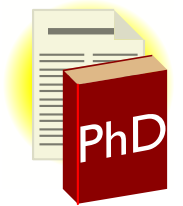
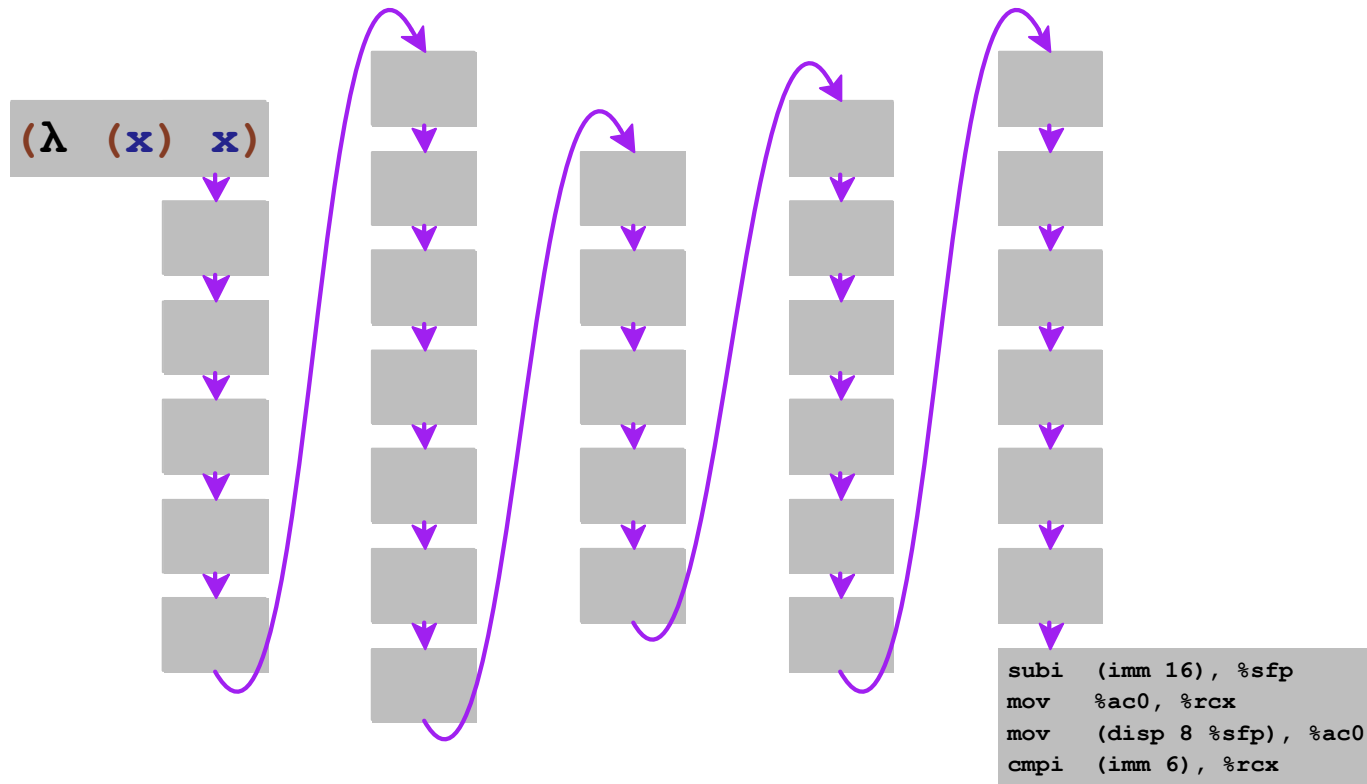


. scm

ca. 2012

Nanopass

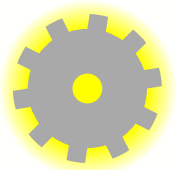
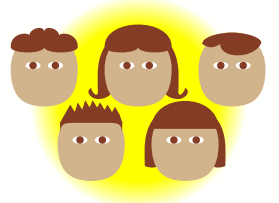
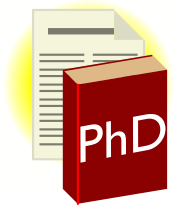
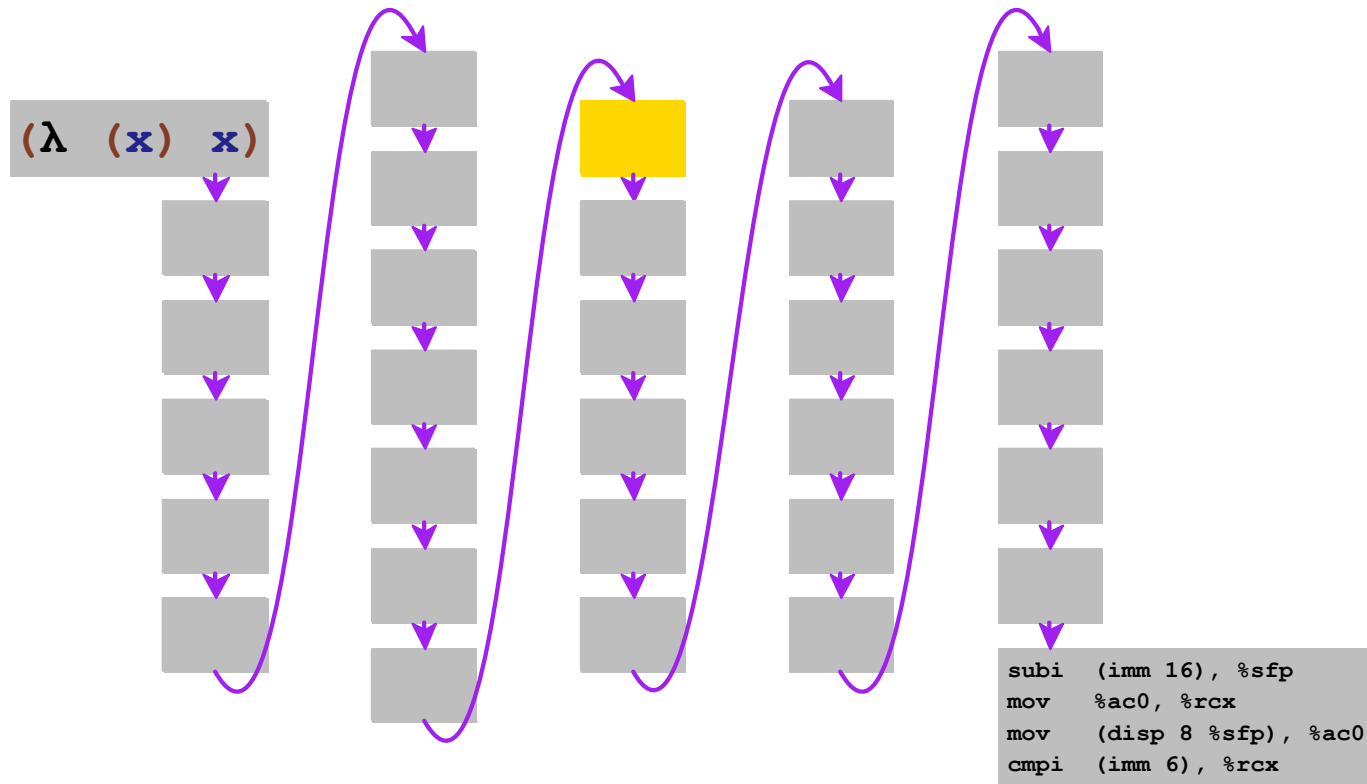
Keep



ca. 2012

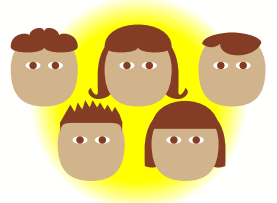
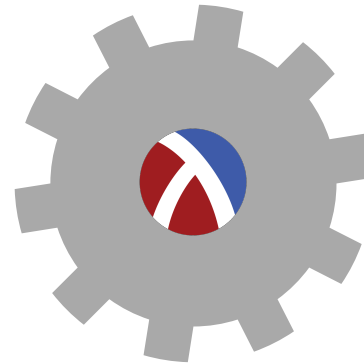
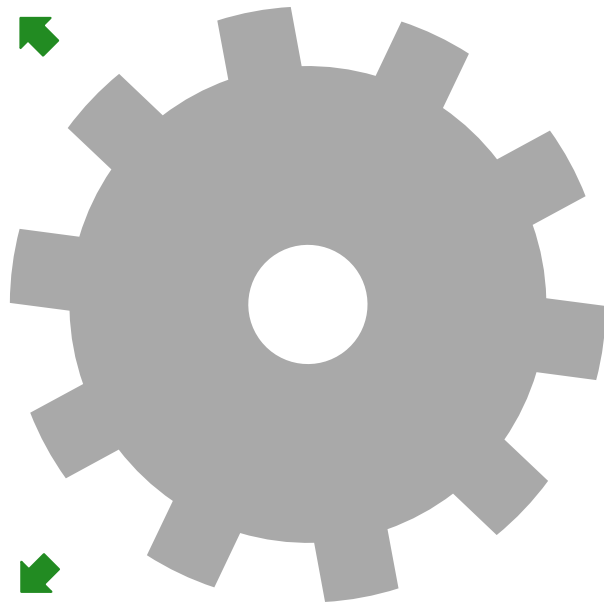
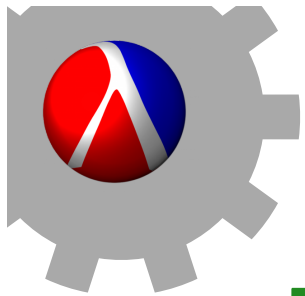
Nanopass

Keep



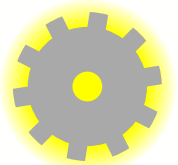
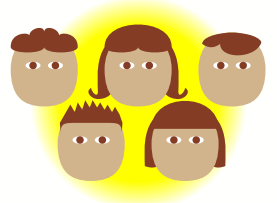
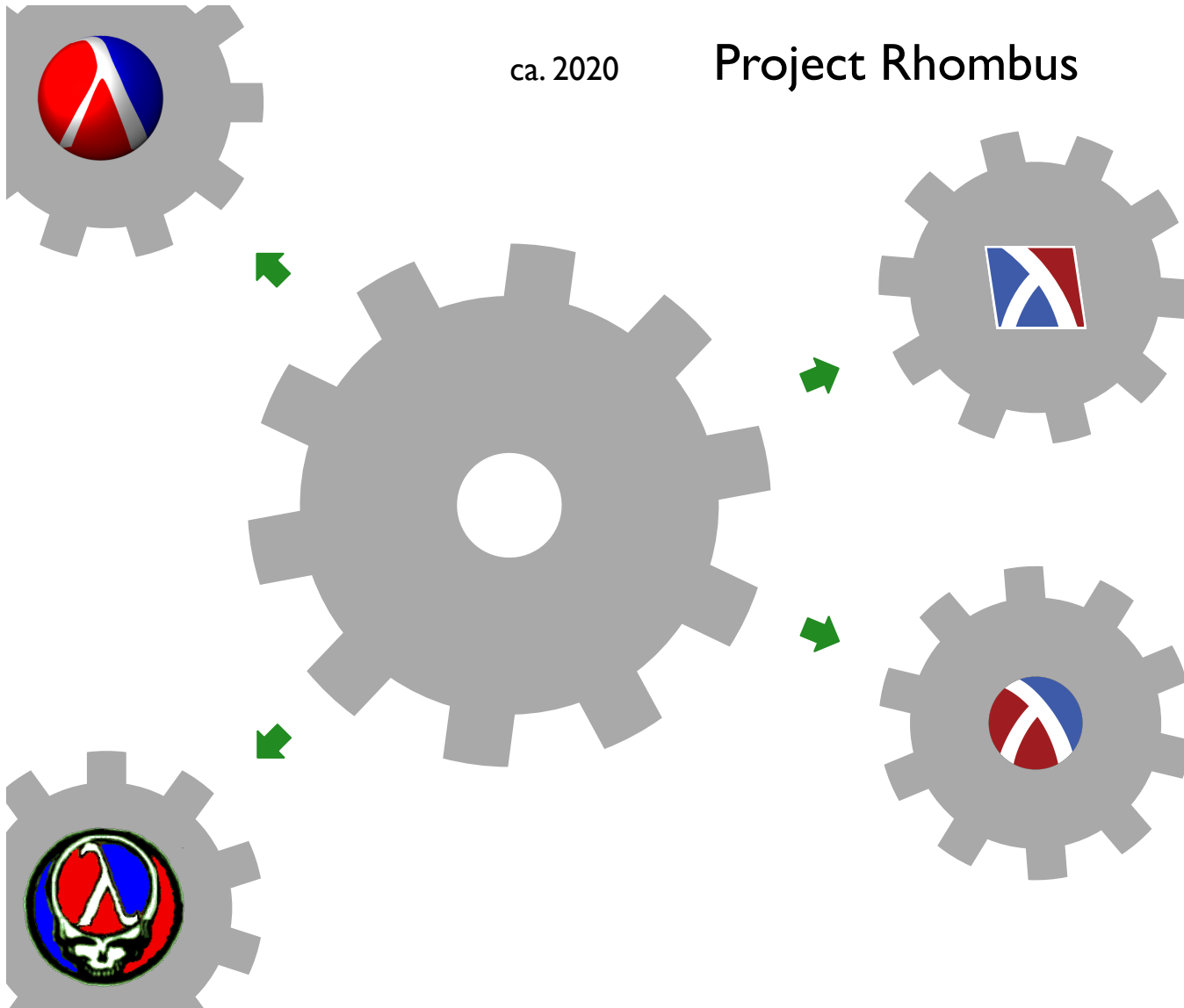
ca. 2020

Project Rhombus



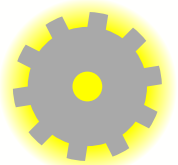
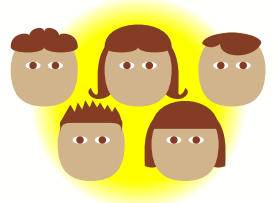
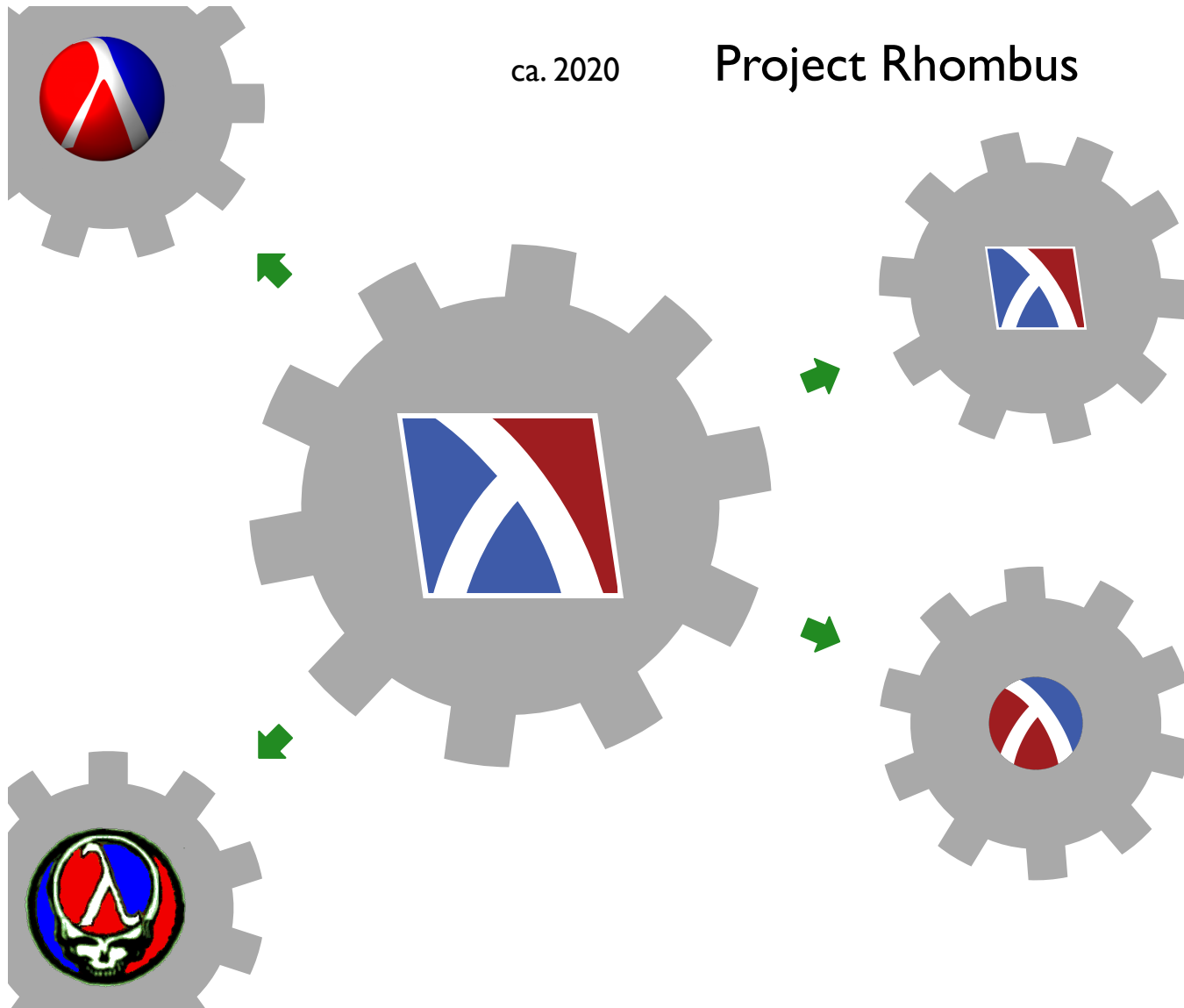
ca. 2020

Project Rhombus



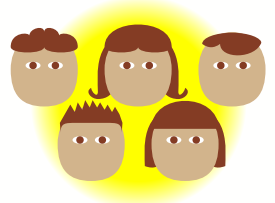
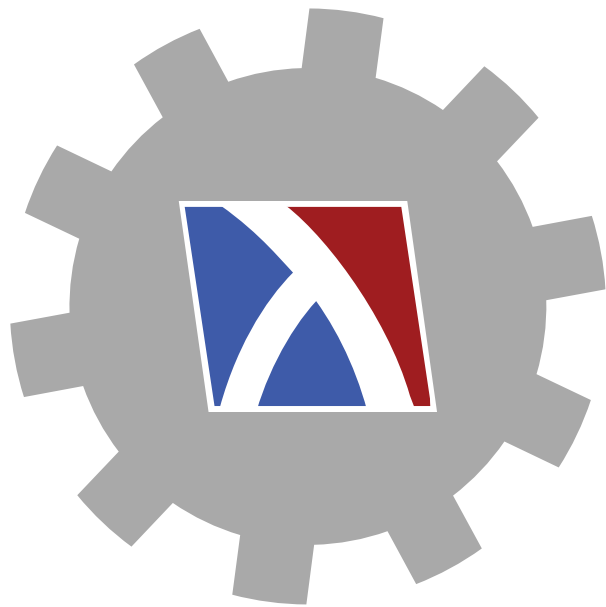
ca. 2020

Project Rhombus



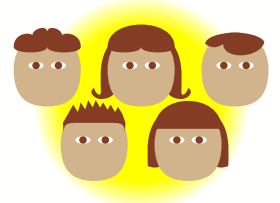
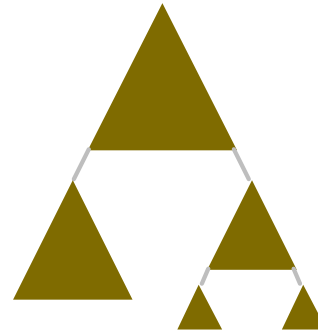
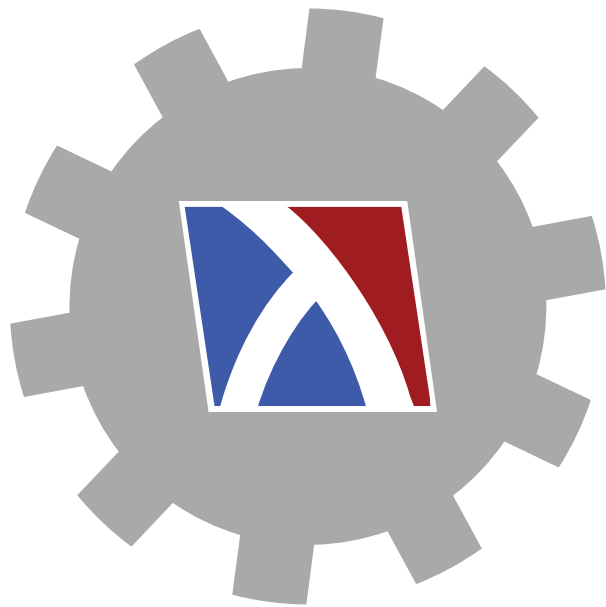
ca. 2020

Project Rhombus



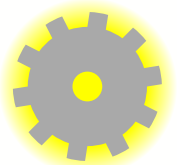
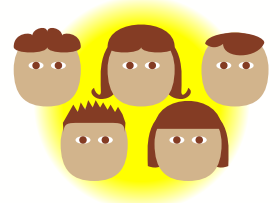
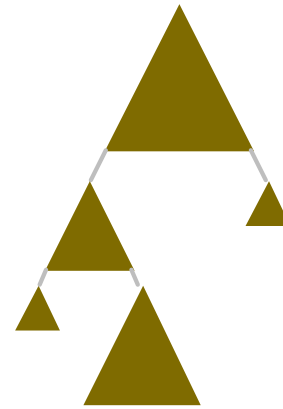
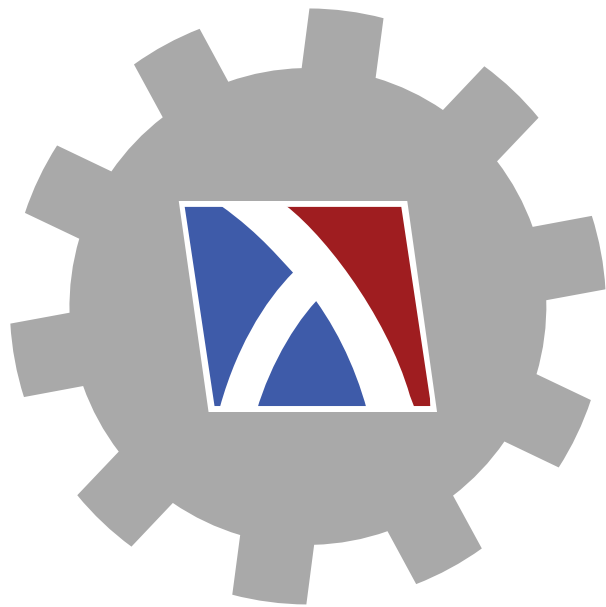
ca. 2020

Project Rhombus



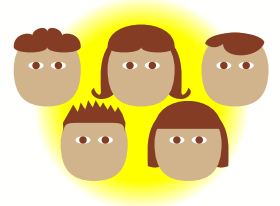
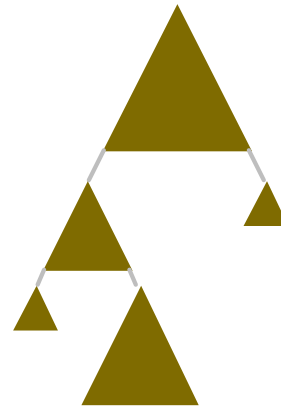
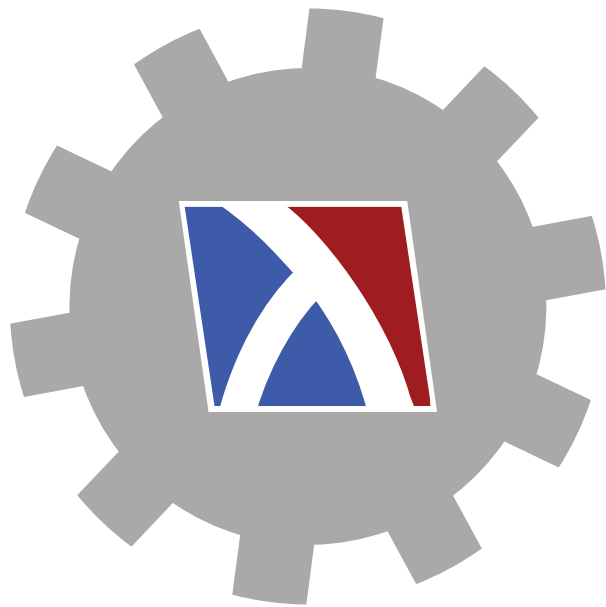
ca. 2020

Project Rhombus



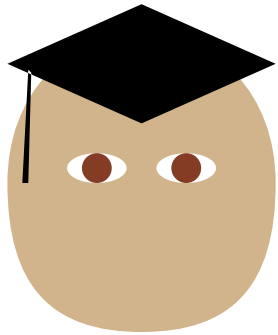
ca. 2020

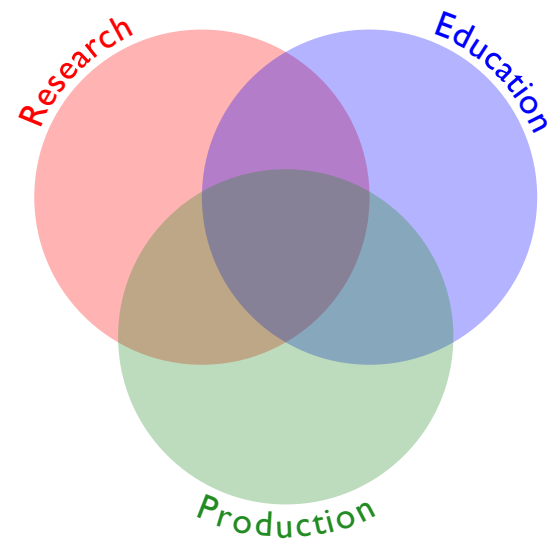
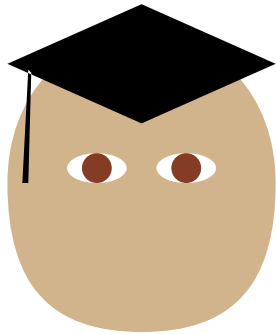
Project Rhombus

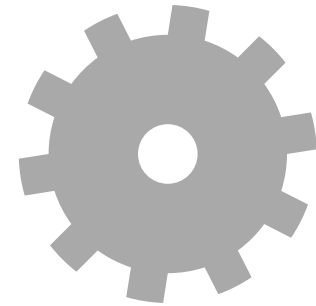
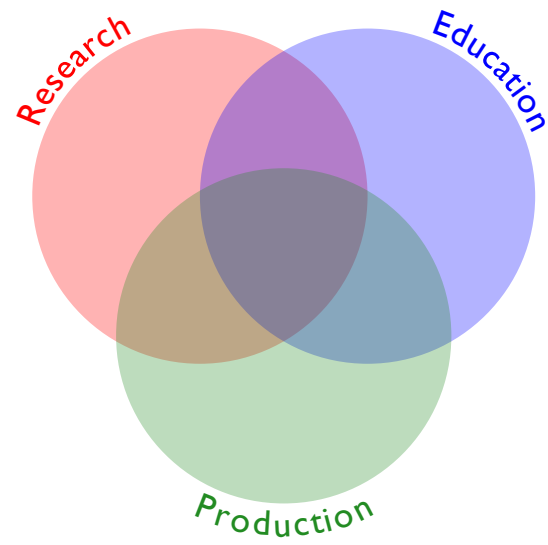
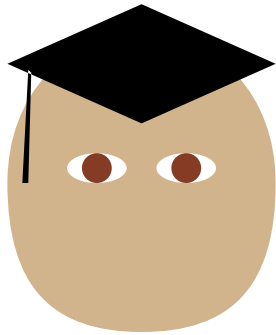


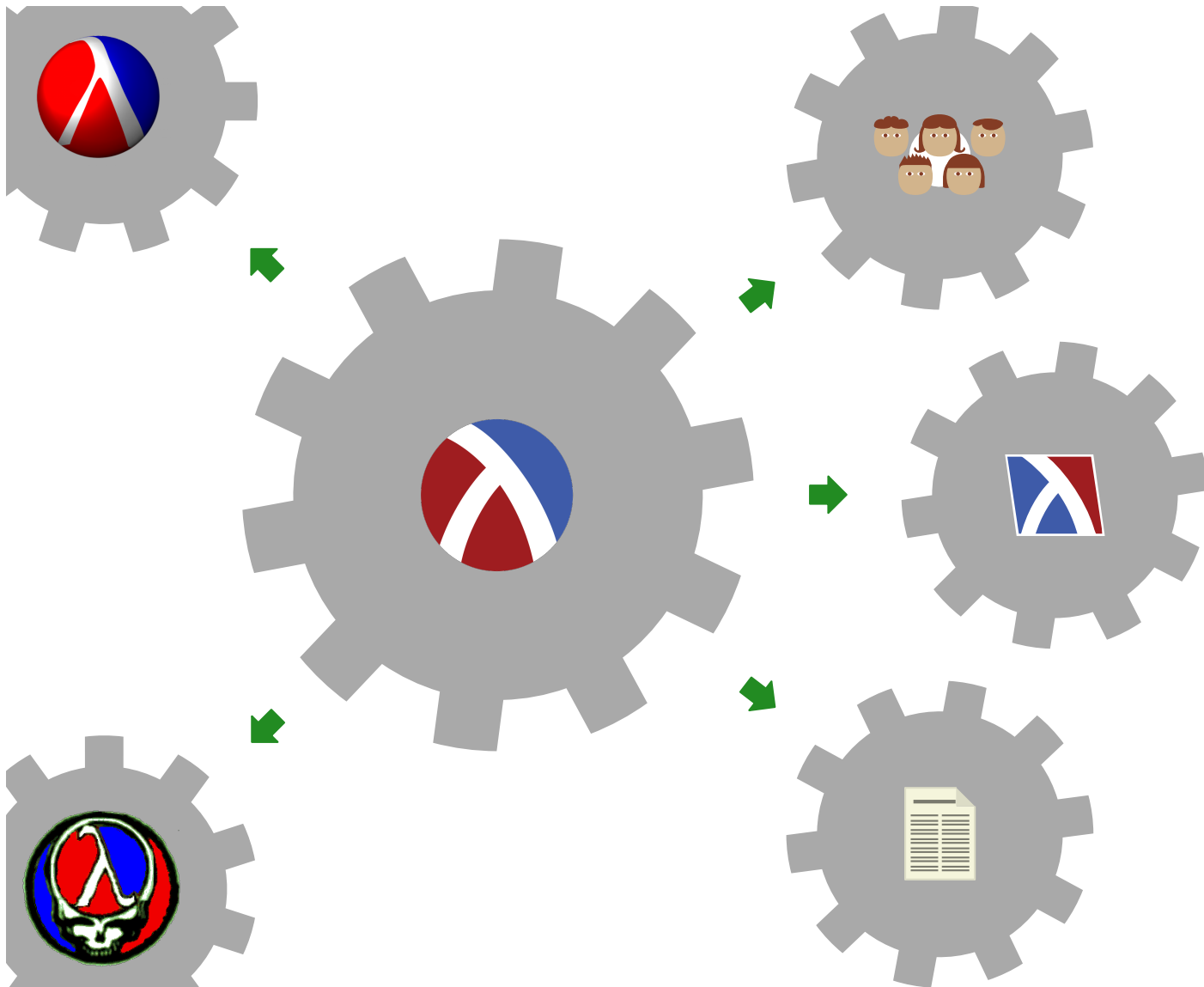
```
define fib(n):  
  match n  
    | 0: 0  
    | 1: 1  
    | n: fib(n-1) + fib(n-2)
```

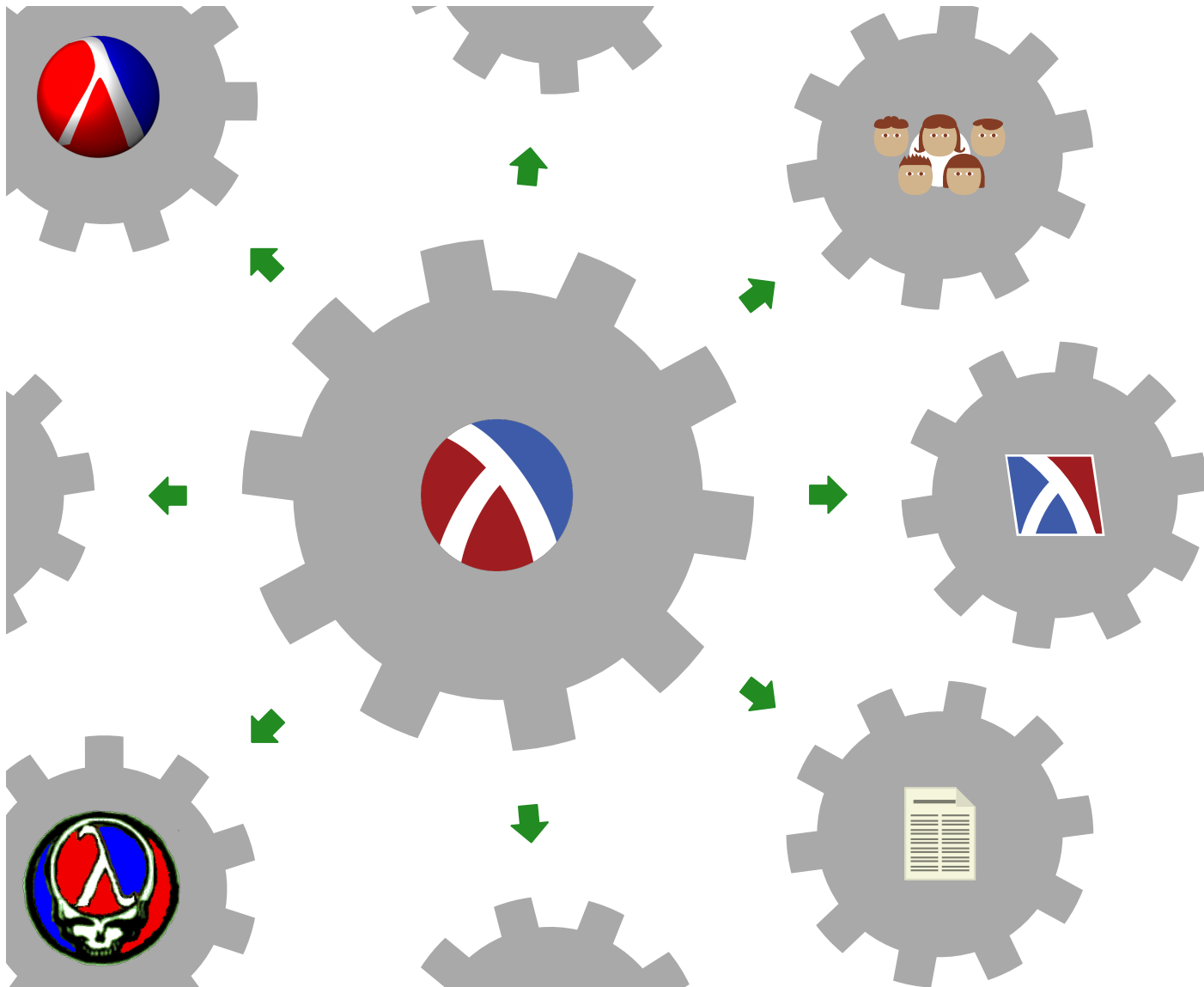


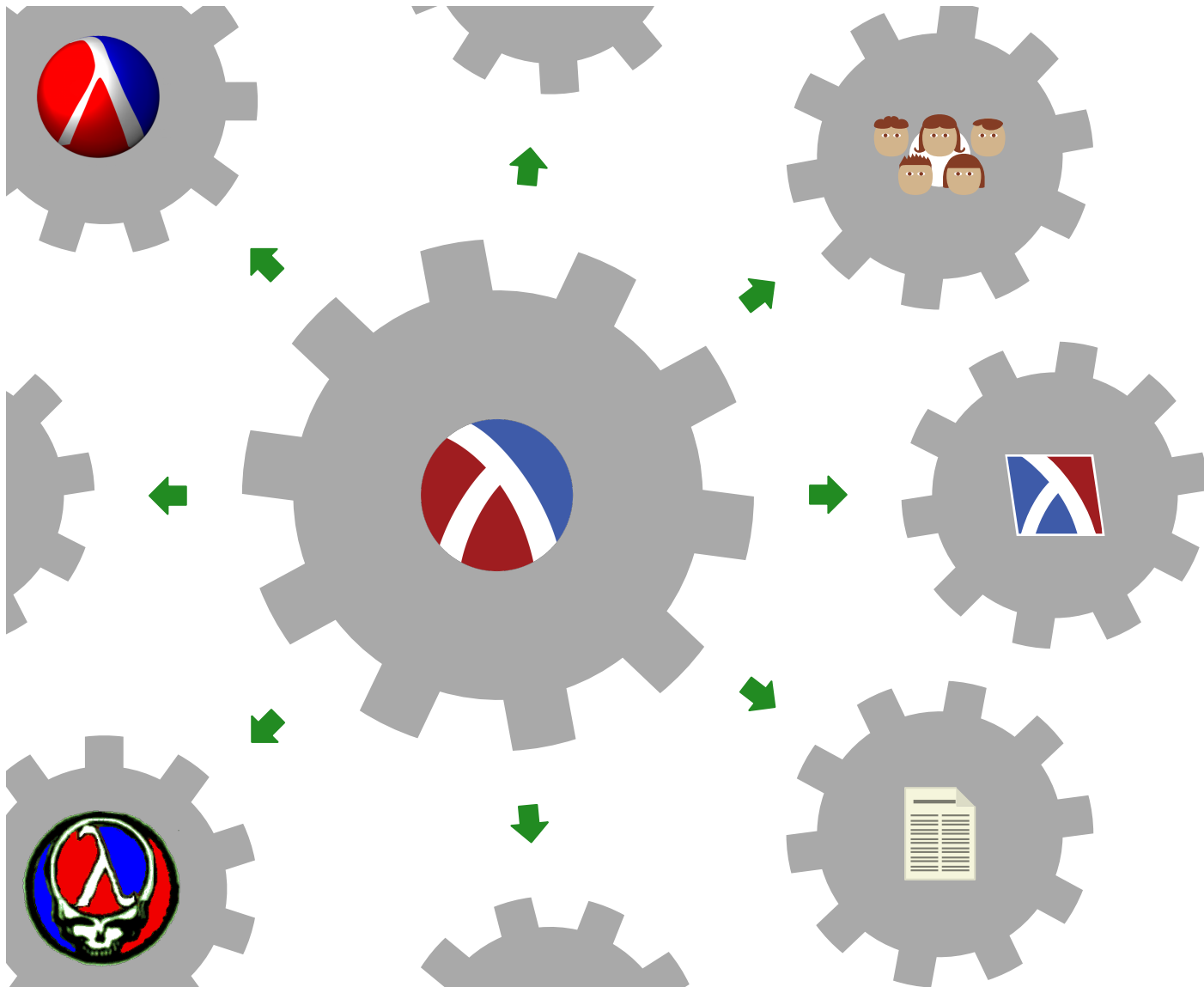












Thanks!