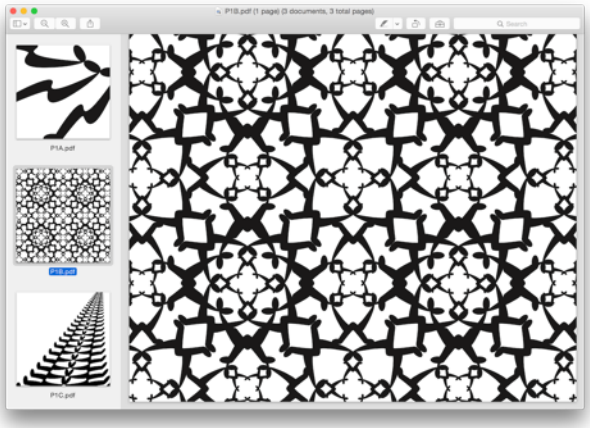
Bidirectional Programming with Direct Manipulation

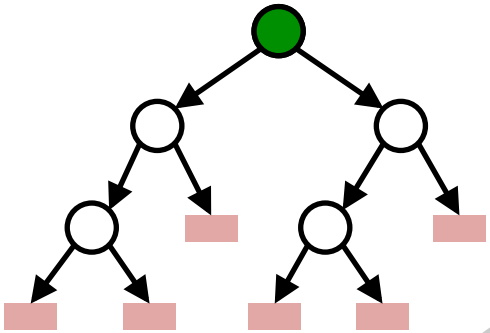
Ravi Chugh

THE UNIVERSITY OF CHICAGO

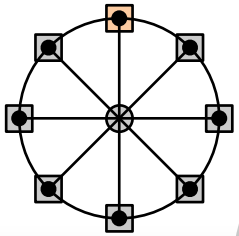
05.22.2019

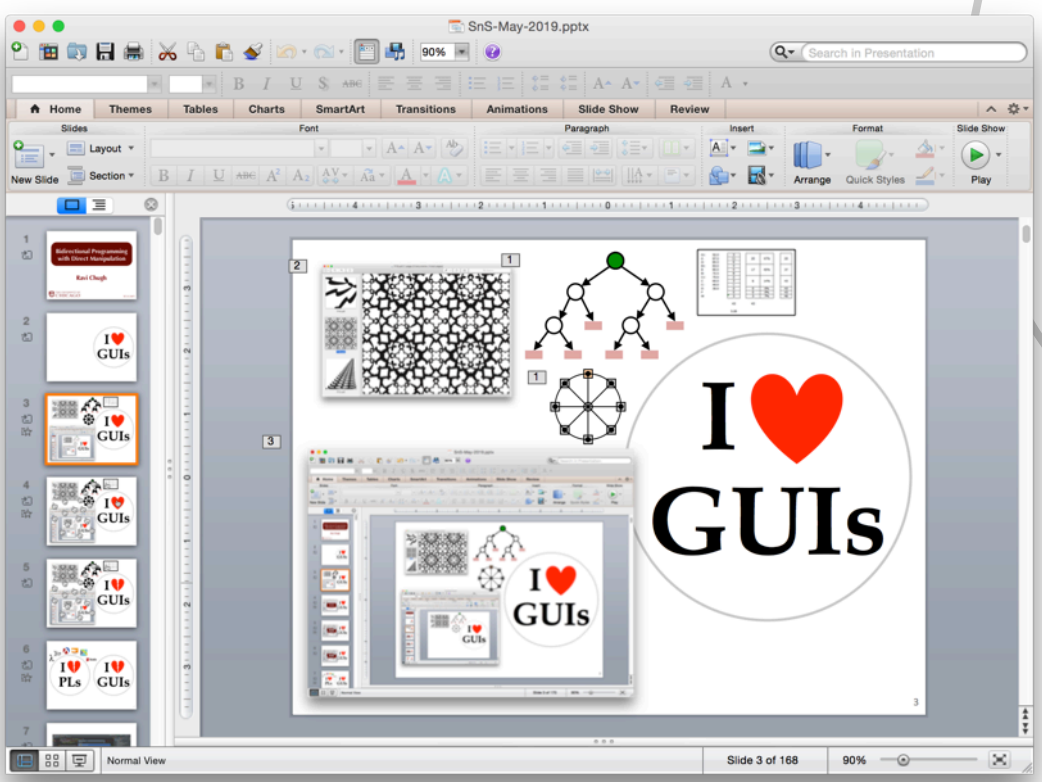

I
GUIs





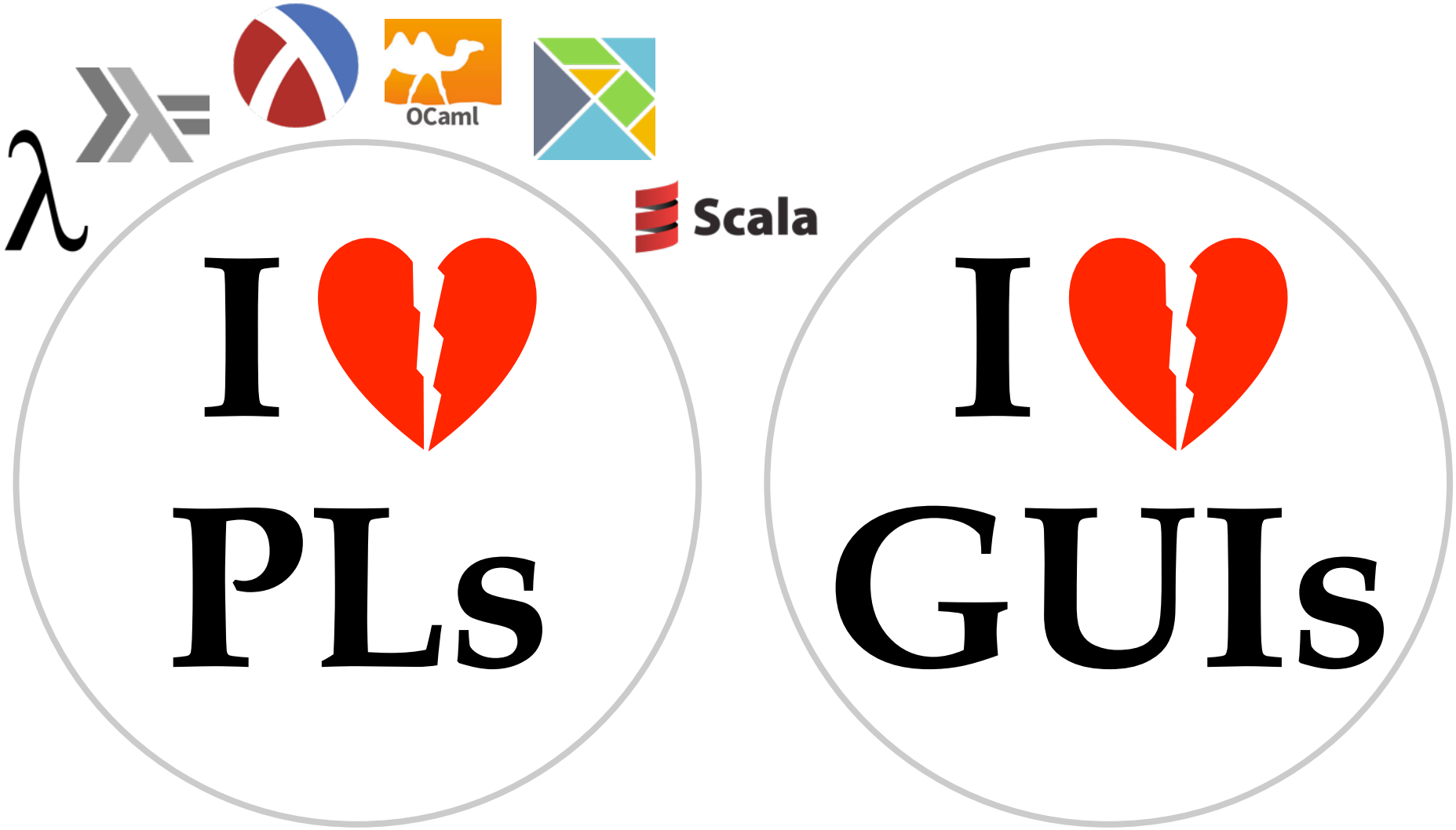
A+	92.0	3			
A	87.0	11	20	47%	20
A-	84.5	6			
B+	83.0	4			
B	80.0	5	17	40%	37
B-	72.5	8			
C+	70.0	2			
C	60.0	3	6	14%	43
C-	40.0	1			
D	30.0	0	0	0%	43
F		0	0	0%	43
W		3	3	7%	46
		43	43		
		3.26			






I
GUIs


I
GUIs
Slide 3 of 168
Normal View


OCaml
Scala
I
PLs
I
GUIs

djtp_first_steps › polls › tests.py

Django_test

tests.py | models.py | admin.py

```
        """
        create_question(question_text="Past question.", days=-30)
        response = self.client.get(reverse('polls:index'))
        self.assertQuerysetEqual(
            response.context['latest_question_list'
            ['<Question: Past question.>']
        )

    def test_index_view_with_a_future_question(self
        """
        Questions with a pub_date in the future sho
        the index page.
        """
        create_question(question_text="Future quest
        response = self.client.get(reverse('polls:i
        self.assertContains(response, "No polls are
                            status_code=200)
        self.assertQuerysetEqual(response.context['

    def test_index_view_with_future_question_and_pa
        """
        Even if both past and future questions exis
        should be displayed.
        """
        create_question(question_text="Past question.", days=-30)
        create_question(question_text="Future question.", days=30)
        response = self.client.get(reverse('polls:index'))
        self.assertQuerysetEqual(
            response.context['latest_question_list'],
            ['<Question: Past question.>']
        )
```

Search Everywhere: Include non-project items (Double ⇧)

result

Classes (⌘O)
- ResultsView (polls.views)

Files (⇧⌘O)
- results.html (djtp_first_steps/polls/templates/polls)

Symbols (⌥⌘O)
- result (FileReader (HTML5.js))
- result (StdSuites.AppleScript_Suite)
- result (e (Python 2.7.8 virtualenv at ~/django_p27))
- result (event (Python 2.7.8 virtualenv at ~/django_p27))
- result (event (Python 2.7.8 virtualenv at ~/django_p27))
- ... more

Actions (⇧⌘A)
- View Offline Inspection Results... Code
- Import Test Results

Database

- Django default
  - tables 13
    - auth_group
    - auth_group_permissions
    - auth_permission
    - auth_user
    - auth_user_groups
    - auth_user_user_permissions
    - django_admin_log
      - id INTEGER
      - action_time TEXT
      - object_id TEXT
      - object_repr TEXT
      - action_flag INTEGER
      - change_message TEXT
      - content_type_id INTEGER
      - user_id INTEGER
      - <unnamed> (id)
      - #FAKE_django_admin_log
      - #FAKE_django_admin_log
      - django_admin_log_417f1
      - django_admin_log_e8701
    - django_content_type
    - django_migrations

Debug Django_test

Debugger | Console

Frames: MainThread
- test_index_view_with_a_future_questi
- run, case.py:329
- __call__, case.py:393

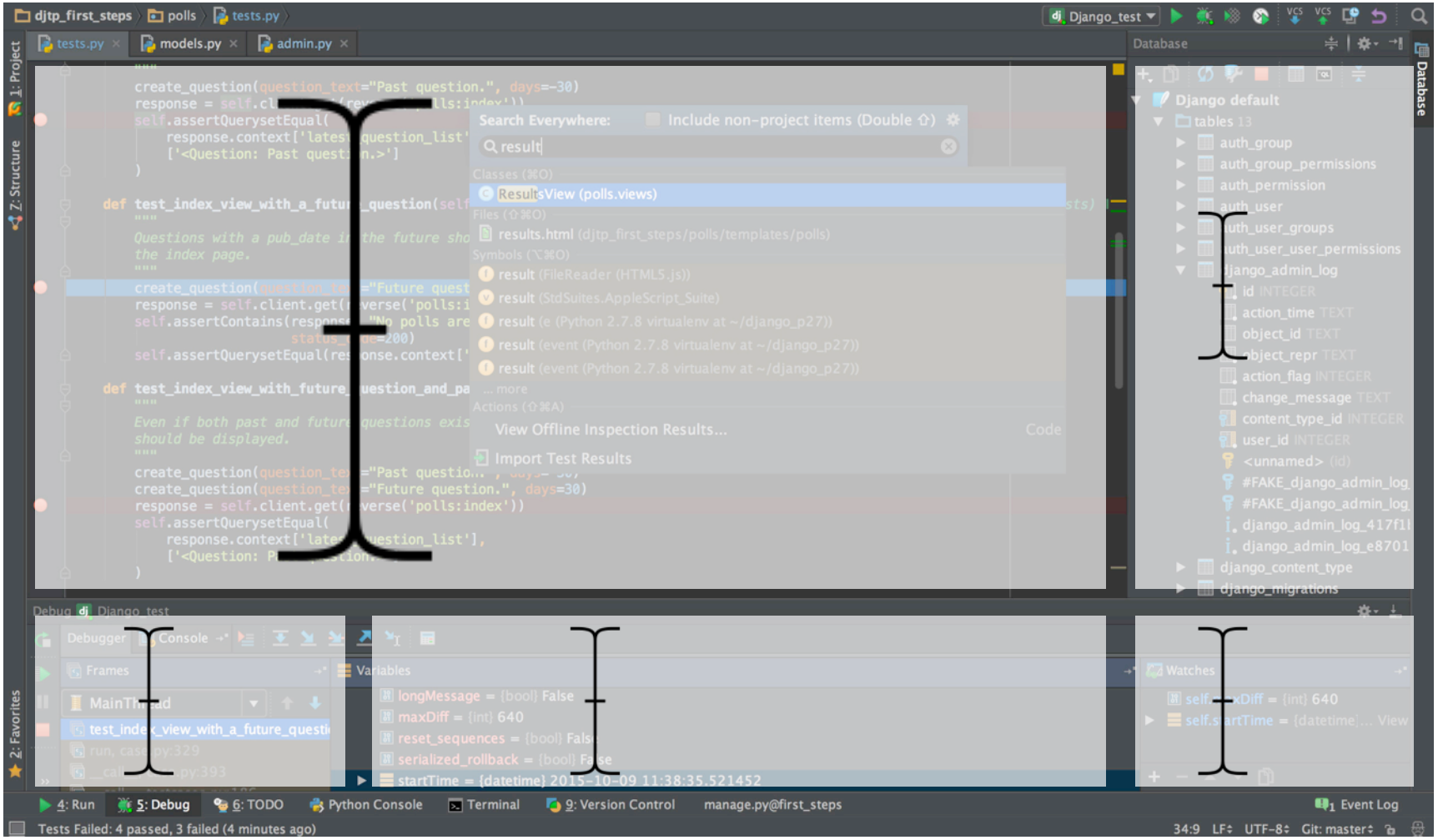
Variables
- longMessage = {bool} False
- maxDiff = {int} 640
- reset_sequences = {bool} False
- serialized_rollback = {bool} False
- startTime = {datetime} 2015-10-09 11:38:35.521452

Watches
- self.maxDiff = {int} 640
- self.startTime = {datetime}... View

4: Run | 5: Debug | 6: TODO | Python Console | Terminal | 9: Version Control | manage.py@first_steps | 1 Event Log

Tests Failed: 4 passed, 3 failed (4 minutes ago)

34:9 LF UTF-8 Git: master


djtp_first_steps
polls
tests.py
Django_test
tests.py
models.py
admin.py
Database
Search Everywhere:
Include non-project items (Double ⇧)
result
Classes (⌘O)
ResultsView (polls.views)
Files (⇧⌘O)
results.html (djtp_first_steps/polls/templates/polls)
Symbols (⌥⌘O)
result (FileReader (HTML5.js))
result (StdSuites.AppleScript_Suite)
result (e (Python 2.7.8 virtualenv at ~/django_p27))
result (event (Python 2.7.8 virtualenv at ~/django_p27))
result (event (Python 2.7.8 virtualenv at ~/django_p27))
... more
Actions (⇧⌘A)
View Offline Inspection Results...
Code
Import Test Results
Debug Django_test
Debugger
Console
Frames
Variables
Watches
longMessage = {bool} False
maxDiff = {int} 640
reset_sequences = {bool} False
serialized_rollback = {bool} False
startTime = {datetime} 2015-10-09 11:38:35.521452
Tests Failed: 4 passed, 3 failed (4 minutes ago)


I
PLs


I
GUIs

	PLs	GUIs
2D Graphics		 
Documents		 
Presentations		 
Spreadsheets		 
Web Apps		 

…

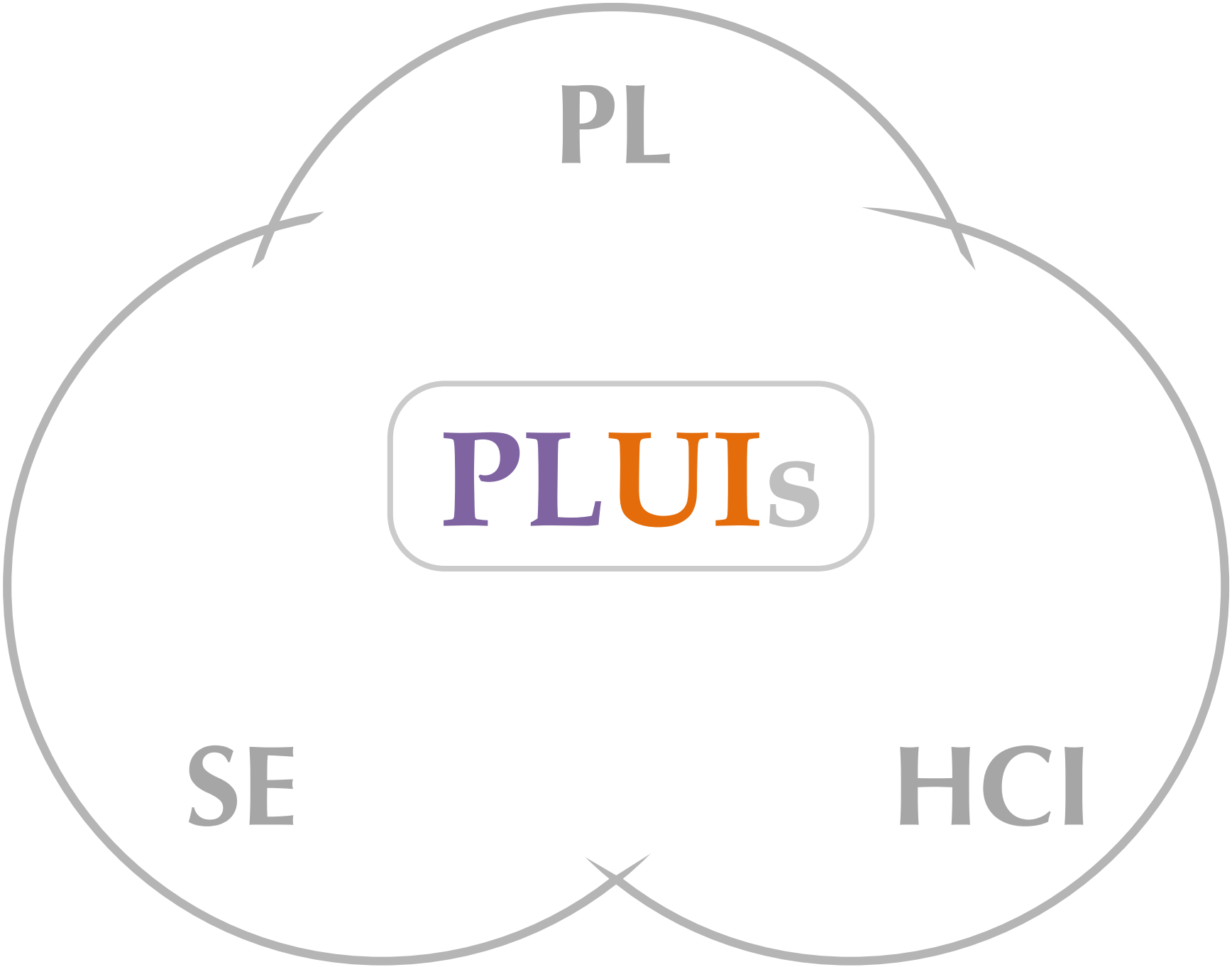

I
PLs


I
GUIs

New PL Features?

PLUIs

New UI Features?


PL
PLUIs
SE
HCI

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

Bidirectional Evaluation for Program Repair
[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis
[POPL 2019, work-in-progress]

PLUIs

Lightweight Structured Text Editing
[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis
[UIST 2016, work-in-progress]

Edit Program

Edit Program; Run

Edit Program; Run; View Output

Edit Program; Run; View Output

Edit Program; Run; View Output

Edit Program; Run; View Output

(Edit Program; Run; View Output)+

Edit Program; Run; View Output

Edit Program; Run; View Output

Edit Program; Run; View Output

Edit Program; Run; View Output

Edit Program; Run; View Output

Edit Program; Run; View Output

(Edit Program; Run; View Output; Edit Output)+

Synthesize Program Repair

Our Goal

```
let f x = x + " 2018" in
```

Our Goal

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

Our Goal

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

Our Goal

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

⇒

"OOPS 2018
@ SPLASH 2018"

⇝

"OOPSLA 2019
@ SPLASH 2018"

Our Goal

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

⇒

```
"OOPS 2018
@ SPLASH 2018"
```

⇝

```
"OOPSLA 2019
@ SPLASH 2018"
```

⇐

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

Our Goal

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

⇒

```
"OOPS 2018
@ SPLASH 2018"
```

⇝

```
"OOPSLA 2019
@ SPLASH 2018"
```

⇐

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

Our Goal

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

⇒

```
"OOPS 2018
@ SPLASH 2018"
```

⇝

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

⇒

```
"OOPSLA 2019
@ SPLASH 2019"
```

Lenses

$$
\begin{tikzcd}
f\ x \arrow[r, Rightarrow, "\text{"get"}"] & y \arrow[d, squiggly] \\
f\ x' & y' \arrow[l, Rightarrow, "\text{"put"}"']
\end{tikzcd}
$$

1. Function f in DSL or restricted style (i.e. point-free)

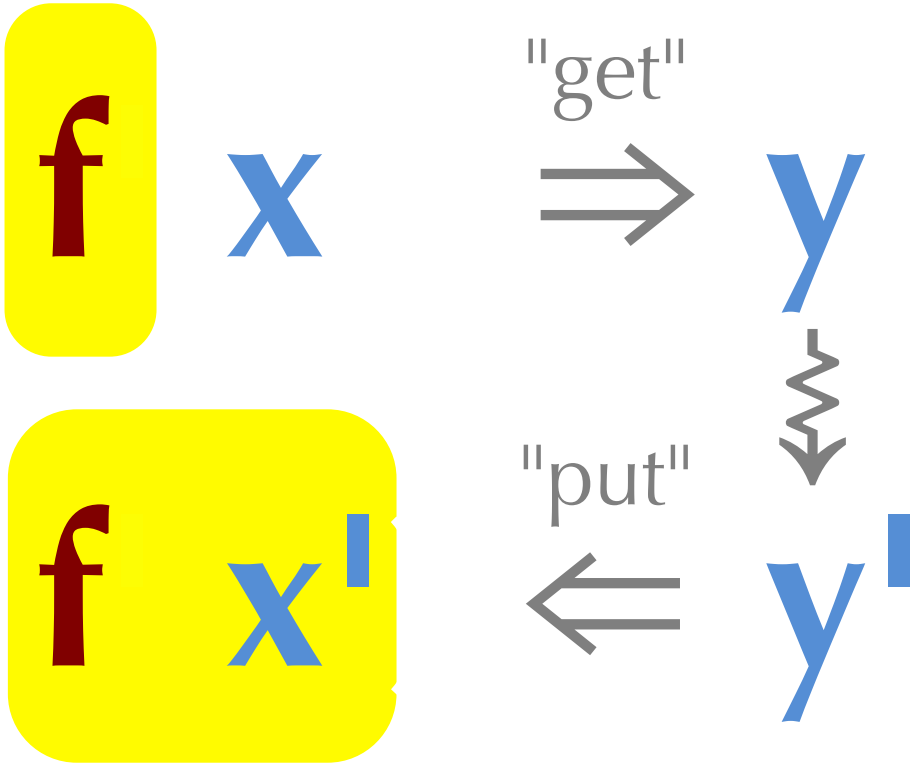
$$\boxed{f}\ x \xRightarrow{\text{"get"}} y$$

$$\rightsquigarrow$$

$$f\ x' \xLeftarrow{\text{"put"}} y'$$

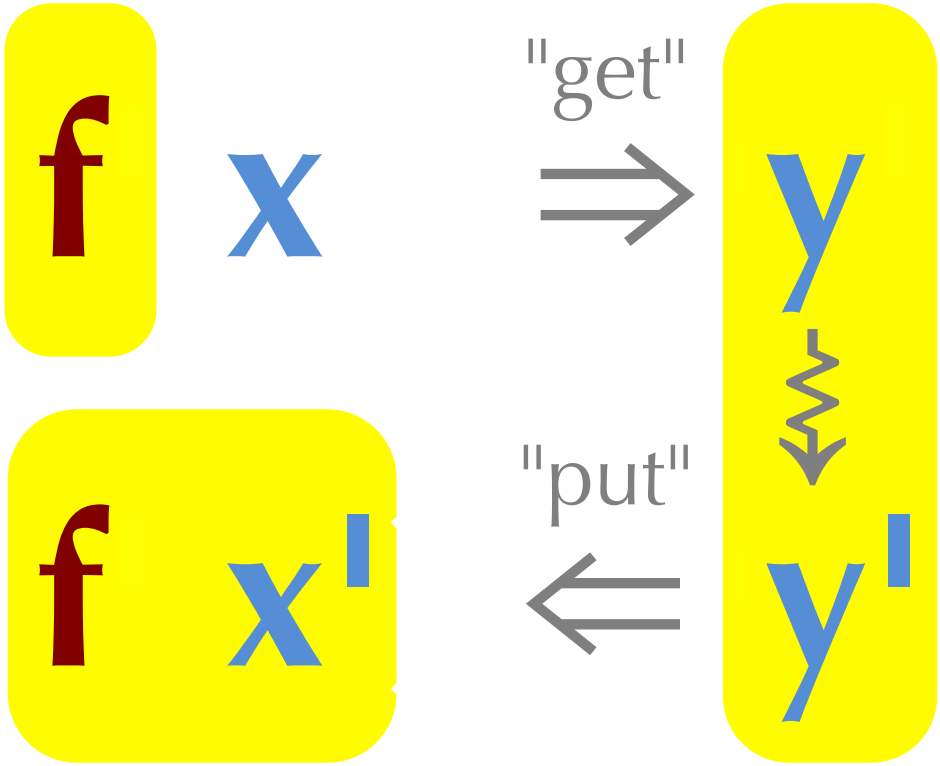
1. Function f in DSL
or restricted style
(i.e. point-free)

2. Update
"data" (x)
but not
"code" (f)



Lenses

1. Function f in DSL
or restricted style
(i.e. point-free)



2. Update
"data" (x)
but not
"code" (f)

3. Restricted structural
changes between y and y'

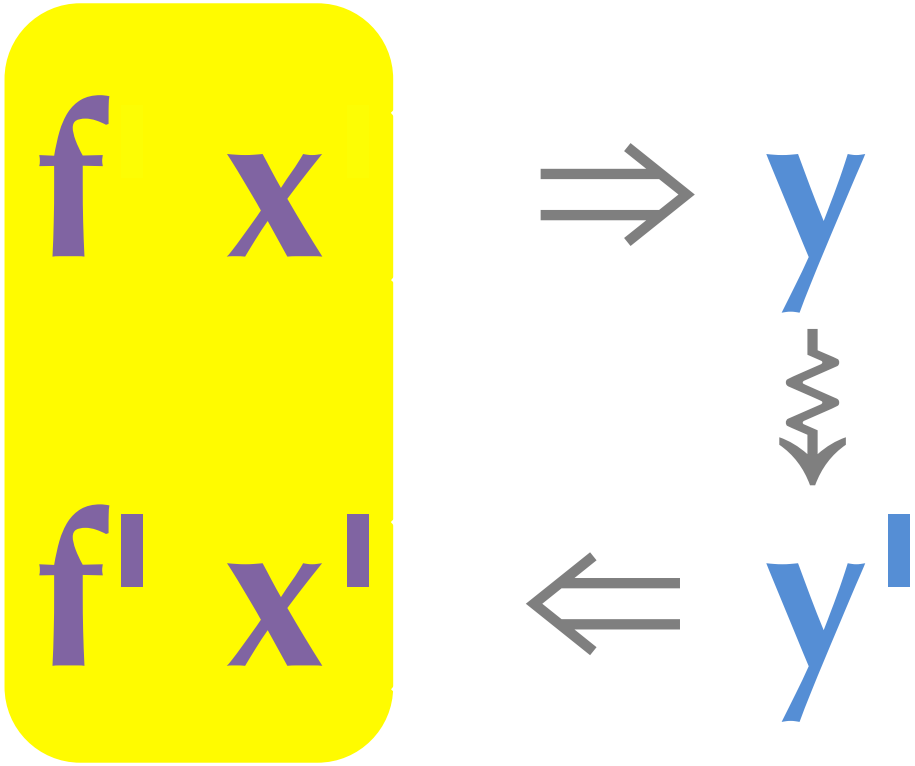
$$
\begin{tikzcd}
f\ x \arrow[r, Rightarrow] & y \arrow[d, squiggly] \\
f\ x' & y' \arrow[l, Rightarrow]
\end{tikzcd}
$$

Bidirectional Evaluation

$$\begin{array}{ccccc} f & x & \Rightarrow & y \\ & & & \rightsquigarrow \\ f & x' & \Leftarrow & y' \end{array}$$

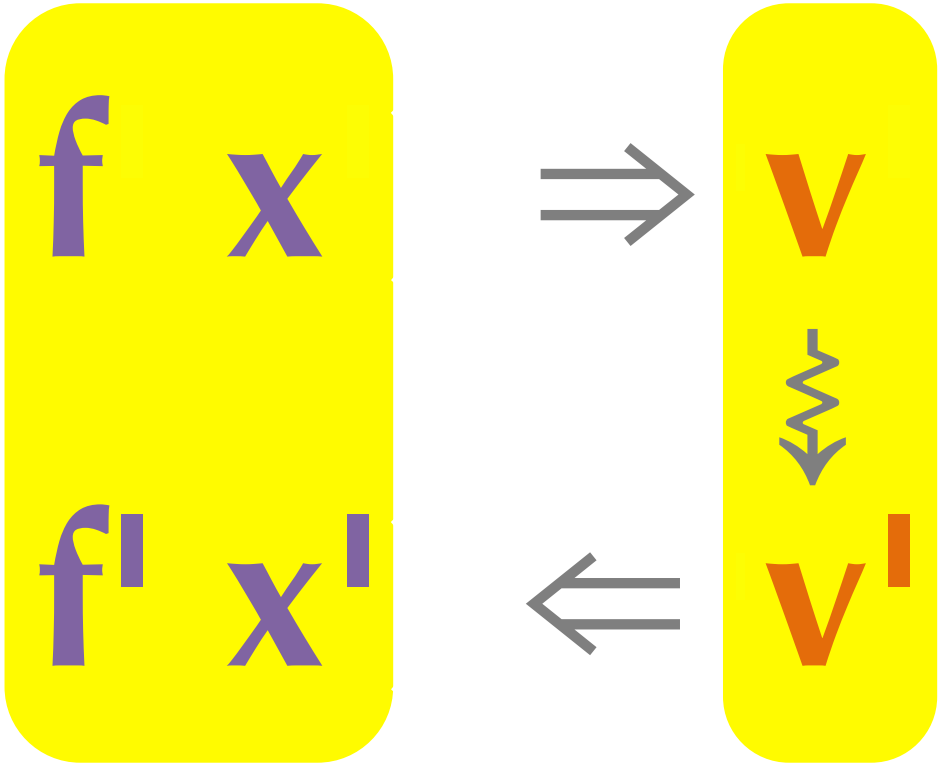
Bidirectional Evaluation

All expressions
(**code** + **data**)
can be run
backwards
and updated



Bidirectional Evaluation

All expressions
(**code** + **data**)
can be run
backwards
and updated



"Small" structural changes to **values**
+ **user-defined lenses** for customization

Bidirectional Evaluation

$$
\begin{array}{ccc}
e & \Rightarrow & v \\
 & & \rightsquigarrow \\
e' & \Leftarrow & v'
\end{array}
$$

Bidirectional Evaluation

$$(\mathbf{E} \vdash \mathbf{e}) \Rightarrow \mathbf{v}$$

$$\rightsquigarrow$$

$$(\mathbf{E}' \vdash \mathbf{e}') \Leftarrow \mathbf{v}'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

$\rightsquigarrow$

```
"OOPSLA 2019
@ SPLASH 2018"
```

$\Leftarrow$

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

$\rightsquigarrow$

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

$\Leftarrow$

```
"OOPSLA 2019
@ SPLASH 2018"
```

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash e_1\ e_2) \Rightarrow v$$
$$\rightsquigarrow$$
$$v'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E\ldots \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$v'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f\ ,\ x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$v'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$\frac{(E_f \vdash e_f)}{((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v}$$

$$\rightsquigarrow$$

$$v'$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f)$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \rightsquigarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \rightsquigarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$v'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E, f \mapsto \quad , x \mapsto \quad) \vdash f\ x) \Leftarrow v'$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E, f \mapsto [E_f']\ \lambda y.e_f', x \mapsto \quad) \vdash f\ x) \Leftarrow v'$$

New Code

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E, f \mapsto [E_f']\ \lambda y.e_f', x \mapsto v_x') \vdash f\ x) \Leftarrow v'$$

New Code **New Data**

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

$\rightsquigarrow$

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

$\Leftarrow$

```
"OOPSLA 2019
@ SPLASH 2018"
```

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

Merge: $E_1 \oplus_{\text{conservative}} E_2$

$(\text{year} \mapsto 2018) \oplus_{\text{conservative}} (\text{year} \mapsto 2019)$ undefined

Enables a "PutGet" theorem

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

Merge: $E_1 \oplus_{\text{conservative}} E_2$

$(\text{year} \mapsto 2018) \oplus_{\text{conservative}} (\text{year} \mapsto 2019)$ undefined

Enables a "PutGet" theorem

Merge: $E_1 \oplus_{\text{optimistic}} E_2$

$(\text{year} \mapsto 2018) \oplus_{\text{optimistic}} (\text{year} \mapsto 2019) = 2019$

Enables many desirable interactions

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

$\rightsquigarrow$

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

$\Leftarrow$

```
"OOPSLA 2019
@ SPLASH 2018"
```

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

$\rightsquigarrow$

```
"OOPSLA 2019
@ SPLASH 2018"
```

$\Leftarrow$

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

```
let f x = x + " 2018" in
f "OOPS"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPS 2018
@ SPLASH 2018"
```

$\rightsquigarrow$

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

$\Rightarrow$

```
"OOPSLA 2019
@ SPLASH 2019"
```

Key Ideas

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

Propagate updates ***through*** function calls

Allow merge conflicts

```
let f x = x + " 2019" in
f "OOPSLA"
  + "\n@ " + f "SPLASH"
```

⇐

```
"OOPSLA 2019
@ SPLASH 2018"
```

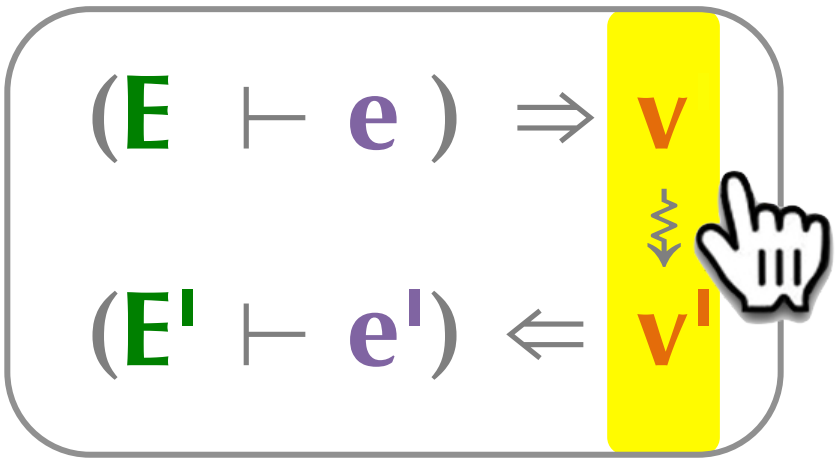
"Bidirectional Evaluation...

$$(E \vdash e) \Rightarrow v$$

$$\rightsquigarrow$$

$$(E' \vdash e') \Leftarrow v'$$

"Bidirectional Evaluation… with Direct Manipulation"



Sketch-n-Sketch

Sketch-n-Sketch File View Options ◂ Previous Example Next Example ▸

Current file: *Untitled (1b: Table of States) **

Undo Redo Run ▸

```
18  let headerRow =
19    let styles = [padding] in
20    Html.tr [] [] (List.map (Html.th styles []) headers)
21  in
22  let stateRows =
23    let colors = ["yellow", "white"] in
24    let drawRow i row =
25      let color = List.nth colors (mod i
26      let columns =
27        List.map
28          (Html.td [padding, ["background-color", color]] [])
```

Output Editor
Update Program ▸
L23 Inserted [ye], L23 Replaced [ightgray] by [low]
Revert to Original Program

State	Capital
Alabama	Montgomery, AL
Alaska	Juneau, AK
Arizona	Phoenix, AZ
Arkansas	Little Rock, AR
California	Sacramento, CA
Colorado	Denver, CO
Connecticut	Hartford, CT

GUI
HTML
Value
Auto Sync

Elements Console Sources Network Performance Memory Application Security Audits

```
▸<tr data-value-id="30" style>…</tr>
▸<tr data-value-id="35" style>…</tr>
▾<tr data-value-id="40" style>
  <td contenteditable="true" data-value-id="37" style="padding:3px;background-color:
  yellow">Connecticut</td>
  <td contenteditable="true" data-value-id="39" style="padding:3px;background-color:
  yellow">Hartford, CT</td> == $0
 </tr>
</table>
<svg id="svgWidgetsLayer" style="left: 730px; top: 42px; width: 306px; height: 308px;">
</svg>
</div>
<div class="output-panel-warning" style="top: -1px; right: -1px; bottom: -1px; left: -1px;">
```

html body div div.work-area div.main-panels div.panel.output-panel div#outputCanvas table tr td

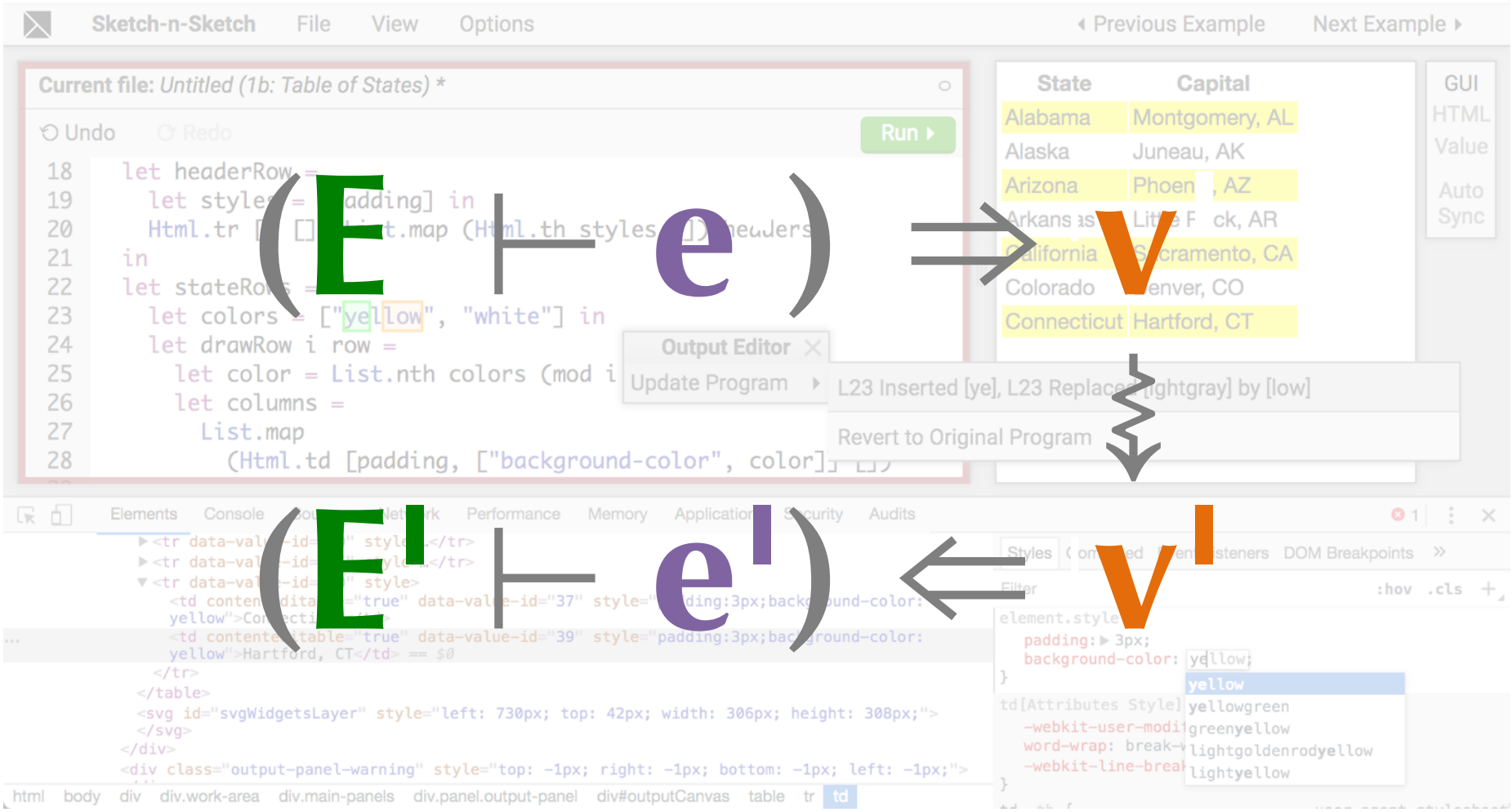
Styles Computed Event Listeners DOM Breakpoints »

Filter :hov .cls +

```
element.style {
  padding: ▸ 3px;
  background-color: yellow;
}
td[Attributes Style]
  -webkit-user-modi
  word-wrap: break-
  -webkit-line-brea
}
```

yellow
yellowgreen
greenyellow
lightgoldenrodyellow
lightyellow

Sketch-n-Sketch

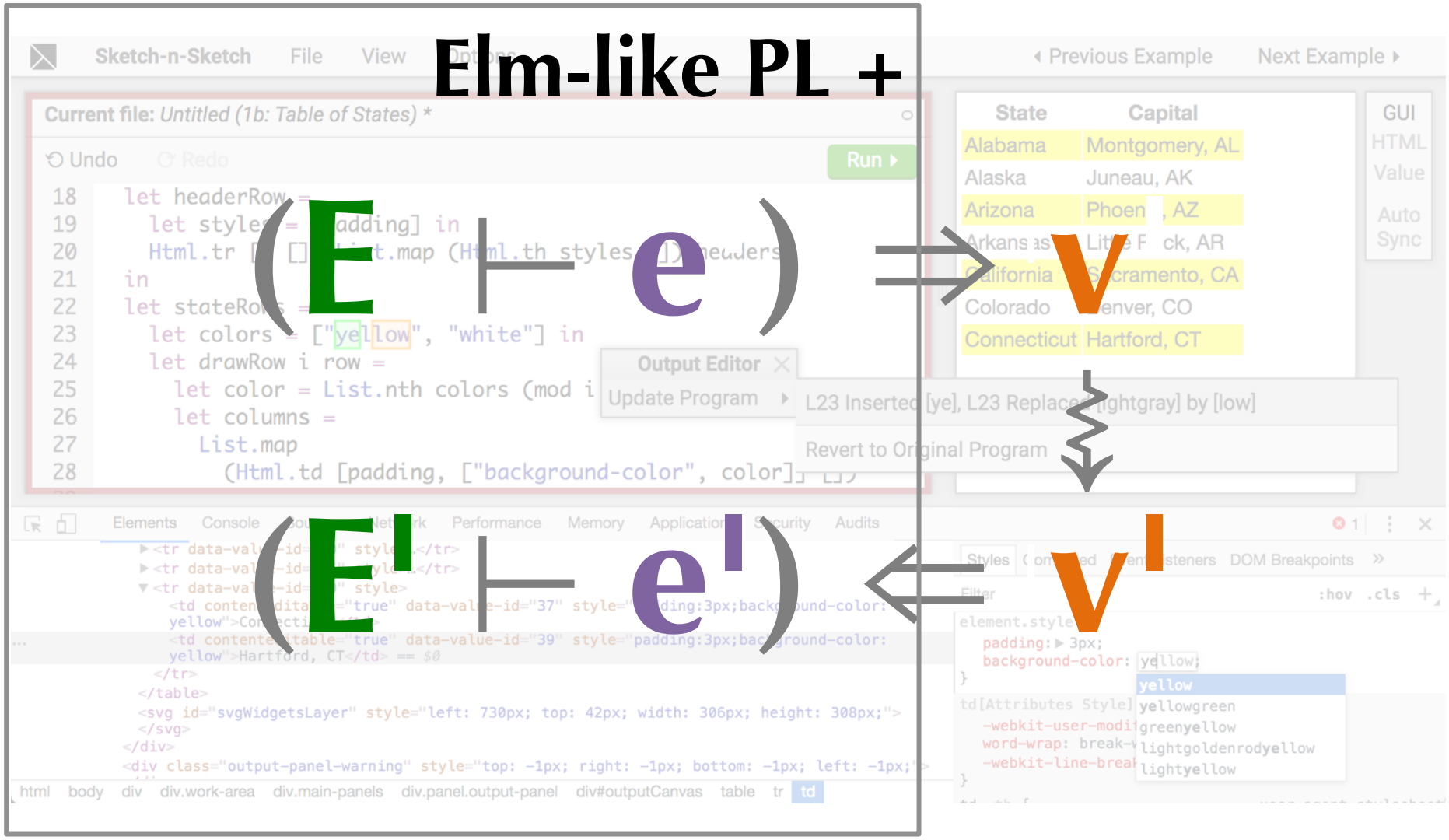


$$(E \vdash e) \Rightarrow v$$

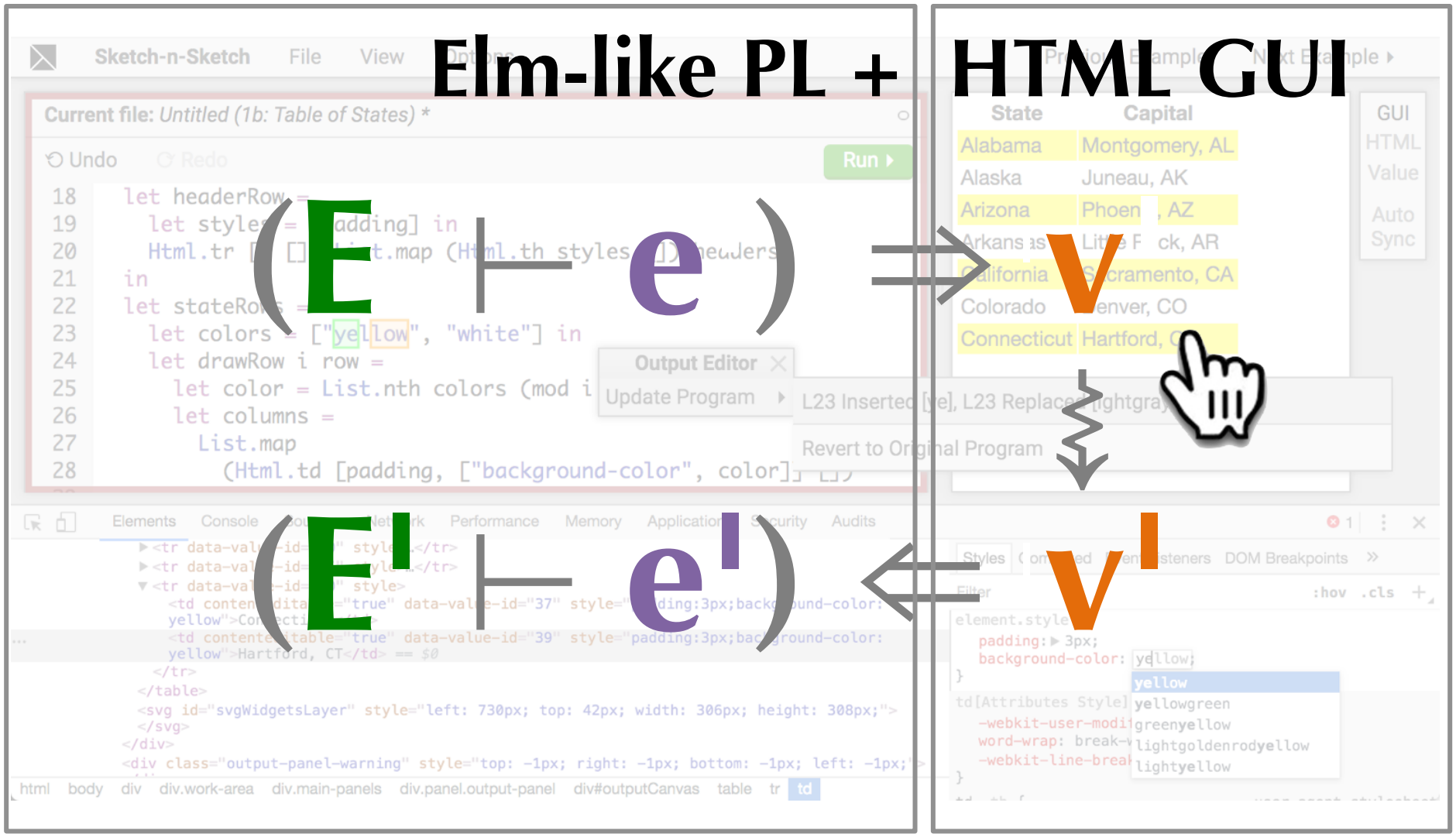
$$(E' \vdash e') \Leftarrow v'$$

Sketch-n-Sketch

Elm-like PL +



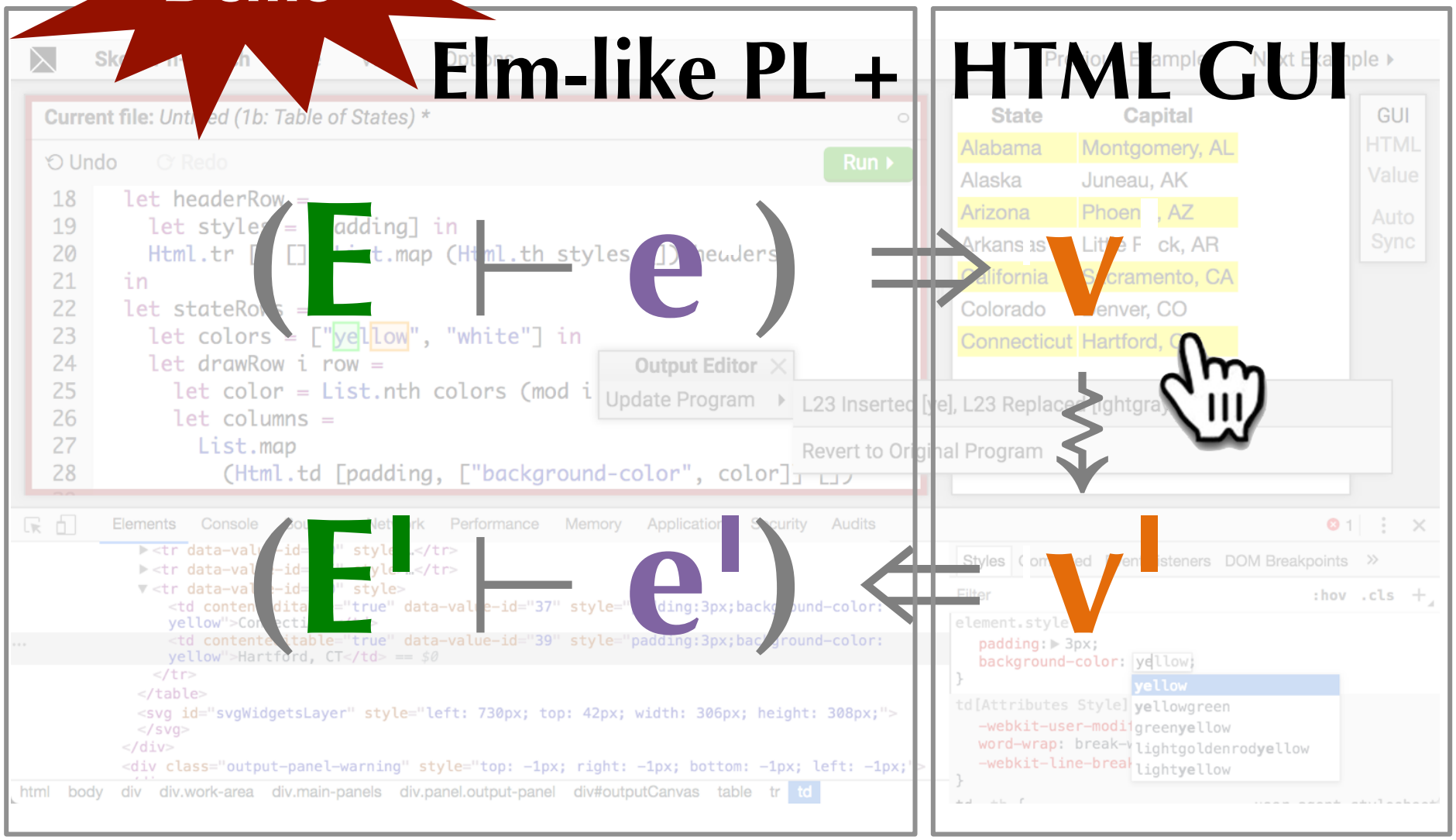
Sketch-n-Sketch



Sketch-n-Sketch

Demo

Elm-like PL + HTML GUI



Bidirectional Evaluation

Future Work

Repair Algorithm	Tracing plus bidirectional evaluation Version space algebra of diffs Synthesis of larger structural repairs
User Interface	Interaction to resolve ambiguities Tools for distributed web development Other source languages and target domains

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

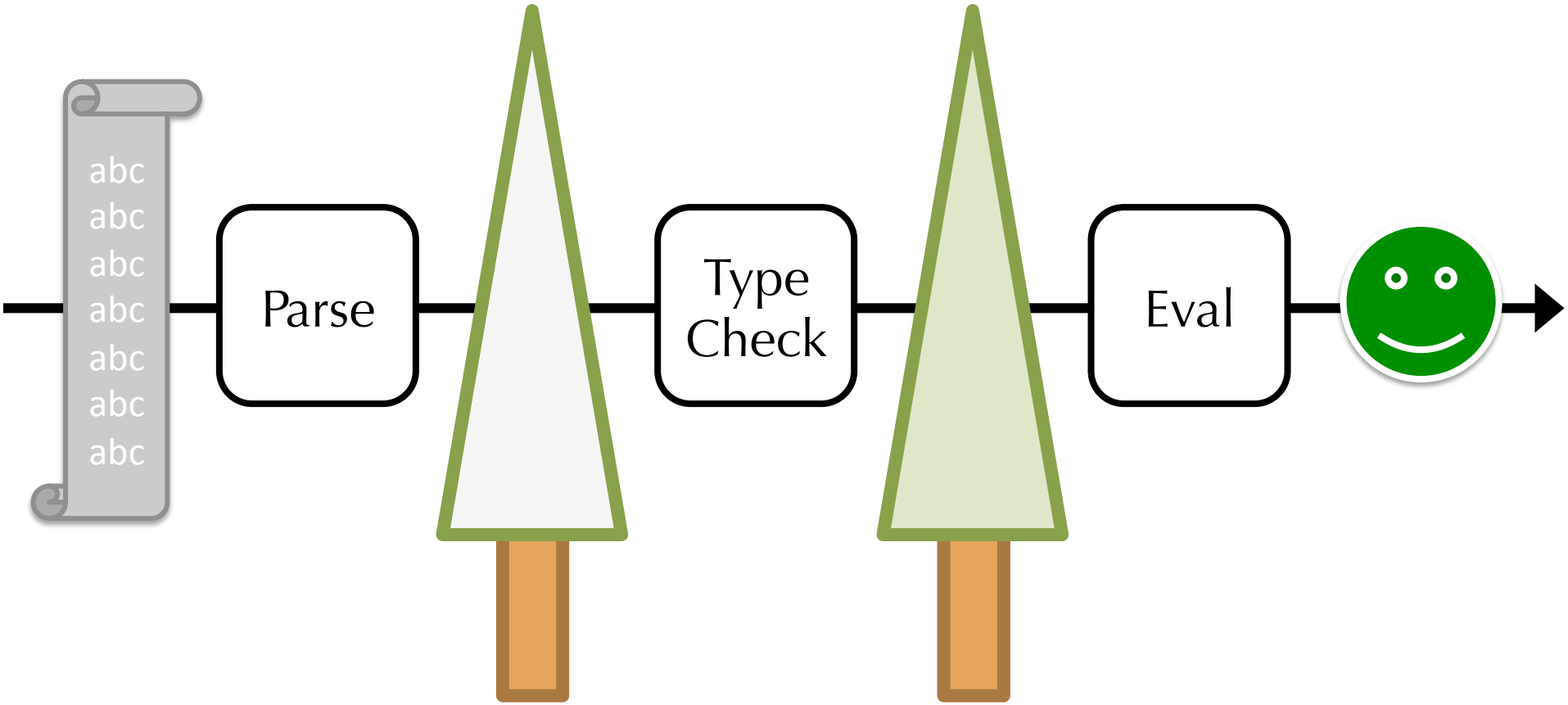
PLUIs

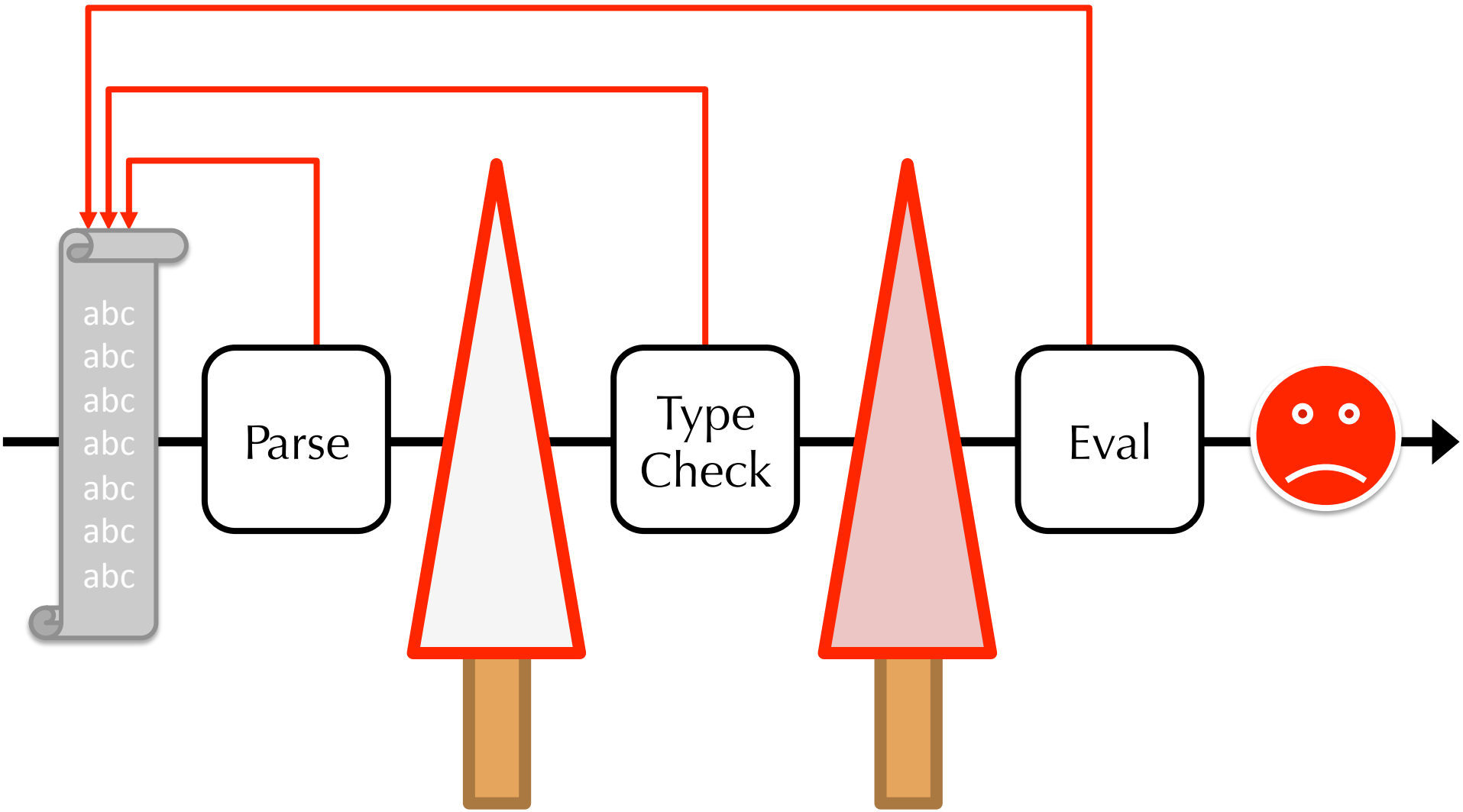
Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]


abc
abc
abc
abc
abc
abc
abc
Parse
Type
Check
Eval

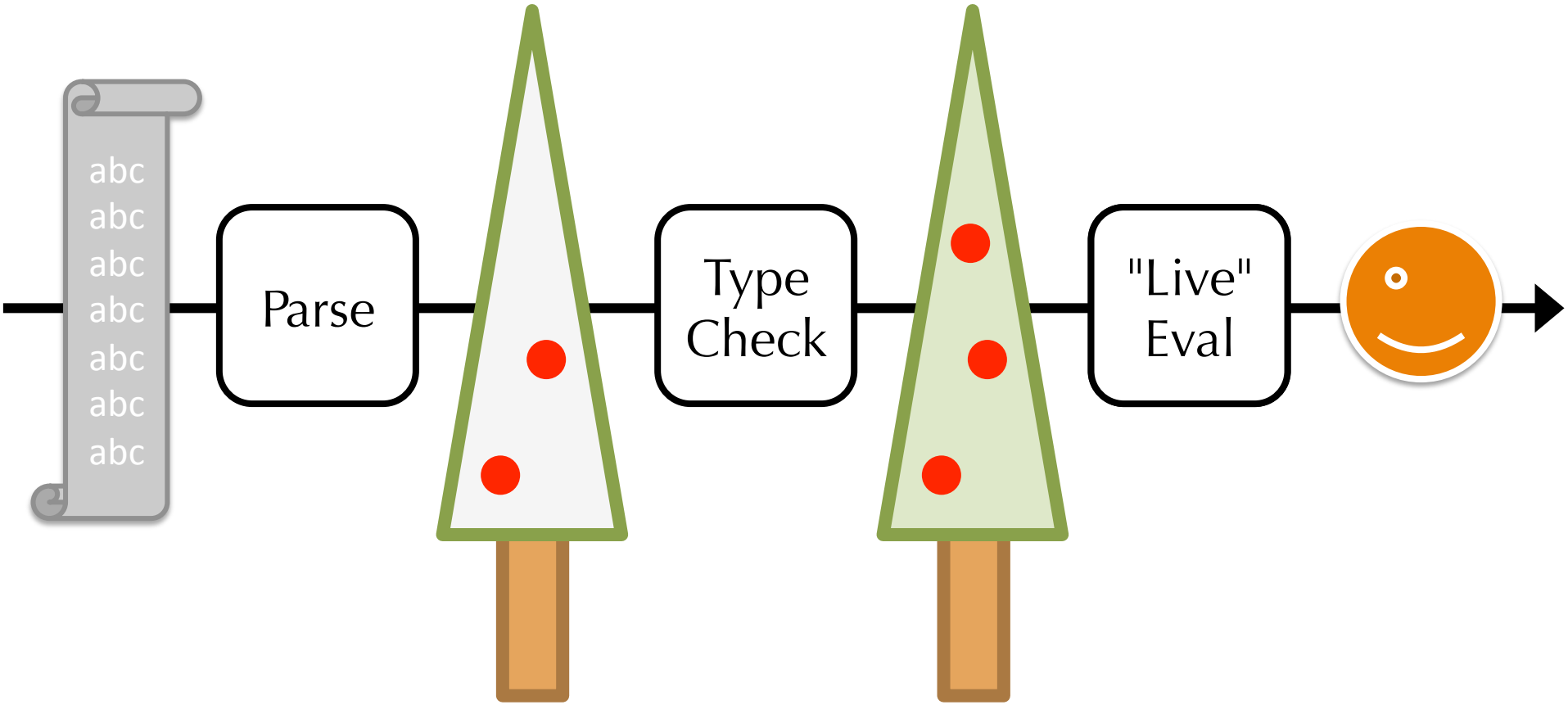

abc
abc
abc
abc
abc
abc
abc
Parse
Type
Check
Eval

Hazel



Semantic Foundations for Hole-Driven Development

[Omar et al. POPL 2017, POPL 2019, wip]

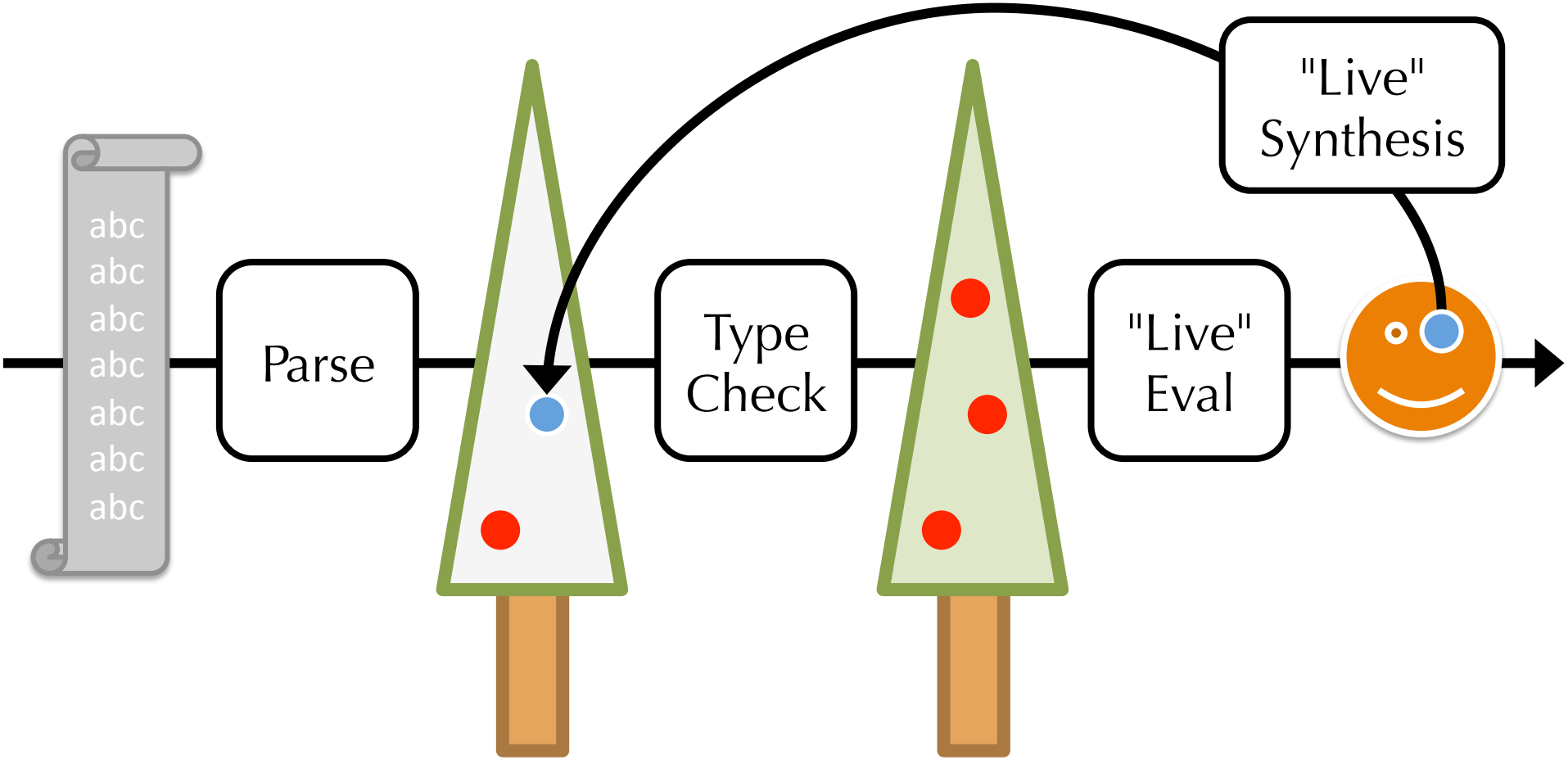


Hazel



Semantic Foundations for Hole-Driven Development

[Omar et al. POPL 2017, POPL 2019, wip]



PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

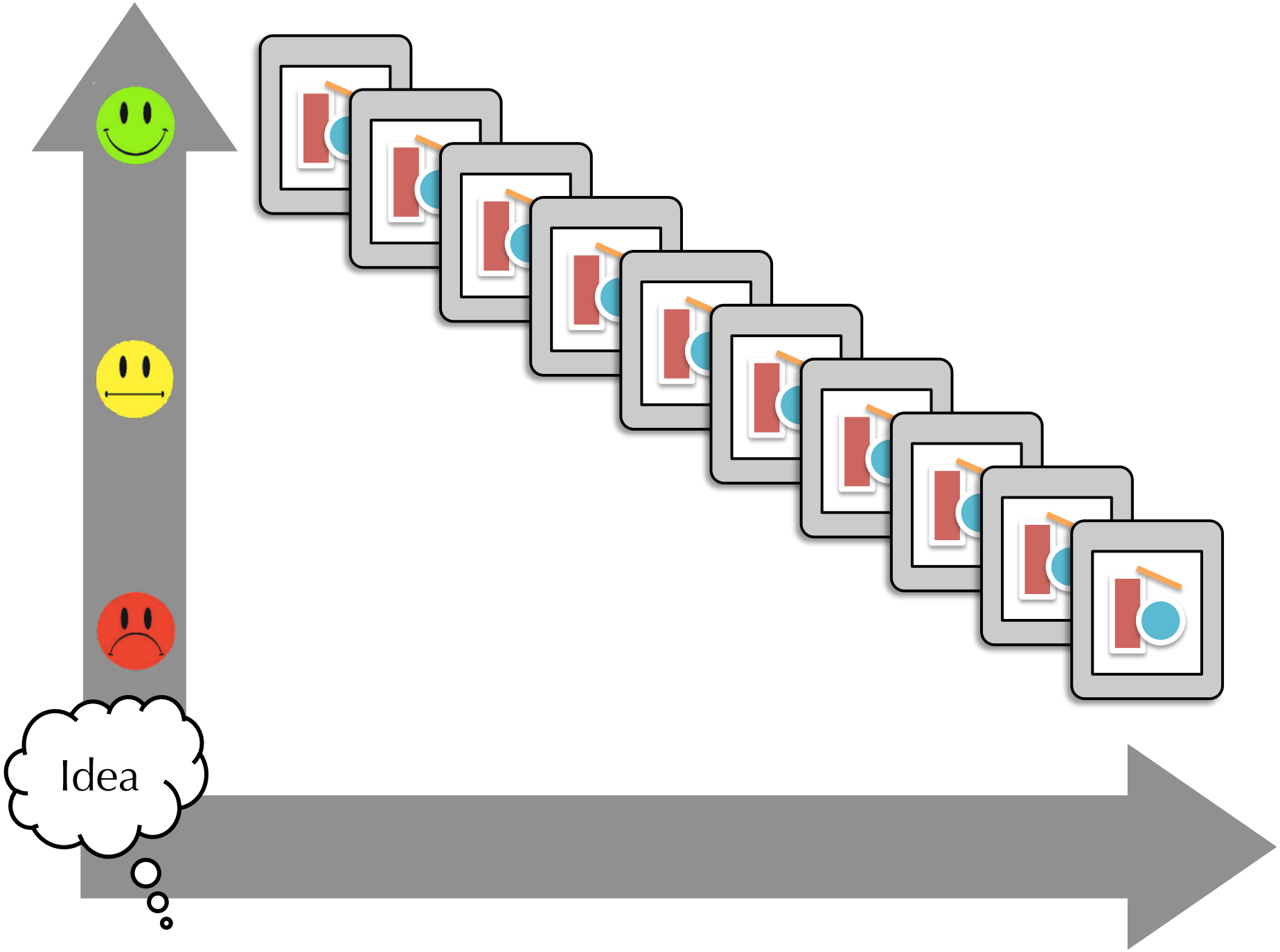
[POPL 2019, work-in-progress]

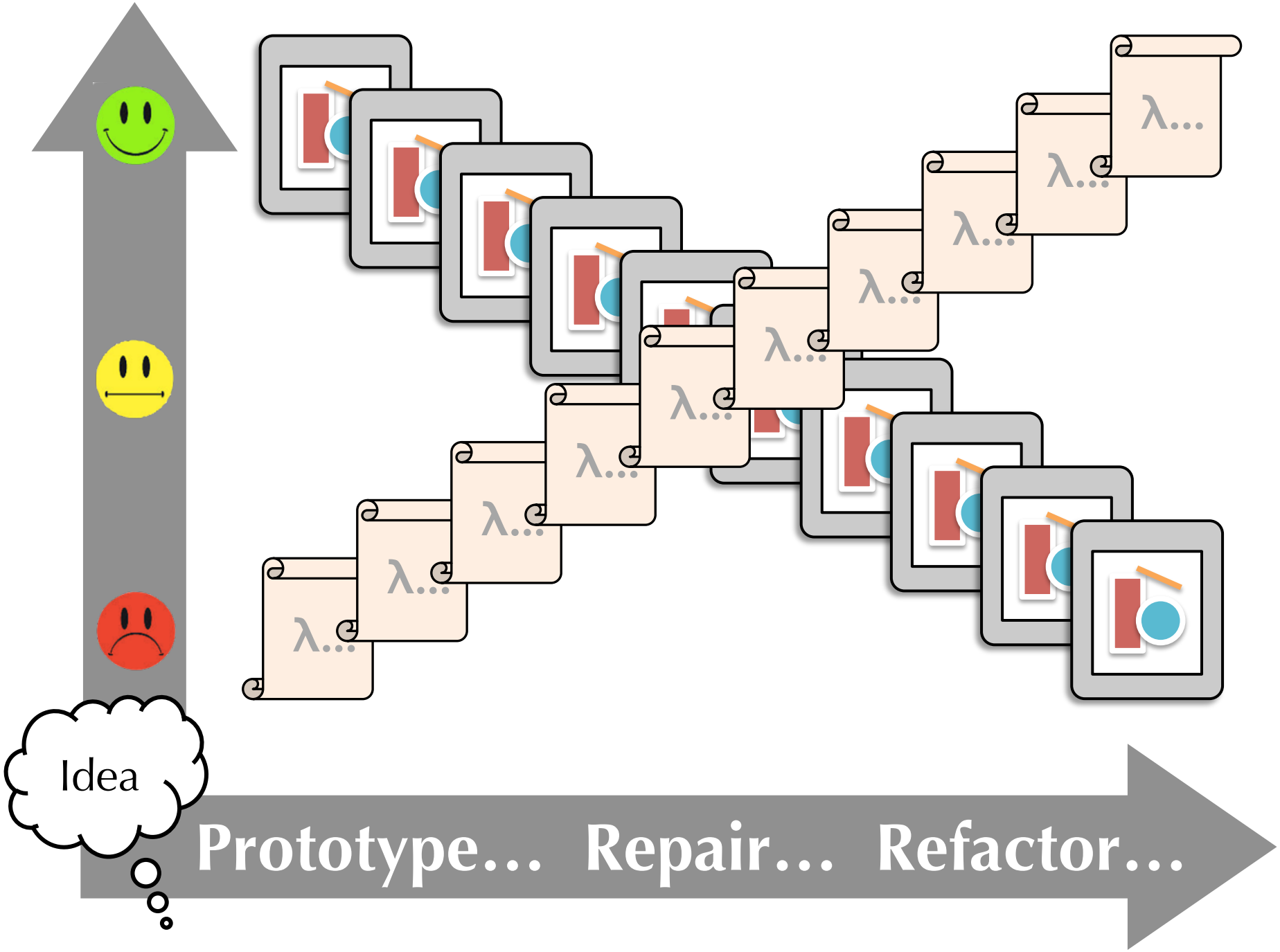
Lightweight Structured Text Editing

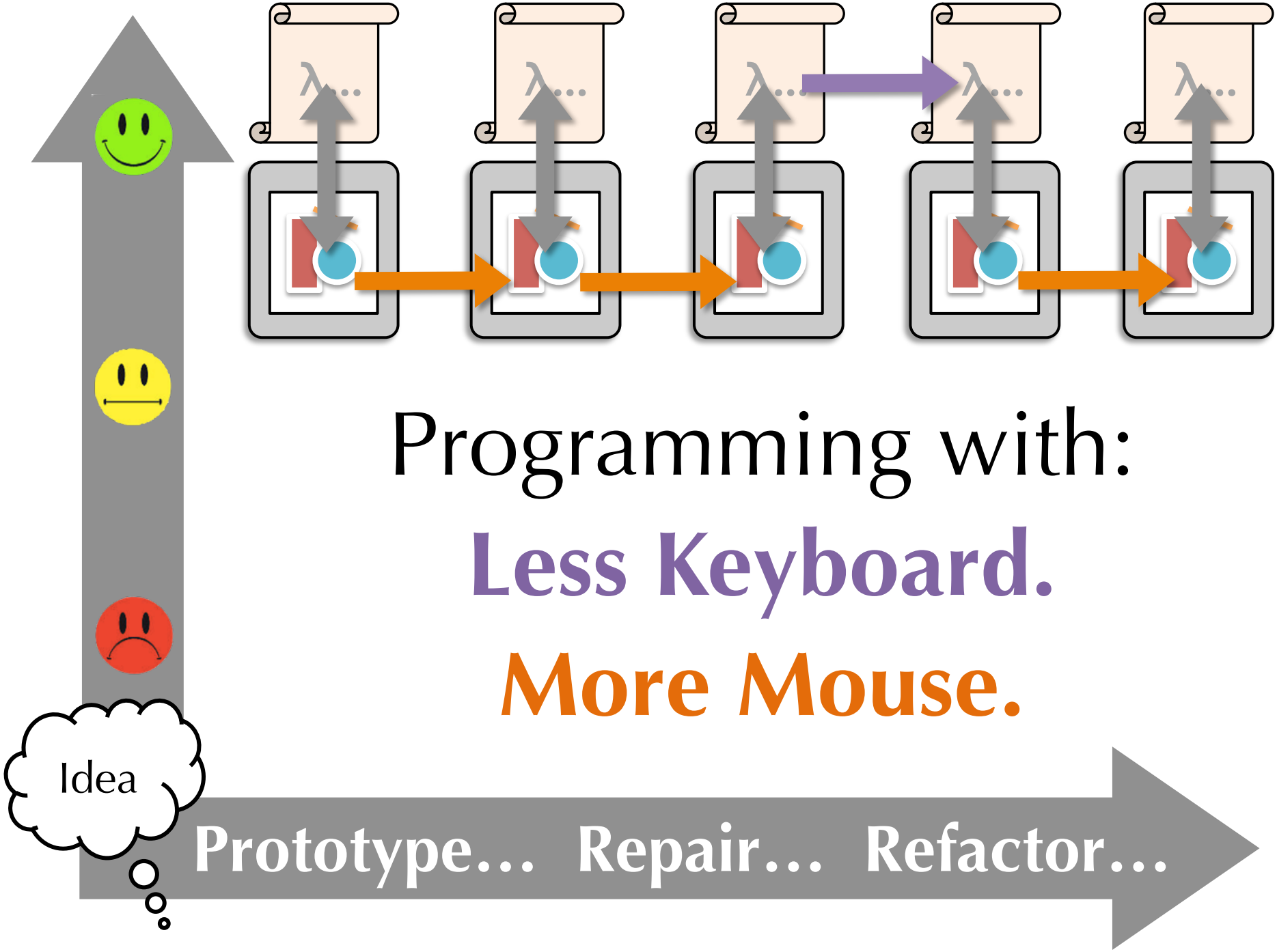
[ICSE 2018, work-in-progress]

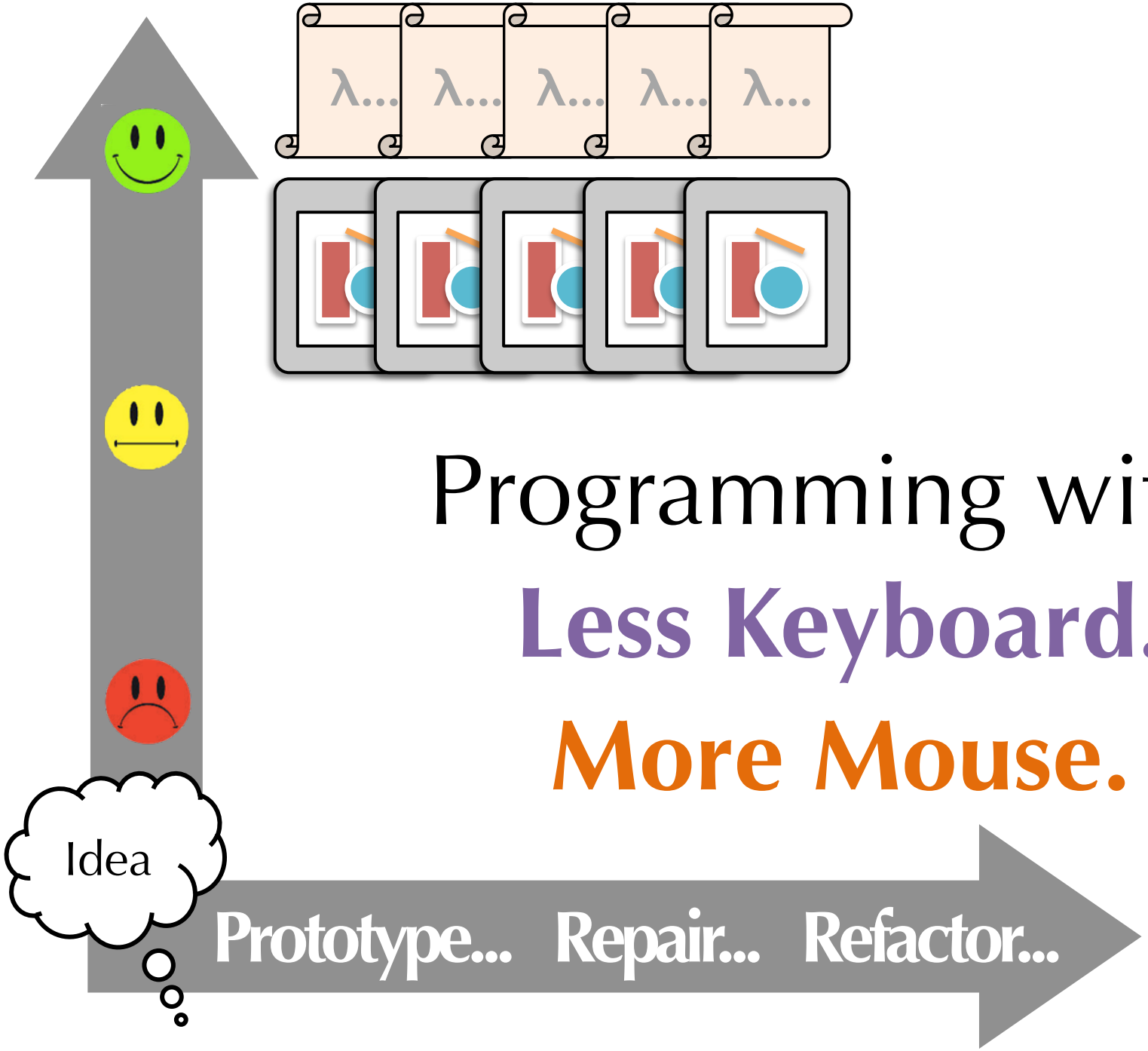
Output-Directed Program Synthesis

[UIST 2016, work-in-progress]


Idea


Idea
Prototype… Repair… Refactor…


Programming with:
Less Keyboard.
More Mouse.
Idea
Prototype… Repair… Refactor…

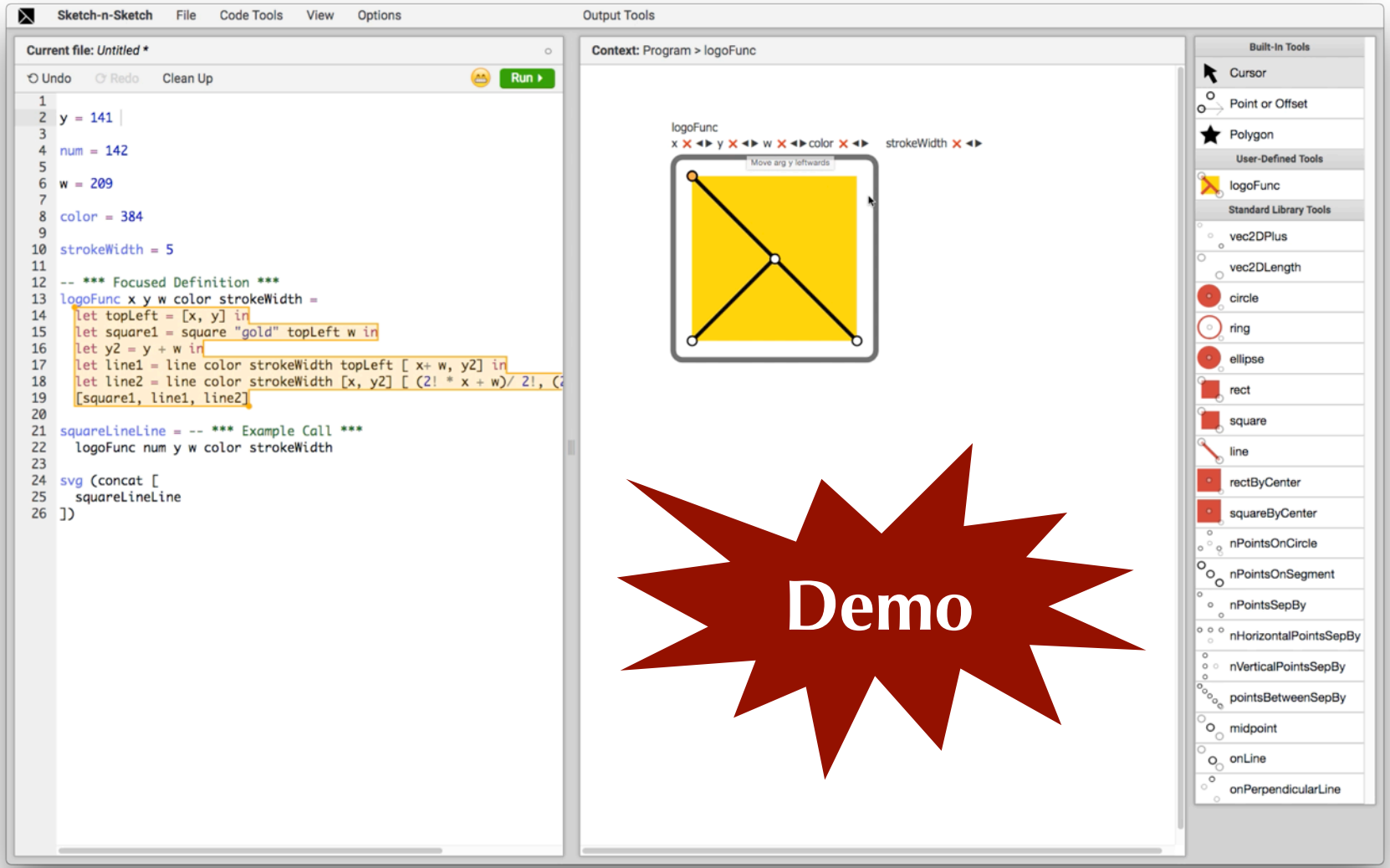


Programming with:

Less Keyboard.

More Mouse.

Output-Directed Programming



Output-Directed Programming

Future Work

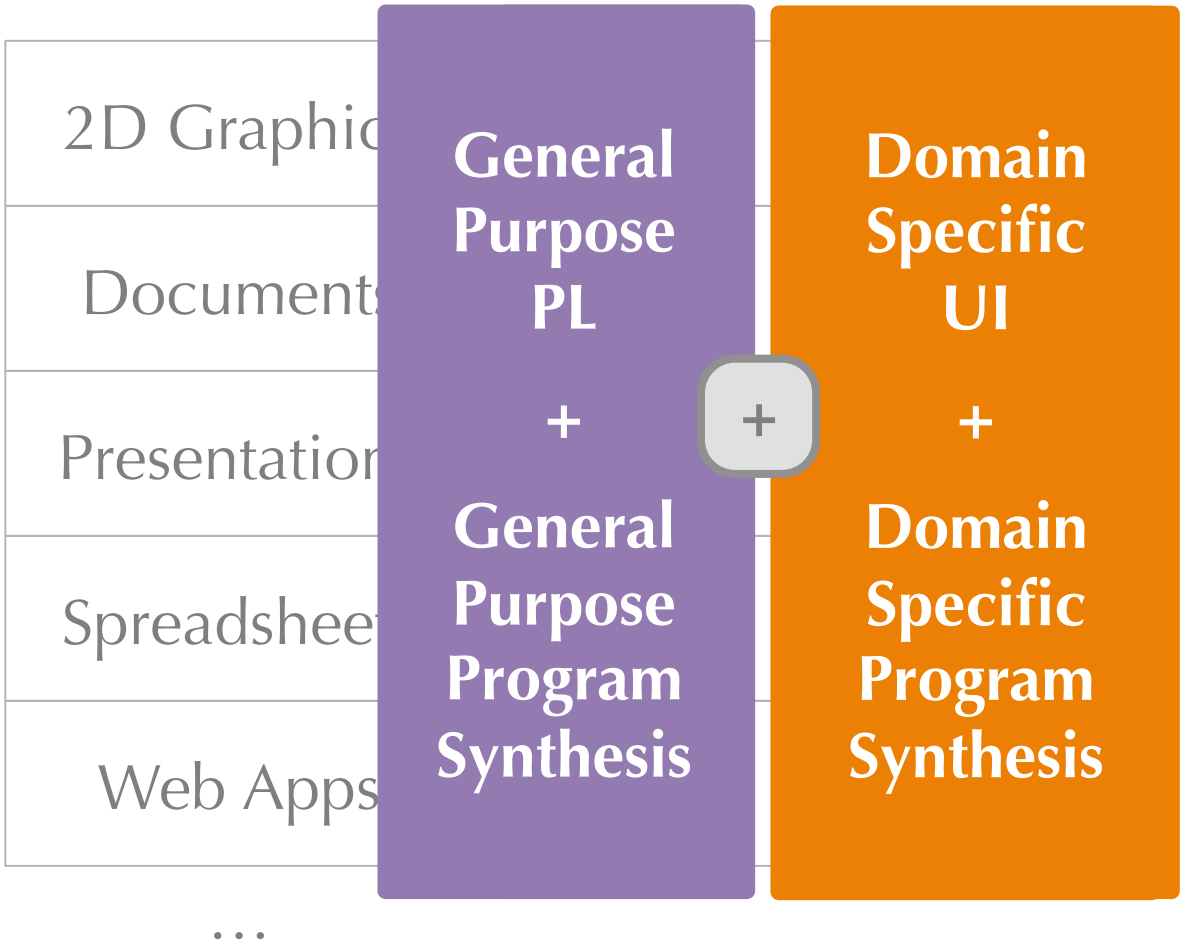
More control over generated code

More relationship / abstraction patterns

Visualization / manipulation of intermediates

DSLs for defining new transformations

PLs + GUIs


2D Graphic
Document
Presentation
Spreadshee
Web Apps
...
General Purpose PL
+
General Purpose Program Synthesis
+
Domain Specific UI
+
Domain Specific Program Synthesis

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

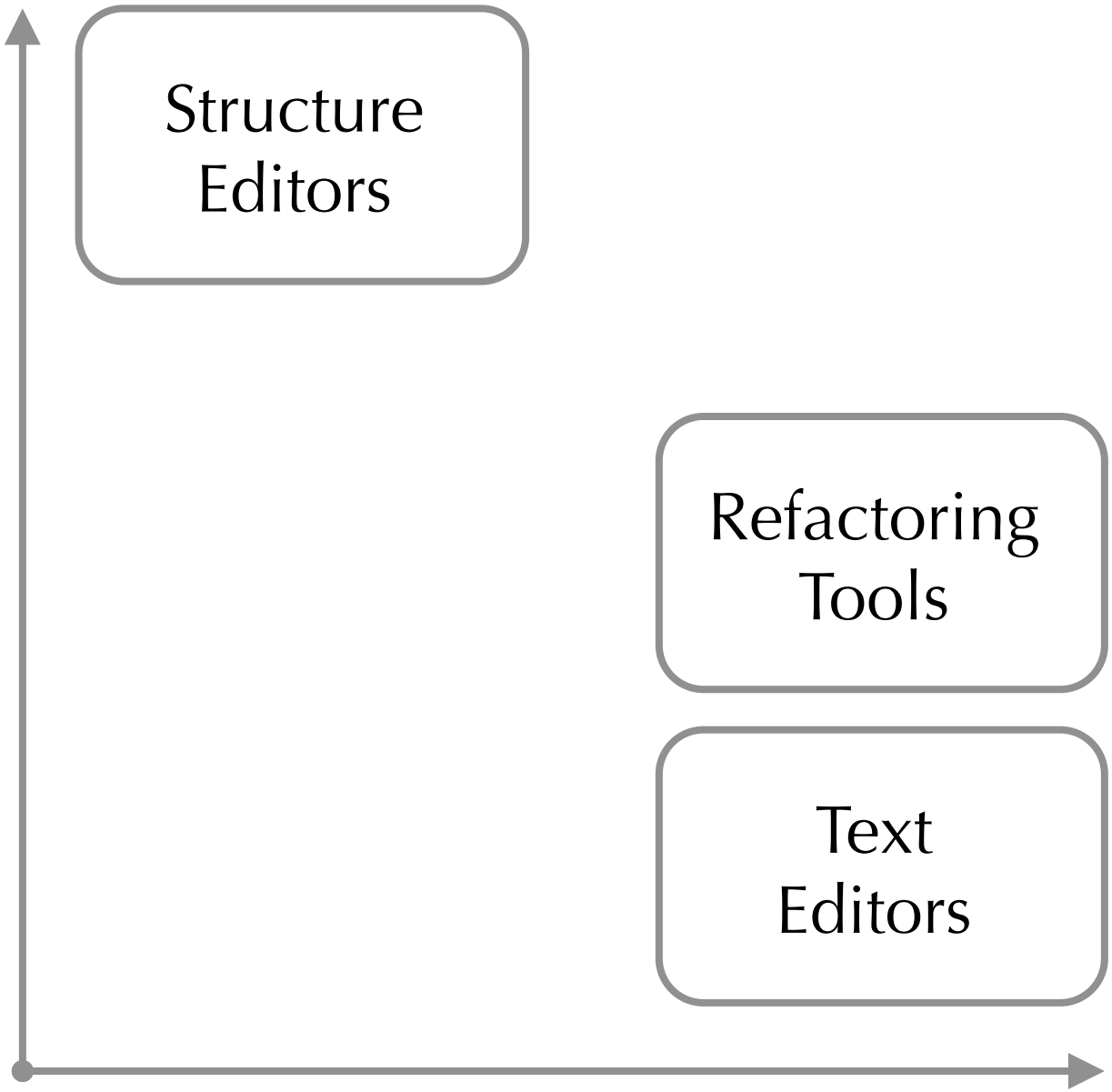
[POPL 2019, work-in-progress]

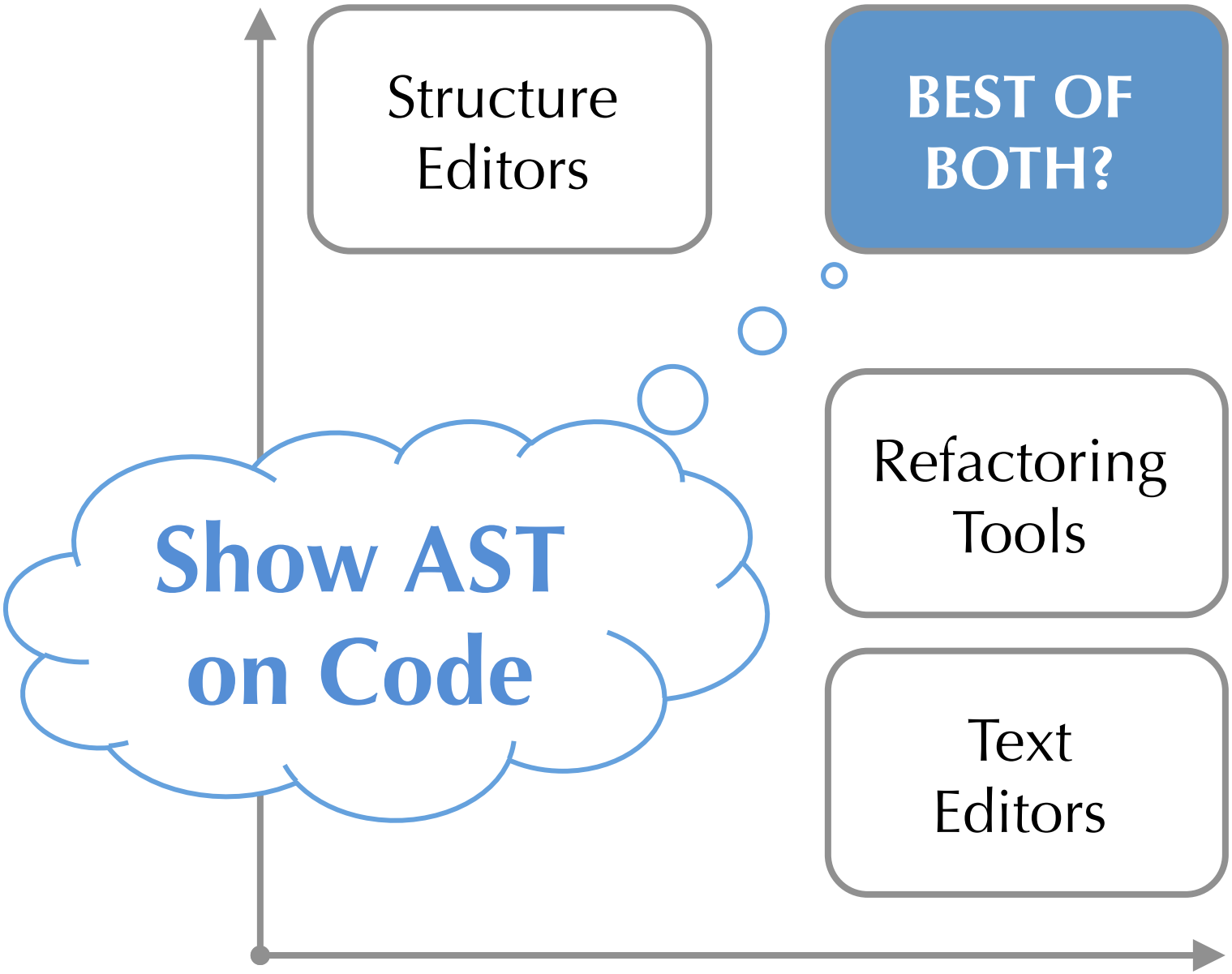
Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

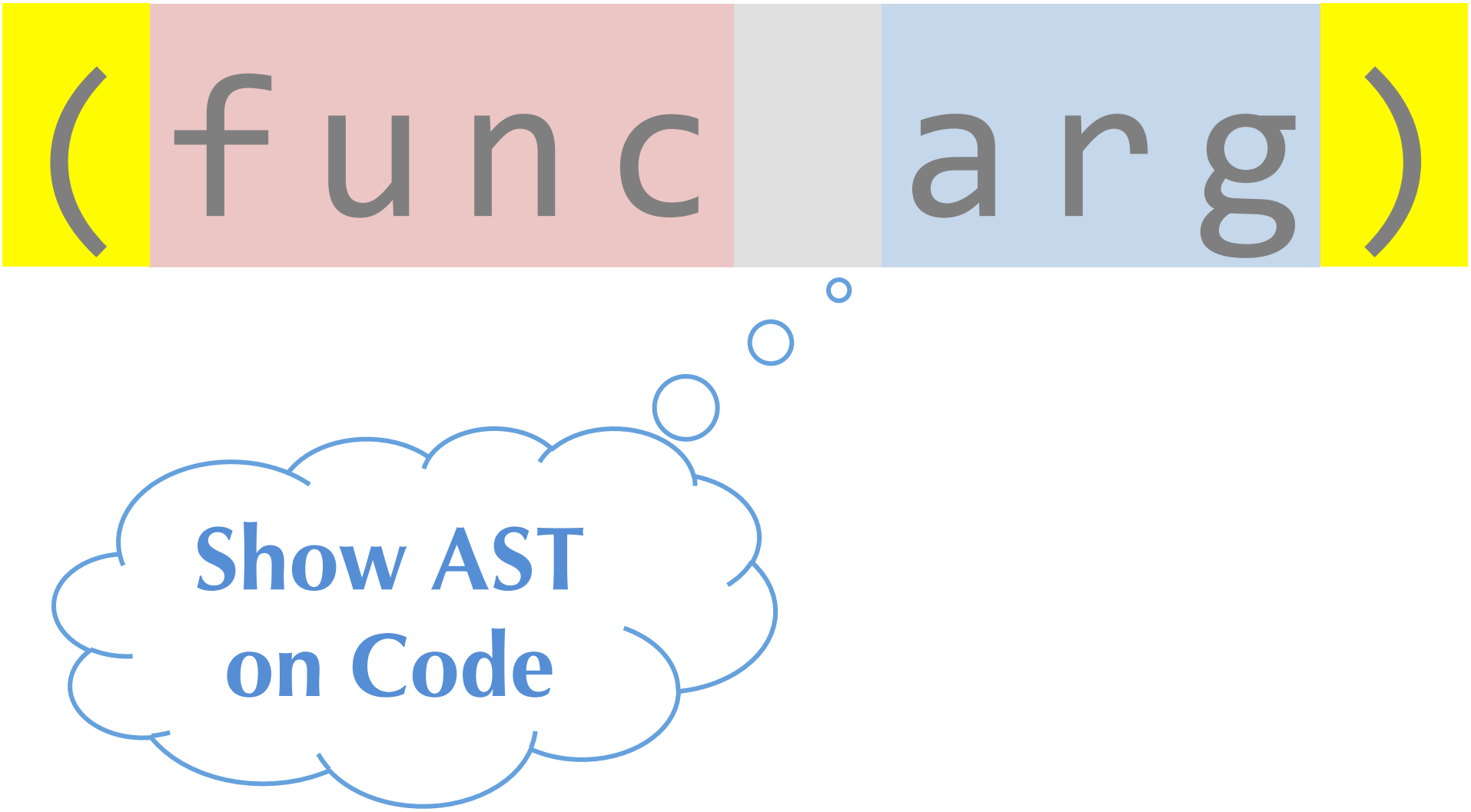

Structure Editors
Refactoring Tools
Text Editors


Structure Editors
BEST OF BOTH?
Show AST on Code
Refactoring Tools
Text Editors

```
(func arg)
```

(	f	u	n	c		a	r	g	)

Show AST on Code


(func arg)
Show AST
on Code

```
(func arg)
```

Show AST on Code

Structural (Multi-)Select

Context-Sensitive Transforms

Structured Text Editing

Sketch-n-Sketch File Code Tools View Options Previous Example Next Example

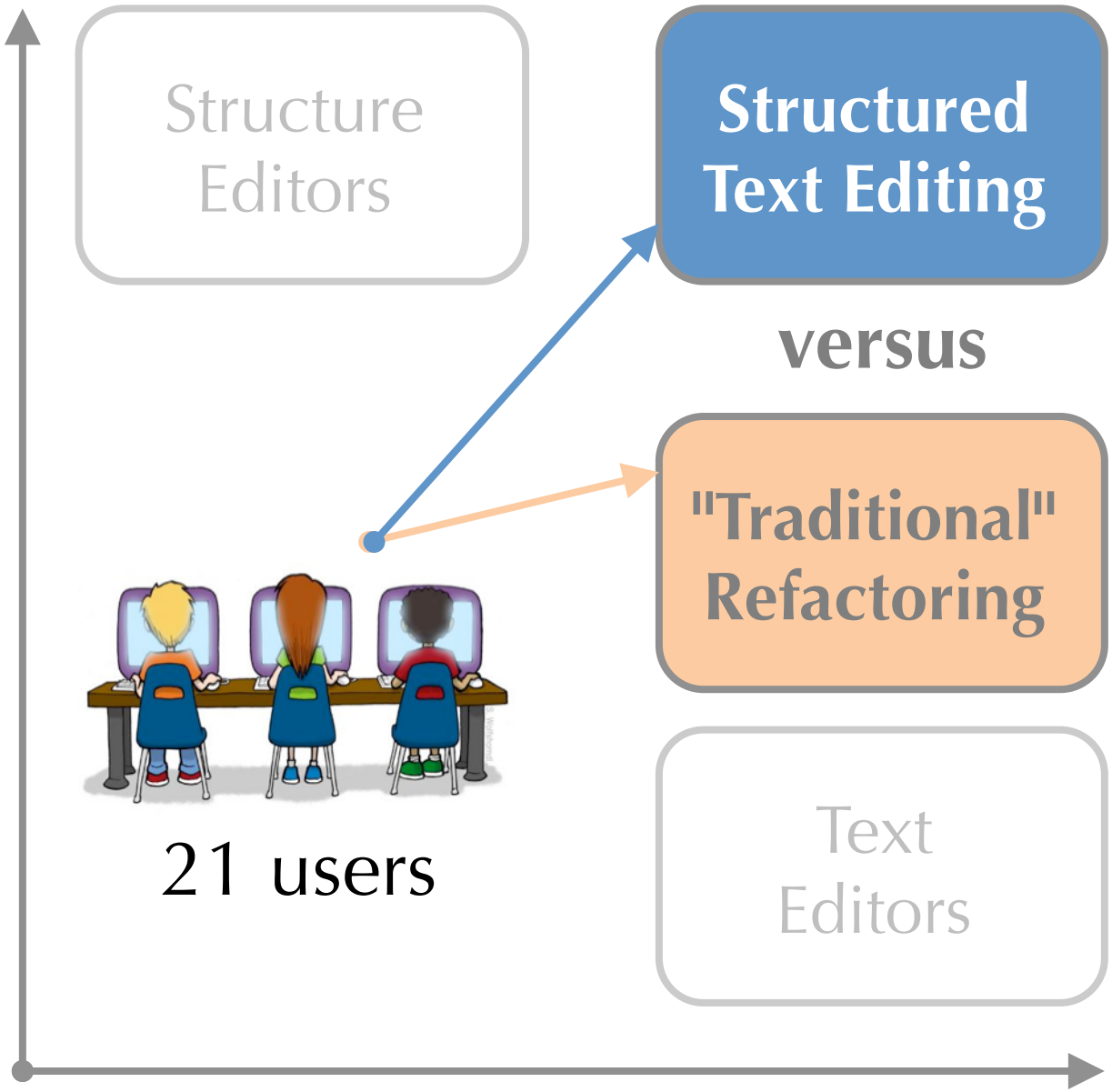
Current file: *Untitled (Get Started)* *

Undo Redo Run

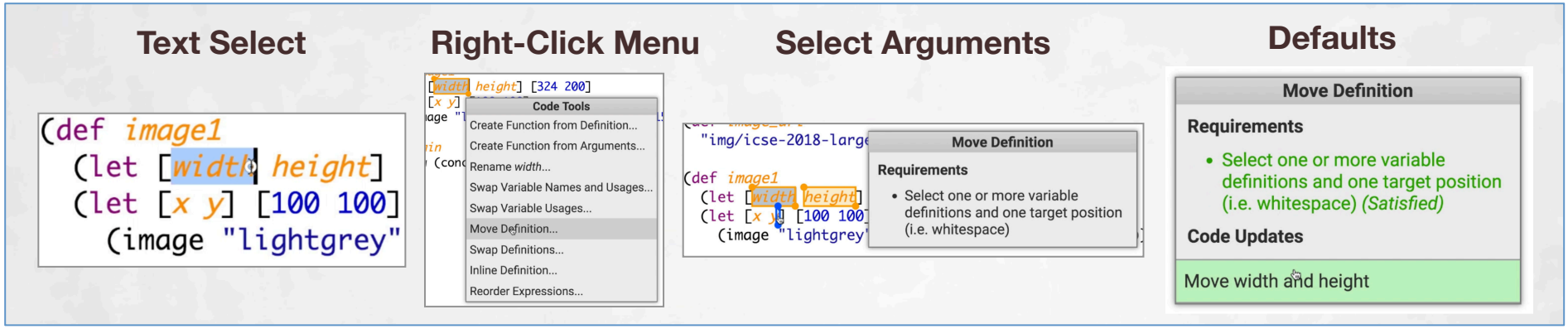
```
1
2  logo x y size
3    let
4      (cx, cy)
5
6      rect1 =
7        rect "slategray" x y size size
8
9      line1 =
10       line "white" 10 x y (x + size) (y + size)
11
12     line2 =
13       line "white" 10 x (y + size) cx cy
14   in
15     [rect1, line1, line2]
16
17 main =
18   svg (logo 25 33 217)
```

Code Tools
- Create Function from Arguments
- Add Argument
  - Insert Argument fill into logo
- Introduce Variable

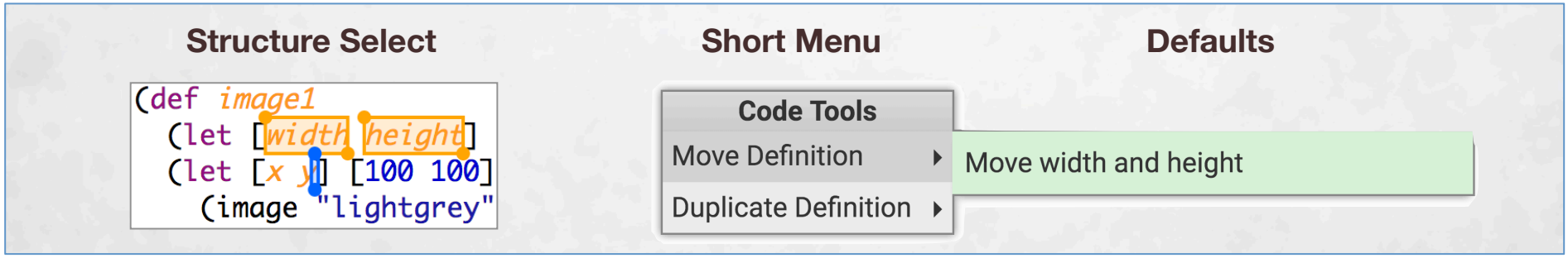
Demo


Structure Editors
Structured Text Editing
versus
"Traditional" Refactoring
21 users
Text Editors

Traditional ("Text-Select") Mode



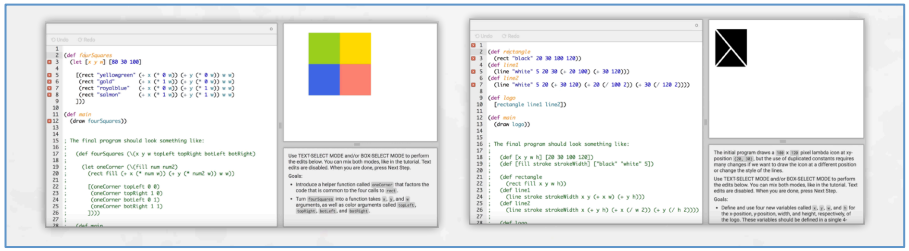
Deuce ("Box-Select") Mode



1. Tutorial

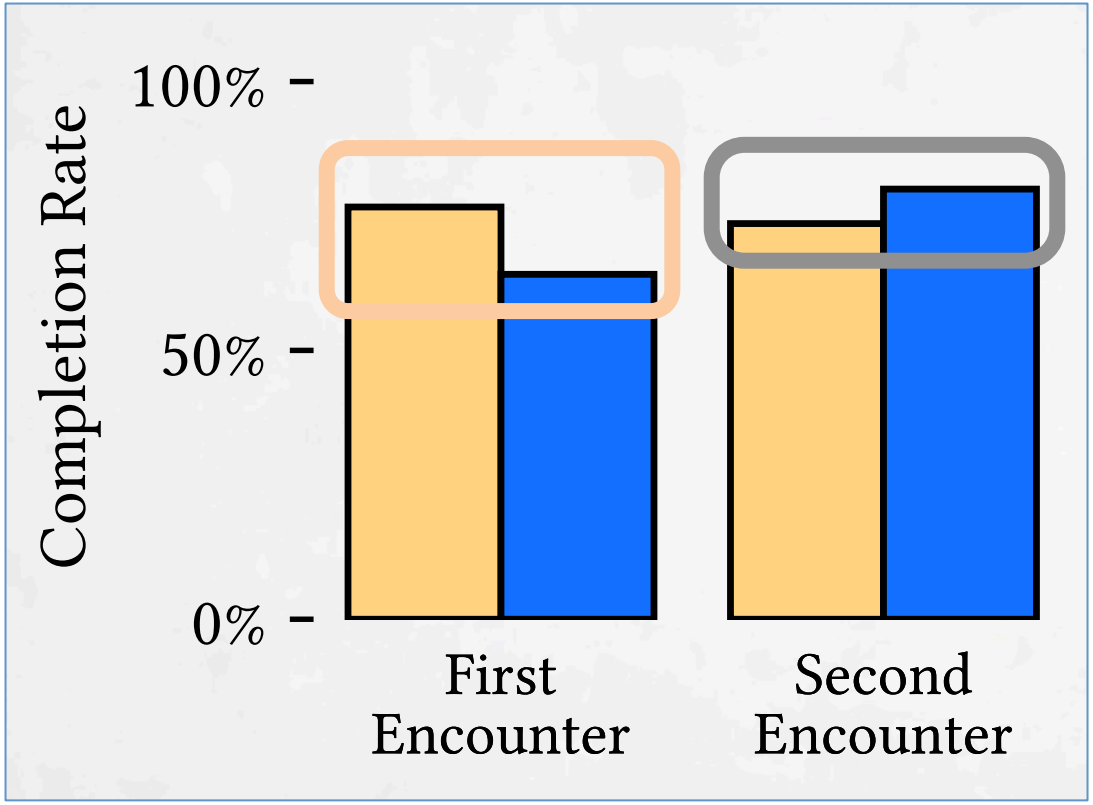
2. Head-to-Head Tasks (2x each; once per mode)

3. Mix-and-Match Tasks (free to use both modes)



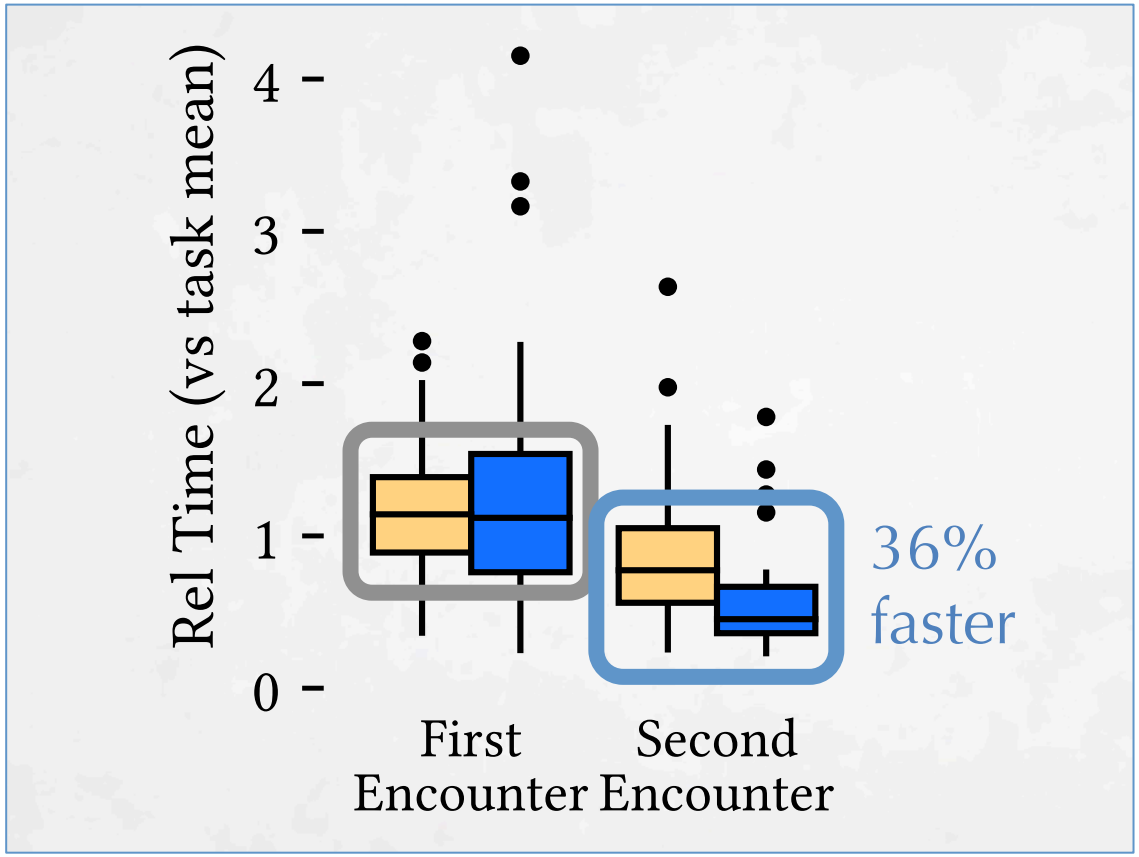
4. Exit Survey

Deuce more effective than Traditional?



Deuce doesn't help discoverability

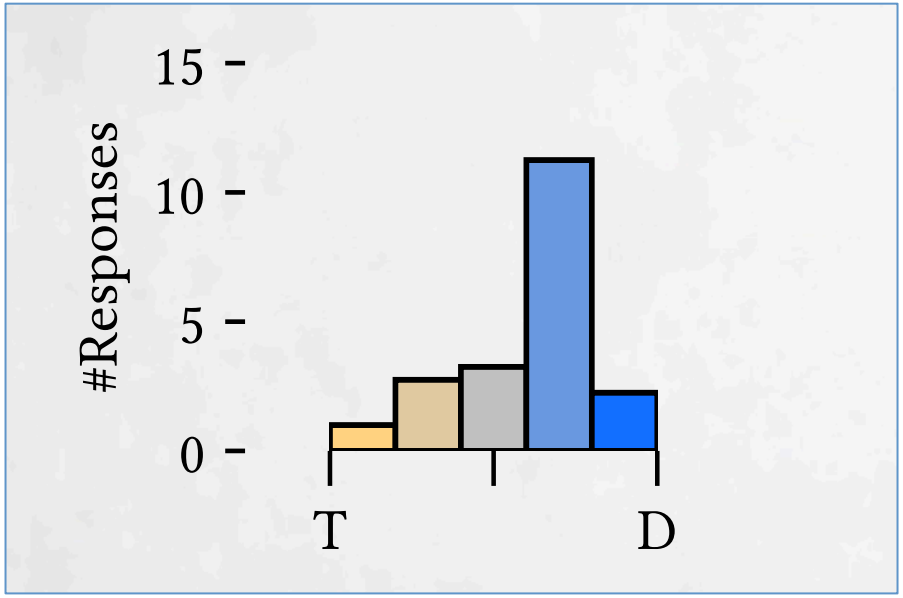
Deuce more effective than Traditional?



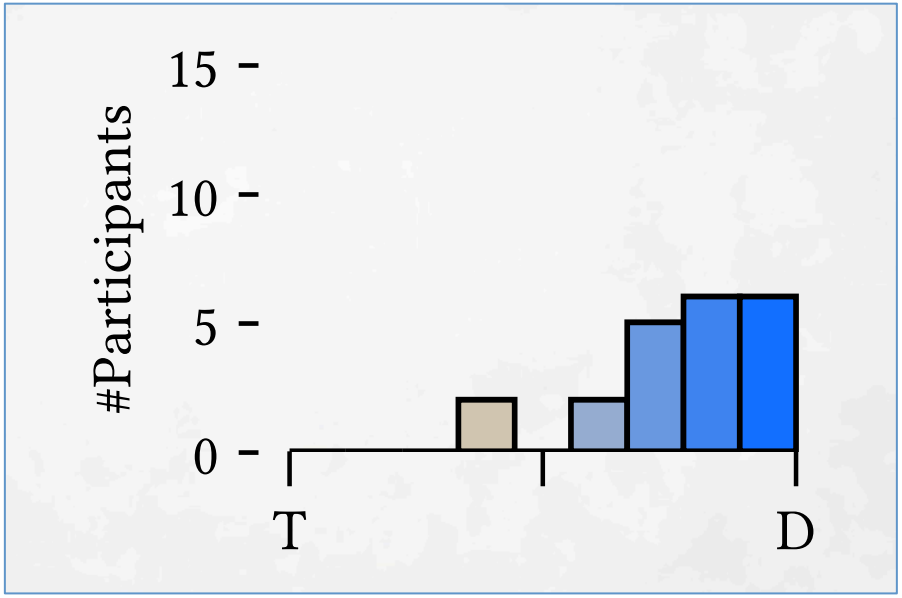
Deuce may be faster once learned

Deuce preferred to Traditional?

Survey



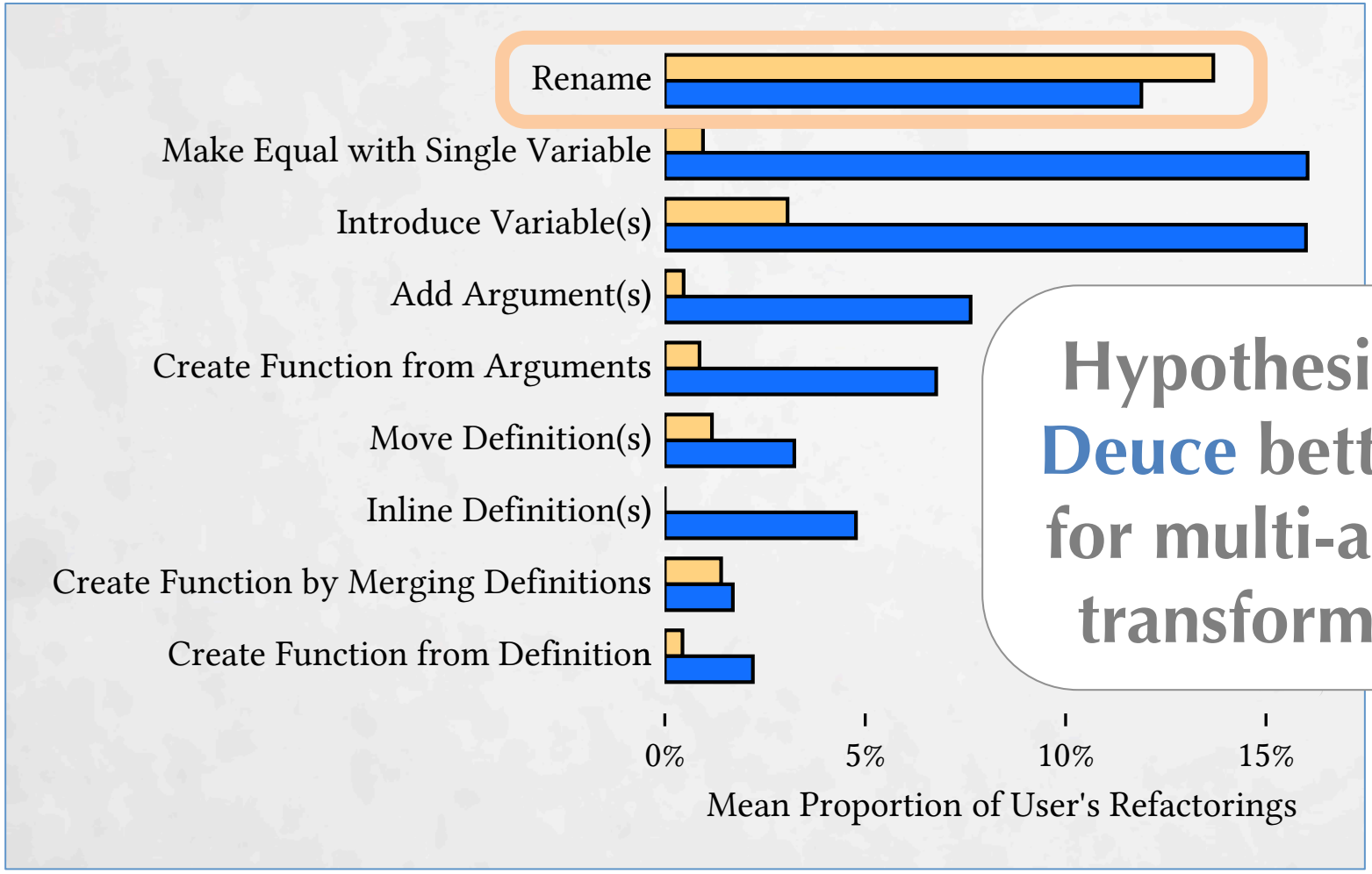
Observed



Modest subjective preference for Deuce

Almost everyone used Deuce more

Mix-and-Match Tool Usage



Hypothesis: Deuce better for multi-arg transforms

Deuce versus Traditional

Traditional may be better for discovery

Deuce may be faster once learned

Deuce strongly preferred

Structured Text Editing

Future Work

UI concerns for larger programs

How to encourage structured transforms?

DSLs for defining new transformations

DSLs for defining custom, type-specific UIs

Real languages in real editors

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

PLUIs

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

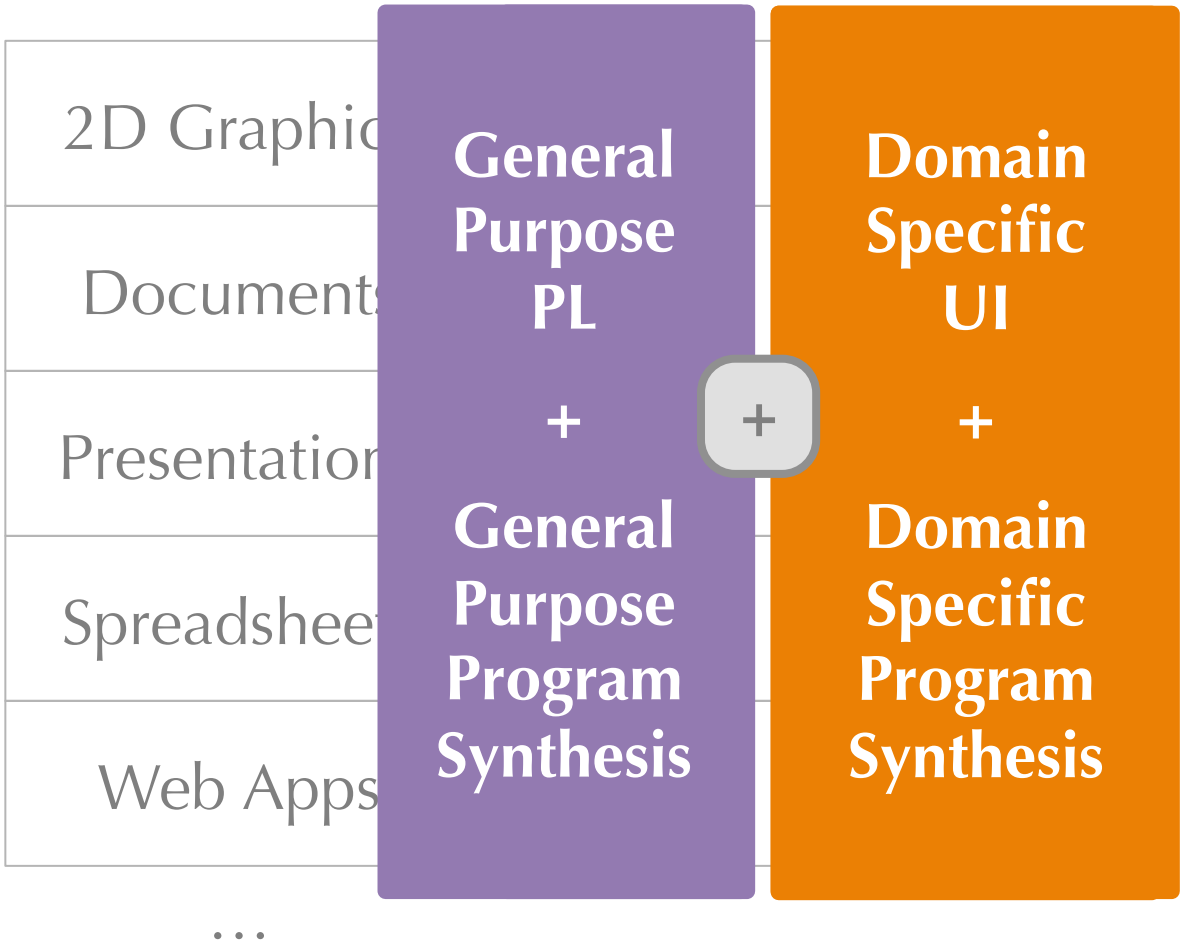
Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

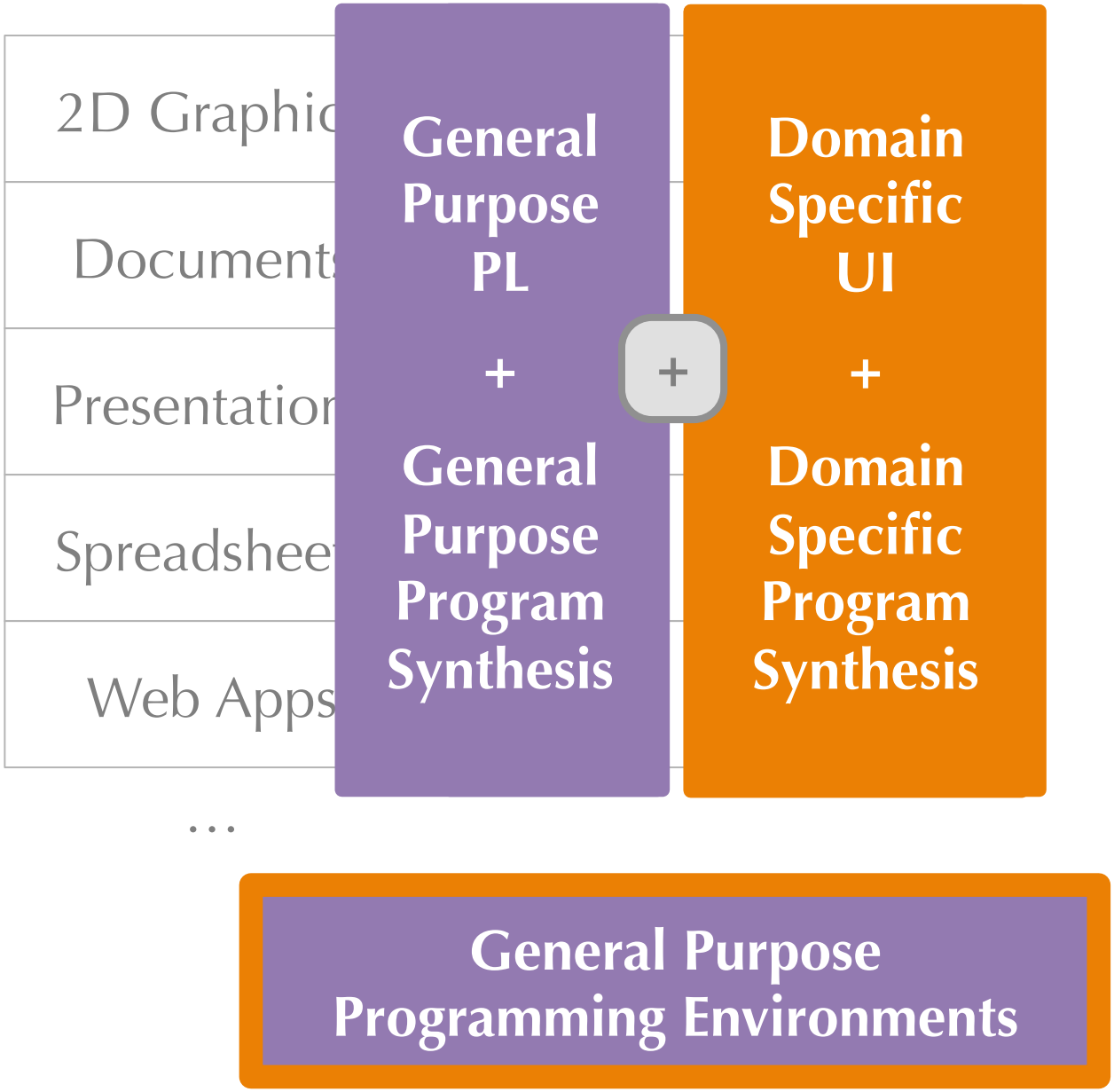
Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

PLs + GUIs


2D Graphic
Document
Presentation
Spreadshee
Web Apps
...
General Purpose PL
+
General Purpose Program Synthesis
+
Domain Specific UI
+
Domain Specific Program Synthesis

PLs + GUIs


2D Graphic
Document
Presentation
Spreadshee
Web Apps
...
General Purpose PL
+
General Purpose Program Synthesis
+
Domain Specific UI
+
Domain Specific Program Synthesis
General Purpose Programming Environments

Thanks!

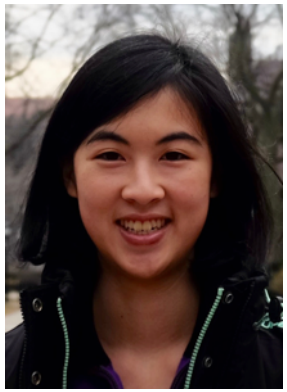
Nick Collins

Brian Hempel

Justin Lubin

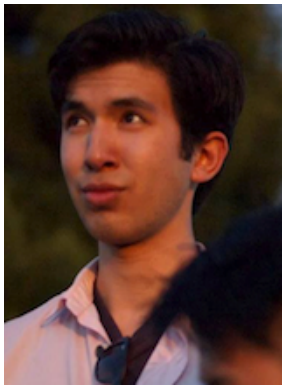
Cyrus Omar

Mikaël Mayer



Grace Lu

Mitch Spradlin



Jacob Albers

SWISS NATIONAL SCIENCE FOUNDATION

Bidirectional Evaluation for Program Repair

[PLDI 2016, OOPSLA 2018]

Live Evaluation with Program Synthesis

[POPL 2019, work-in-progress]

Sketch-n-Sketch

http://ravichugh.github.io/sketch-n-sketch/

Lightweight Structured Text Editing

[ICSE 2018, work-in-progress]

Output-Directed Program Synthesis

[UIST 2016, work-in-progress]

EXTRA SLIDES

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$

$$\rightsquigarrow$$

$$(E' \vdash e') \Leftarrow v'$$

e	$::=$	c	$\mid$	$\lambda x.e$	$\mid$	x	$\mid$	(e_1, e_2)	$\mid$	$e_1\ e_2$
v	$::=$	c	$\mid$	$[E]\ \lambda x.e$	$\mid$	(v_1, v_2)				
E	$::=$	$-$	$\mid$	$(E, x \mapsto v)$						

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rotatebox{90}{$\rightsquigarrow$} \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$
\begin{array}{lcl}
e & ::= & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
v & ::= & c \mid [E]\ \lambda x.e \mid (v_1, v_2) \\
E & ::= & - \mid (E, x \mapsto v)
\end{array}
$$

$$
\begin{array}{c}
(E \vdash e) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

$$
\begin{array}{c}
(E \vdash c) \Rightarrow c \\
\rightsquigarrow \\
c'
\end{array}
$$

$$
\begin{array}{lcl}
e & ::= & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
v & ::= & c \mid [E]\ \lambda x.e \mid (v_1, v_2) \\
E & ::= & - \mid (E, x \mapsto v)
\end{array}
$$

$$
\begin{array}{c}
(E \vdash e) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

$$
\begin{array}{c}
(E \vdash c) \Rightarrow c \\
\rightsquigarrow \\
(E \vdash c') \Leftarrow c'
\end{array}
$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$(E \vdash \lambda x.e) \Rightarrow [E]\ \lambda x.e \rightsquigarrow [E']\ \lambda x.e'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$
$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$
$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash \lambda x.e) \Rightarrow [E]\ \lambda x.e$$
$$\rightsquigarrow$$
$$(E' \vdash \lambda x.e') \Leftarrow [E']\ \lambda x.e'$$

$$e ::= c \mid \lambda x.e \mid \boxed{x} \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E, x \mapsto v) \vdash x) \Rightarrow v \rightsquigarrow v'$$

$$e ::= c \mid \lambda x.e \mid \boxed{x} \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$

$$\rightsquigarrow$$

$$(E' \vdash e') \Leftarrow v'$$

$$((E, x \mapsto v) \vdash x) \Rightarrow v$$

$$\rightsquigarrow$$

$$((E, x \mapsto v') \vdash x) \Leftarrow v'$$

$$e ::= c \mid \lambda x.e \mid x \mid \boxed{(e_1, e_2)} \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash (e_1, e_2)) \Rightarrow (v_1, v_2)$$
$$\rightsquigarrow$$
$$(v_1', v_2')$$

$$e ::= c \mid \lambda x.e \mid x \mid \boxed{(e_1, e_2)} \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\begin{array}{c} (E \vdash e_1) \Rightarrow v_1 \\ \rightsquigarrow \\ (E_1' \vdash e_1') \Leftarrow v_1' \end{array}$$

$$\begin{array}{c} (E \vdash (e_1, e_2)) \Rightarrow (v_1, v_2) \\ \rightsquigarrow \\ (v_1', v_2') \end{array}$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\frac{\begin{array}{c} (E \vdash e_1) \Rightarrow v_1 \\ \rightsquigarrow \\ (E_1' \vdash e_1') \Leftarrow v_1' \end{array} \qquad \begin{array}{c} (E \vdash e_2) \Rightarrow v_2 \\ \rightsquigarrow \\ (E_2' \vdash e_2') \Leftarrow v_2' \end{array}}{\begin{array}{c} (E \vdash (e_1, e_2)) \Rightarrow (v_1, v_2) \\ \rightsquigarrow \\ (v_1', v_2') \end{array}}$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\begin{array}{c} (E \vdash e_1) \Rightarrow v_1 \\ \rightsquigarrow \\ (E_1' \vdash e_1') \Leftarrow v_1' \end{array} \qquad \begin{array}{c} (E \vdash e_2) \Rightarrow v_2 \\ \rightsquigarrow \\ (E_2' \vdash e_2') \Leftarrow v_2' \end{array}$$

$$\begin{array}{c} (E \vdash (e_1, e_2)) \Rightarrow (v_1, v_2) \\ \rightsquigarrow \\ (E_1' \oplus E_2' \vdash (e_1', e_2')) \Leftarrow (v_1', v_2') \end{array}$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\begin{array}{c} (E \vdash e_1) \Rightarrow v_1 \\ \rightsquigarrow \\ (E_1' \vdash e_1') \Leftarrow v_1' \end{array} \qquad \begin{array}{c} (E \vdash e_2) \Rightarrow v_2 \\ \rightsquigarrow \\ (E_2' \vdash e_2') \Leftarrow v_2' \end{array}$$

$$\begin{array}{c} (E \vdash (e_1, e_2)) \Rightarrow (v_1, v_2) \\ \rightsquigarrow \\ (E_1' \oplus E_2' \vdash (e_1', e_2')) \Leftarrow (v_1', v_2') \end{array}$$

$$e ::= c \mid \lambda x.e \mid x \mid \boxed{(e_1, e_2)} \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((x \mapsto 1) \vdash (x, x)) \Rightarrow (1, 1)$$
$$\rightsquigarrow$$
$$(2, 3)$$

$$e ::= c \mid \lambda x.e \mid x \mid \boxed{(e_1, e_2)} \mid e_1\ e_2$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\begin{array}{c} ((x \mapsto 1) \vdash x) \Rightarrow 1 \\ \rightsquigarrow \\ ((x \mapsto 2) \vdash x) \Leftarrow 2 \end{array} \qquad \begin{array}{c} ((x \mapsto 1) \vdash x) \Rightarrow 1 \\ \rightsquigarrow \\ ((x \mapsto 3) \vdash x) \Leftarrow 3 \end{array}$$

$$\begin{array}{c} ((x \mapsto 1) \vdash (x, x)) \Rightarrow (1, 1) \\ \rightsquigarrow \\ (2, 3) \end{array}$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$
$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$
$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\begin{array}{c} ((x \mapsto 1) \vdash x) \Rightarrow 1 \\ \rightsquigarrow \\ ((x \mapsto 2) \vdash x) \Leftarrow 2 \end{array} \qquad \begin{array}{c} ((x \mapsto 1) \vdash x) \Rightarrow 1 \\ \rightsquigarrow \\ ((x \mapsto 3) \vdash x) \Leftarrow 3 \end{array}$$

$$\begin{array}{c} ((x \mapsto 1) \vdash (x, x)) \Rightarrow (1, 1) \\ \rightsquigarrow \\ ((x \mapsto 2) \oplus (x \mapsto 3) \vdash (x, x)) \Leftarrow (2, 3) \end{array}$$

$$
\begin{aligned}
e &::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
v &::= c \mid [E]\ \lambda x.e \mid (v_1, v_2) \\
E &::= - \mid (E, x \mapsto v)
\end{aligned}
$$

$$
\begin{array}{c}
(E \vdash e) \Rightarrow v \\
\rotatebox{90}{$\rightsquigarrow$} \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

Merge: $E_1 \oplus_{\text{conservative}} E_2$

$(x \mapsto 2) \oplus_{\text{conservative}} (x \mapsto 3)$ undefined

Enables a "PutGet" theorem

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$
$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$
$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

Merge: $E_1 \oplus_{\text{conservative}} E_2$

$(x \mapsto 2) \oplus_{\text{conservative}} (x \mapsto 3)$ undefined

Enables a "PutGet" theorem

Merge: $E_1 \oplus_{\text{optimistic}} E_2$

$(x \mapsto 2) \oplus_{\text{optimistic}} (x \mapsto 3) = 2$ or 3 (not 1)

Enables many desirable interactions

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$(E \vdash e_1\ e_2) \Rightarrow v \rightsquigarrow v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E\ldots \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f\ , x \mapsto v_x\ ) \vdash f\ x) \Rightarrow v$$

$$\rightsquigarrow$$

$$v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$\frac{(E_f \vdash e_f)}{((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'}$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f)$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \rightsquigarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \rightsquigarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \rightsquigarrow ((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow v'$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rightsquigarrow \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$\begin{array}{c} ((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \\ \rightsquigarrow \\ ((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v' \end{array}$$

$$\begin{array}{c} ((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \\ \rightsquigarrow \\ ((E, f \mapsto \qquad\qquad, x \mapsto \qquad) \vdash f\ x) \Leftarrow v' \end{array}$$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E, f \mapsto [E_f']\ \lambda y.e_f', x \mapsto \quad) \vdash f\ x) \Leftarrow v'$$

New Code

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid \boxed{e_1\ e_2}$$

$$v ::= c \mid [E]\ \lambda x.e \mid (v_1, v_2)$$

$$E ::= - \mid (E, x \mapsto v)$$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v \rightsquigarrow ((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v \rightsquigarrow ((E, f \mapsto [E_f']\ \lambda y.e_f', x \mapsto v_x') \vdash f\ x) \Leftarrow v'$$

New Code New Data

Key Idea

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E_f', y \mapsto v_x') \vdash e_f') \Leftarrow v'$$

$$((E, f \mapsto [E_f]\ \lambda y.e_f, x \mapsto v_x) \vdash f\ x) \Rightarrow v$$
$$\rightsquigarrow$$
$$((E, f \mapsto [E_f']\ \lambda y.e_f', x \mapsto v_x') \vdash f\ x) \Leftarrow v'$$

New Code New Data

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2$$

$$\begin{array}{c} (E \vdash e) \Rightarrow v \\ \rotatebox{90}{$\rightsquigarrow$} \\ (E' \vdash e') \Leftarrow v' \end{array}$$

$$
\begin{aligned}
e ::= \; & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
\mid \; & e_1 :: e_2 \mid [\, e_1, \ldots, e_n \,] \\
\mid \; & \{\, f_1 = e_1, \ldots, f_n = e_n \,\} \\
\mid \; & \text{let } x = e_1 \text{ in } e_2 \\
\mid \; & \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
\mid \; & e_1 + e_2 \mid \neg e \mid \ldots \\
\mid \; & \text{dict\_get } e_1\ e_2 \mid \ldots \\
\mid \; & \text{re\_match } e_1\ e_2 \mid \ldots \\
\mid \; & \text{eval } e
\end{aligned}
$$

$$
\begin{array}{c}
(E \vdash e) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

Bidirectional Evaluation with Direct Manipulation

MIKAËL MAYER, University of Chicago, USA
VIKTOR KUNČAK, École Polytechnique Fédérale de Lausanne, Switzerland
RAVI CHUGH, University of Chicago, USA

We present an *evaluation update* (or simply, *update*) algorithm for a full-featured functional programming language, which synthesizes program changes based on output changes. Intuitively, the update algorithm retraces the steps of the original evaluation, rewriting the program as needed to reconcile differences between the original and updated output values. Our approach, furthermore, allows expert users to define custom *lenses* that augment the update algorithm with more advanced or domain-specific program updates.

To demonstrate the utility of evaluation update, we implement the algorithm in Sketch-n-Sketch, a novel *direct manipulation programming system* for generating HTML documents. In Sketch-n-Sketch, the user writes an ML-style functional program to generate HTML output. When the user directly manipulates the output using a graphical user interface, the update algorithm reconciles the changes. We evaluate bidirectional evaluation in Sketch-n-Sketch by authoring ten examples comprising approximately 1400 lines of code in total. These examples demonstrate how a variety of HTML documents and applications can be developed and edited interactively in Sketch-n-Sketch, mitigating the tedious edit-run-view cycle in traditional programming environments.

CCS Concepts: • **Software and its engineering** → General programming languages; Specialized application languages; • **Human-centered computing** → Graphical user interfaces;

Additional Key Words and Phrases: Bidirectional Programming, Direct Manipulation, Sketch-n-Sketch

ACM Reference Format:
Mikaël Mayer, Viktor Kunčak, and Ravi Chugh. 2018. Bidirectional Evaluation with Direct Manipulation. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 127 (November 2018), 28 pages. https://doi.org/10.1145/3276497

1 INTRODUCTION

Expert programmers often choose to write programs to generate digital objects that might otherwise be created in graphical user interfaces (GUIs) using *direct manipulation* [Hutchins et al. 1985; Shneiderman 1983], because GUIs typically lack powerful mechanisms for abstraction and reuse. To name just a few example languages and libraries, LaTeX is particularly popular for generating documents; JavaScript, Ruby, and Elm for web applications; Processing and p5.js (http://p5js.org/) for graphic designs; LaTeX and Slideshow [Findler and Flatt 2006] for slide-based presentations; and D3 [Bostock et al. 2011] for data visualizations.

The benefits of programming, however, come at a steep cost: to change the output of a program, the user must edit the source code, run it again, and view the new output, often repeating this loop ad nauseaum. Effort wasted in this way is particularly galling when successive program changes—and the resulting output changes—are small and narrow in scope. Ideally, the user would "directly manipulate" the program output, and the system would run the program "in reverse" to synthesize necessary program repairs.

127

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
© 2018 Copyright held by the owner/author(s).
2475-1421/2018/11-ART127
https://doi.org/10.1145/3276497

Proc. ACM Program. Lang., Vol. 2, No. OOPSLA, Article 127. Publication date: November 2018.

$$
\begin{array}{rcl}
e & ::= & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
& \mid & e_1 :: e_2 \mid [\, e_1, \ldots, e_n \,] \\
& \mid & \{\, f_1 = e_1, \ldots, f_n = e_n \,\} \\
& \mid & \text{let } x = e_1 \text{ in } e_2 \\
& \mid & \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
& \mid & e_1 + e_2 \mid \neg e \mid \ldots \\
& \mid & \text{dict\_get } e_1\ e_2 \mid \ldots \\
& \mid & \text{re\_match } e_1\ e_2 \mid \ldots \\
& \mid & \text{eval } e
\end{array}
$$

$$
\begin{array}{c}
(E \vdash e\,) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

$$
\begin{aligned}
e \ ::= \ & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
\mid\ & e_1 :: e_2 \mid [\, e_1, \ldots, e_n \,] \\
\mid\ & \{\, f_1 = e_1, \ldots, f_n = e_n \,\} \\
\mid\ & \text{let } x = e_1 \text{ in } e_2 \\
\mid\ & \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
\mid\ & e_1 + e_2 \mid \neg e \mid \ldots \\
\mid\ & \text{dict\_get } e_1\ e_2 \mid \ldots \\
\mid\ & \text{re\_match } e_1\ e_2 \mid \ldots \\
\mid\ & \text{eval } e
\end{aligned}
$$

$$
\begin{array}{c}
(E \vdash e\,) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

Structural updates through, e.g.,

List.map f' xs' ⇐ vs' ?

$$
\begin{array}{rcl}
e & ::= & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
& \mid & e_1 :: e_2 \mid [\, e_1, \ldots, e_n \,] \\
& \mid & \{\, f_1 = e_1, \ldots, f_n = e_n \,\} \\
& \mid & \text{let } x = e_1 \text{ in } e_2 \\
& \mid & \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
& \mid & e_1 + e_2 \mid \neg e \mid \ldots \\
& \mid & \text{dict\_get } e_1\ e_2 \mid \ldots \\
& \mid & \text{re\_match } e_1\ e_2 \mid \ldots \\
& \mid & \text{eval } e
\end{array}
$$

$$
\begin{array}{c}
(E \vdash e\,) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

Structural updates through, e.g.,

List.map f' xs' $\Leftarrow$ **vs'** ?

Domain-specific primitives?

$$
\begin{array}{rcl}
e & ::= & c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \\
& \mid & e_1 :: e_2 \mid [\, e_1, \ldots, e_n \,] \\
& \mid & \{\, f_1 = e_1, \ldots, f_n = e_n \,\} \\
& \mid & \text{let } x = e_1 \text{ in } e_2 \\
& \mid & \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
& \mid & e_1 + e_2 \mid \neg e \mid \ldots \\
& \mid & \text{dict\_get } e_1\ e_2 \mid \ldots \\
& \mid & \text{re\_match } e_1\ e_2 \mid \ldots \\
& \mid & \text{eval } e \\
& \mid & \text{appLens } (e_{get}, e_{put})\ e_x
\end{array}
$$

$$
\begin{array}{c}
(E \vdash e\,) \Rightarrow v \\
\rightsquigarrow \\
(E' \vdash e') \Leftarrow v'
\end{array}
$$

Structural updates through, e.g.,

List.map f' xs' ⇐ vs' ?

Domain-specific primitives?

User-defined lenses for customization

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash \text{appLens}\ (f_{get}, f_{put})\ e_x) \Rightarrow$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow v_y$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow v_y$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow v_y$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow v_y \rightsquigarrow v_y'$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow v_y$$

$$(E \vdash f_{put}\ v_x\ v_y\ v_y') \Rightarrow$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow v_y \rightsquigarrow v_y'$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v \rightsquigarrow (E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow v_y$$

$$(E \vdash f_{put}\ v_x\ v_y\ v_y') \Rightarrow v_x'$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow v_y \rightsquigarrow v_y'$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow v_y$$
$$(E \vdash f_{put}\ v_x\ v_y\ v_y') \Rightarrow v_x'$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow v_y$$
$$\rightsquigarrow$$
$$((E, x \mapsto v_x') \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Leftarrow v_y'$$

$e ::= \ldots \mid \text{appLens}\ (e_{get}, e_{put})\ e_x$

$$(E \vdash e) \Rightarrow v$$
$$\rightsquigarrow$$
$$(E' \vdash e') \Leftarrow v'$$

$$(E \vdash f_{get}\ v_x) \Rightarrow v_y$$
$$(E \vdash f_{put}\ v_x\ v_y\ v_y') \Rightarrow v_x'$$

$$((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Rightarrow v_y$$
$$\rightsquigarrow$$
$$((E, x \mapsto v_x') \vdash \text{appLens}\ (f_{get}, f_{put})\ x) \Leftarrow v_y'$$

Related Work

Bidirectional Programming

Lenses [Foster et al. 2007, many others]

DSLs for relational data, trees, strings, graphs

Point-free programming style

HOBiT [Matsuda and Wang 2018]

Instead of lens combinators, staged evaluator for λ-calculus

First evaluation stage eliminates function calls ($\mathbf{e_1}\ \mathbf{e_2}$)

Bidirectional evaluation for "residual" first-order programs

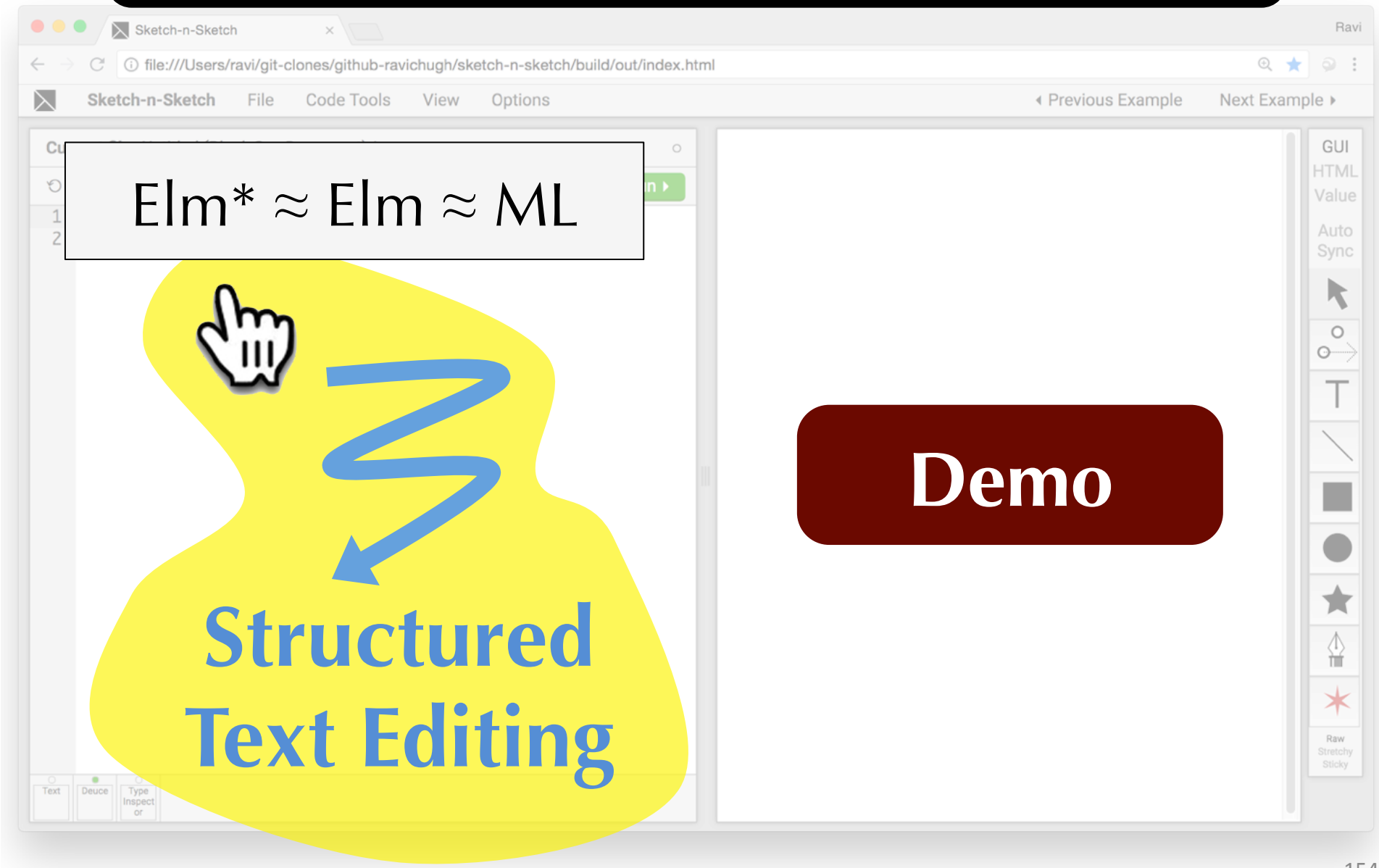
More direct support for "branch switching" than our approach

Related Work

Sketch-n-Sketch

	[Chugh et al. PLDI 2016]	[OOPSLA 2018]
Repair Algorithm	Evaluation with tracing + constraint solving	Re-run evaluation "in reverse" only **as needed** Lens API for **customization**
User Interface	Little λ-calculus ⇒ SVG Auto disambiguation via simple heuristics	**Elm**-like language ⇒ **HTML** **User disambiguation** via dropdown menu

Sketch-n-Sketch : Deuce Code Editor



Direct Code Manipulation

Sketch-n-Sketch

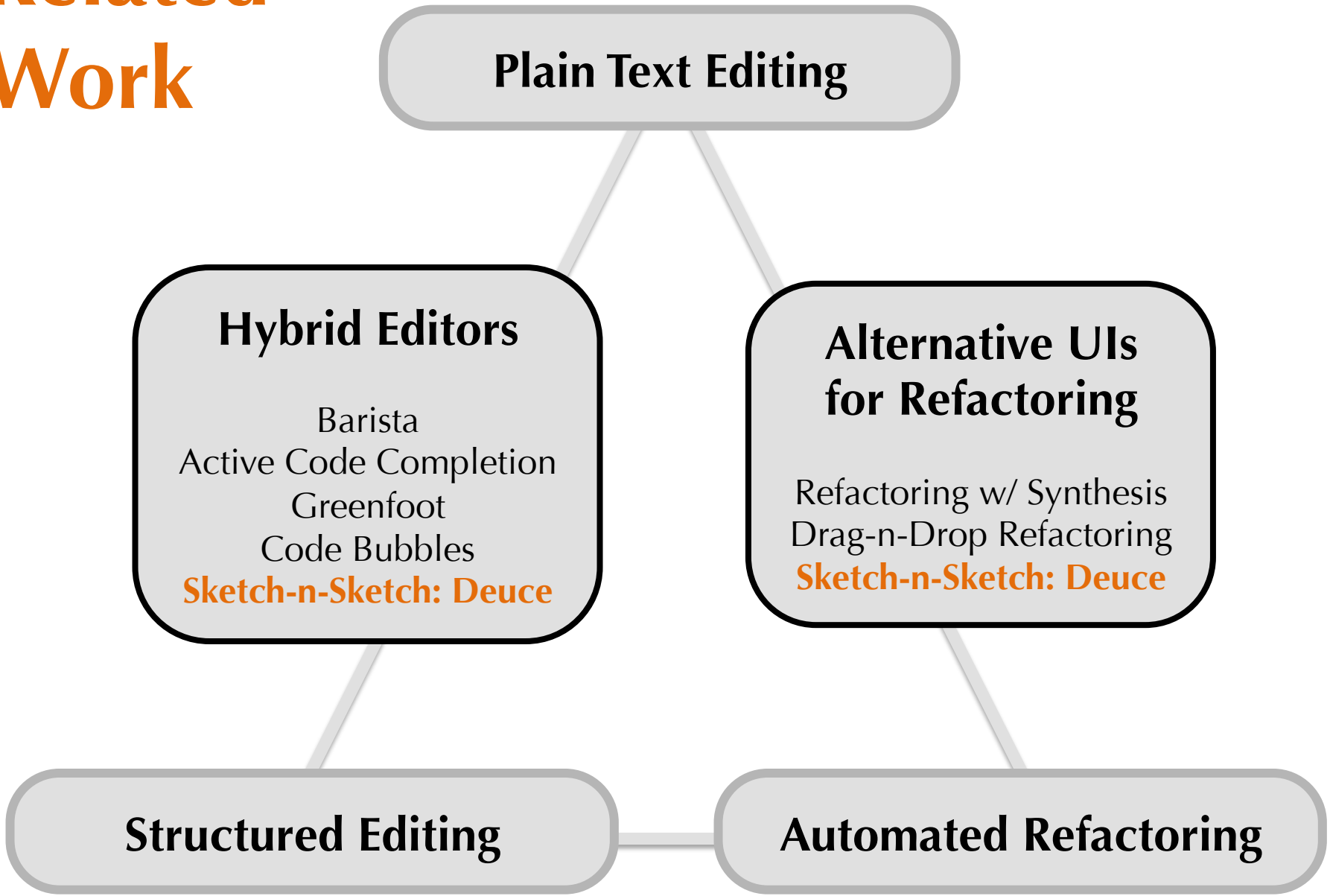
Brian @ ICSE 2018

Ravi @ elm-conf 2018

Structured Text Editing

[ICSE 2018, work-in-progress]

Related Work



Lightweight Structured Editing via Direct Manipulation

Structural (Multi-)Selection; Context-Sensitive Tool Menu; Hover Preview and Confirm

Traditional UI

Text Selection; Tool Menu; Configuration Wizard

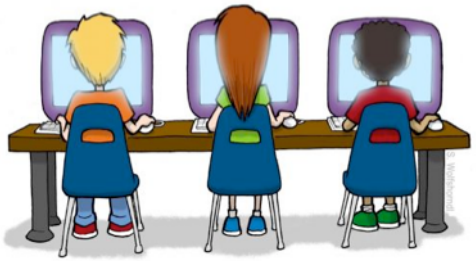
Automated Refactoring

1. Learning Curve for New UI + Tools
2. Faster Tool Invocations using New UI
3. Overwhelming Preference for New UI

Structural (Multi-)Selection;
Context-Sensitive Tool Menu;
Hover Preview and Confirm

Traditional UI

Text Selection;
Tool Menu;
Configuration Wizard



21 users

END