Reimagining the User Interfaces for Programming

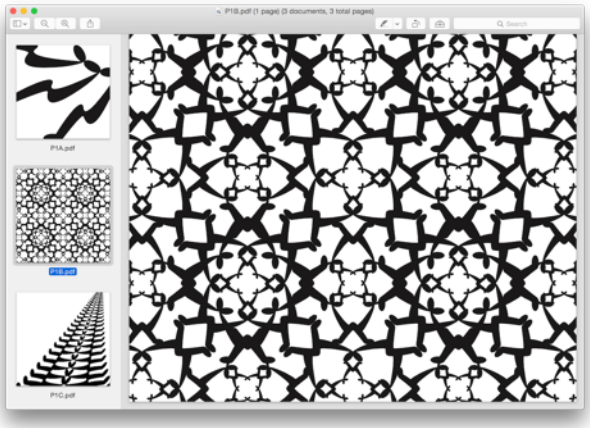
Ravi Chugh

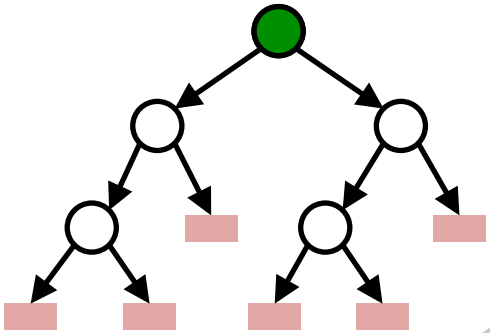
THE UNIVERSITY OF CHICAGO

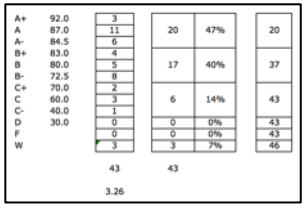
10.02.2020

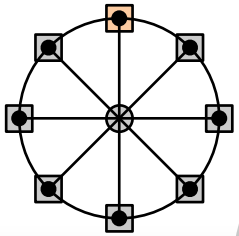
User Interfaces for Programming

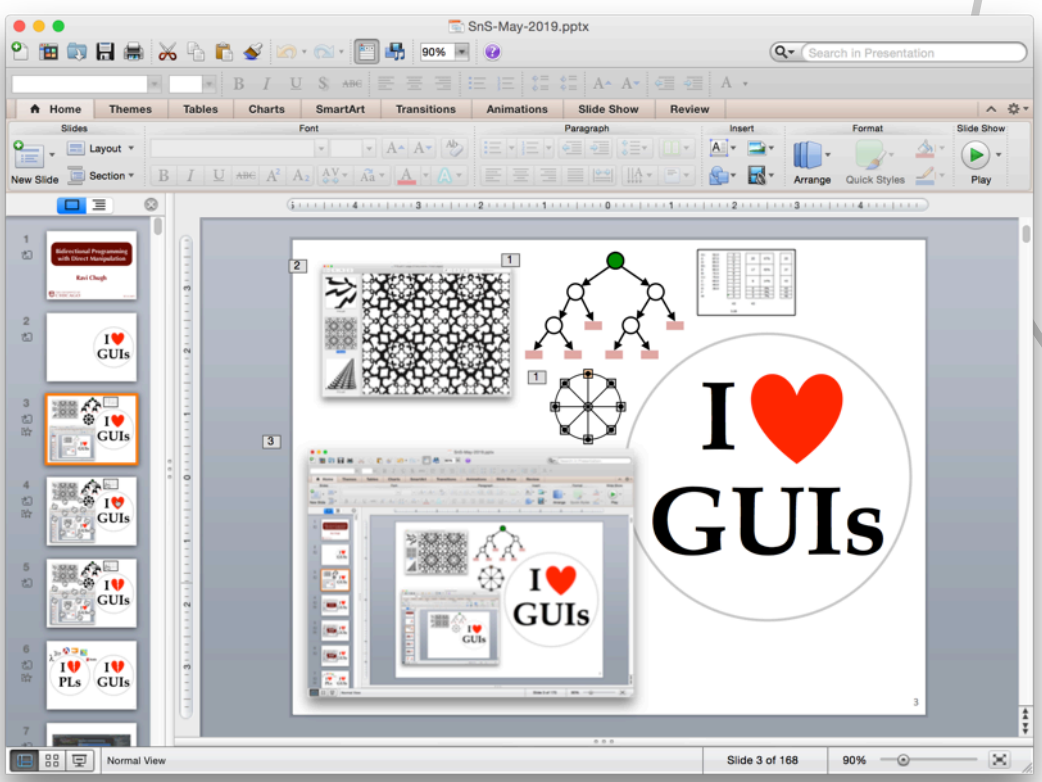
I
GUIs







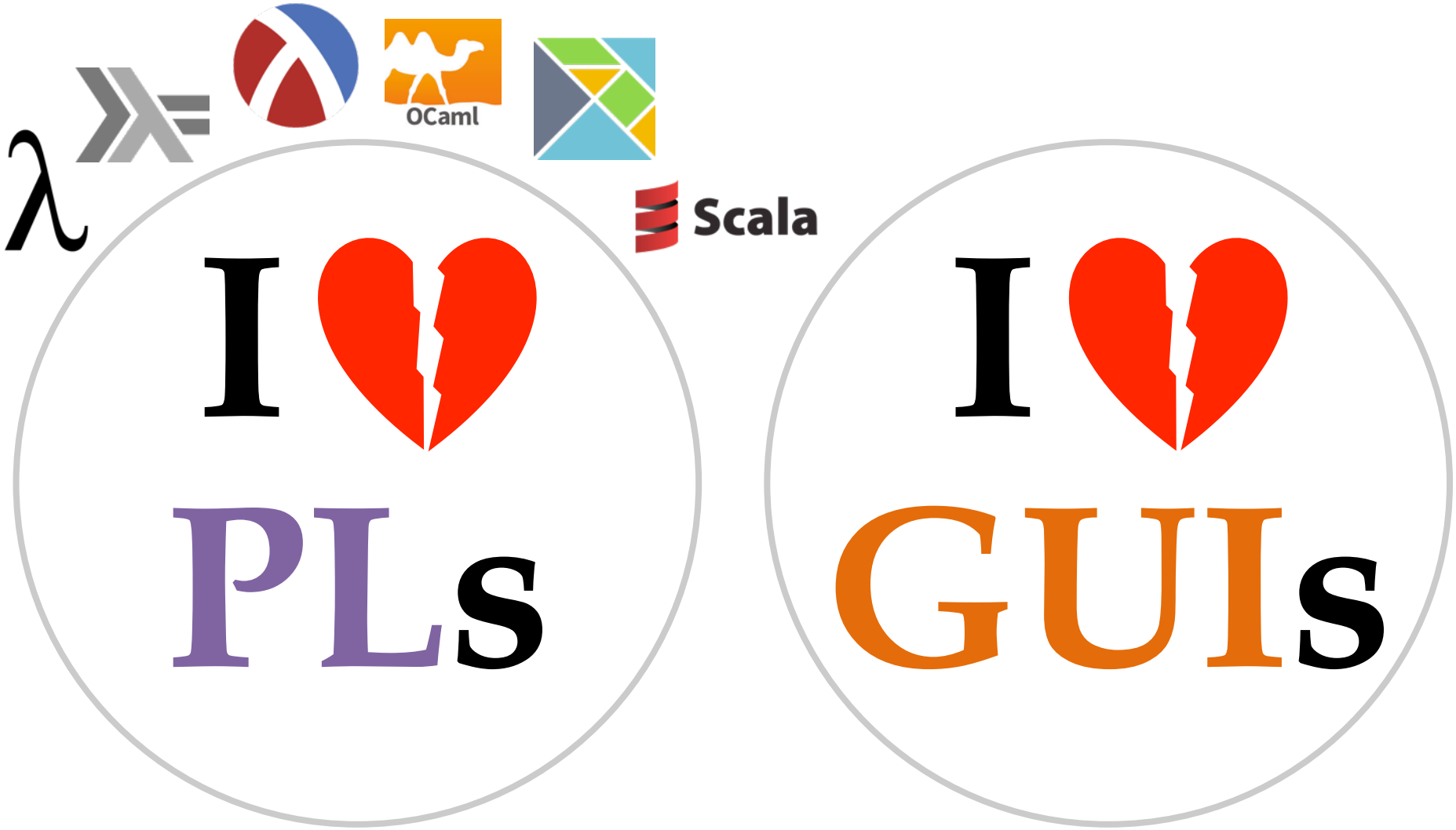


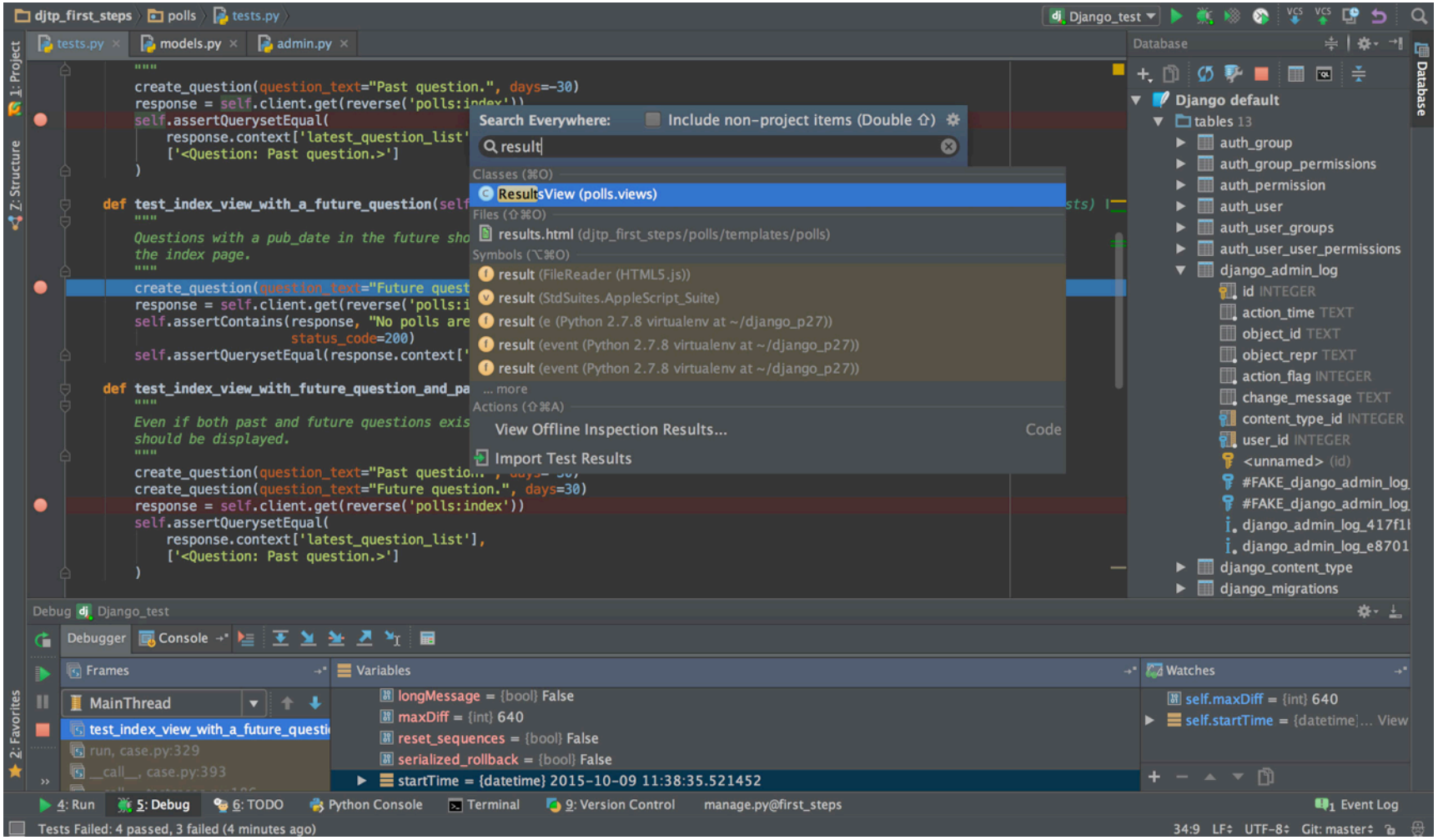

SnS-May-2019.pptx
I ♥ GUIs

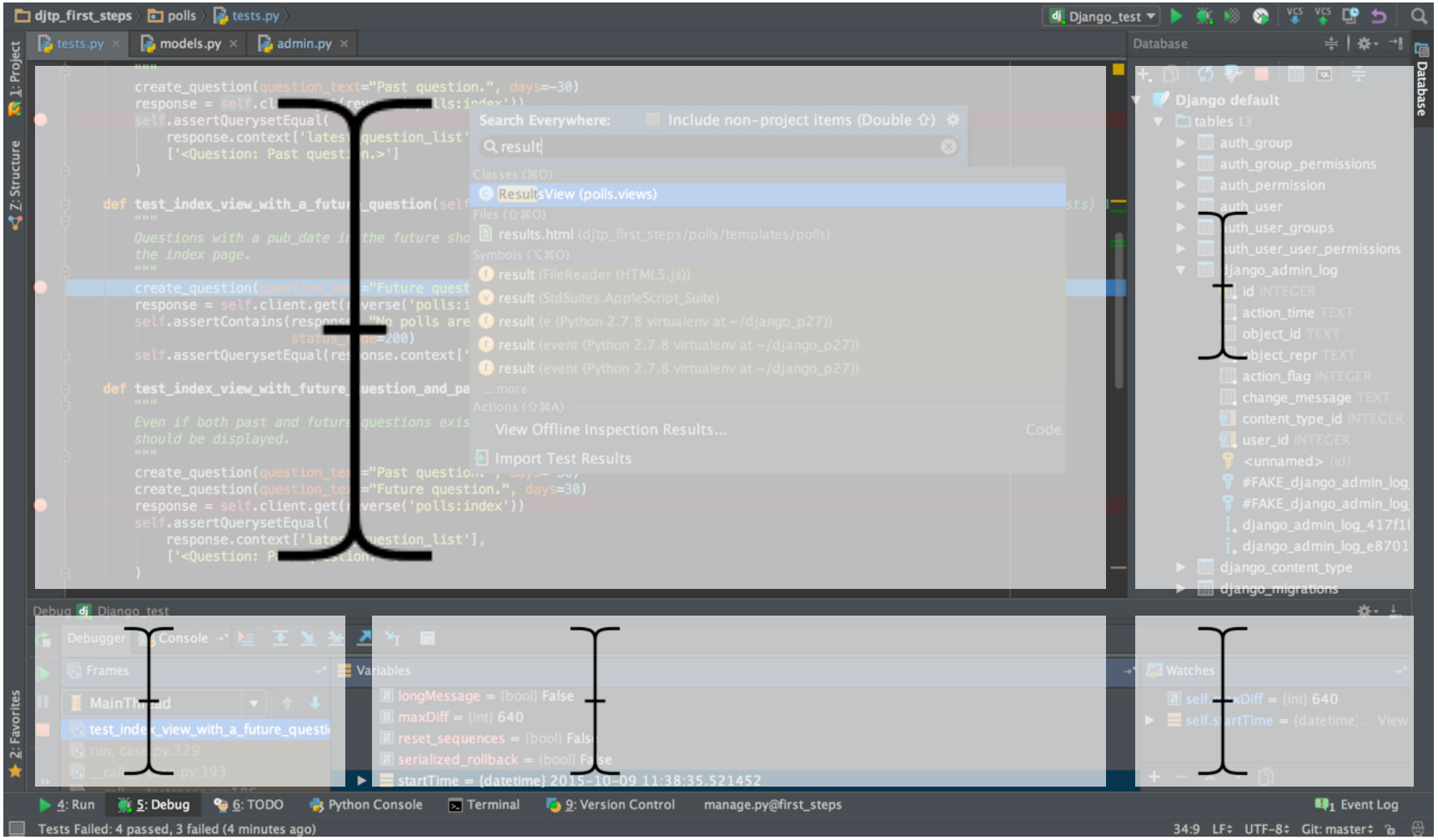
I ♥ GUIs

I
GUIs

I ♥ GUIs


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
Variables
Watches
MainThread
test_index_view_with_a_future_questi
run, case.py:329
__call__, case.py:393
longMessage = {bool} False
maxDiff = {int} 640
reset_sequences = {bool} False
serialized_rollback = {bool} False
startTime = {datetime} 2015-10-09 11:38:35.521452
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
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
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

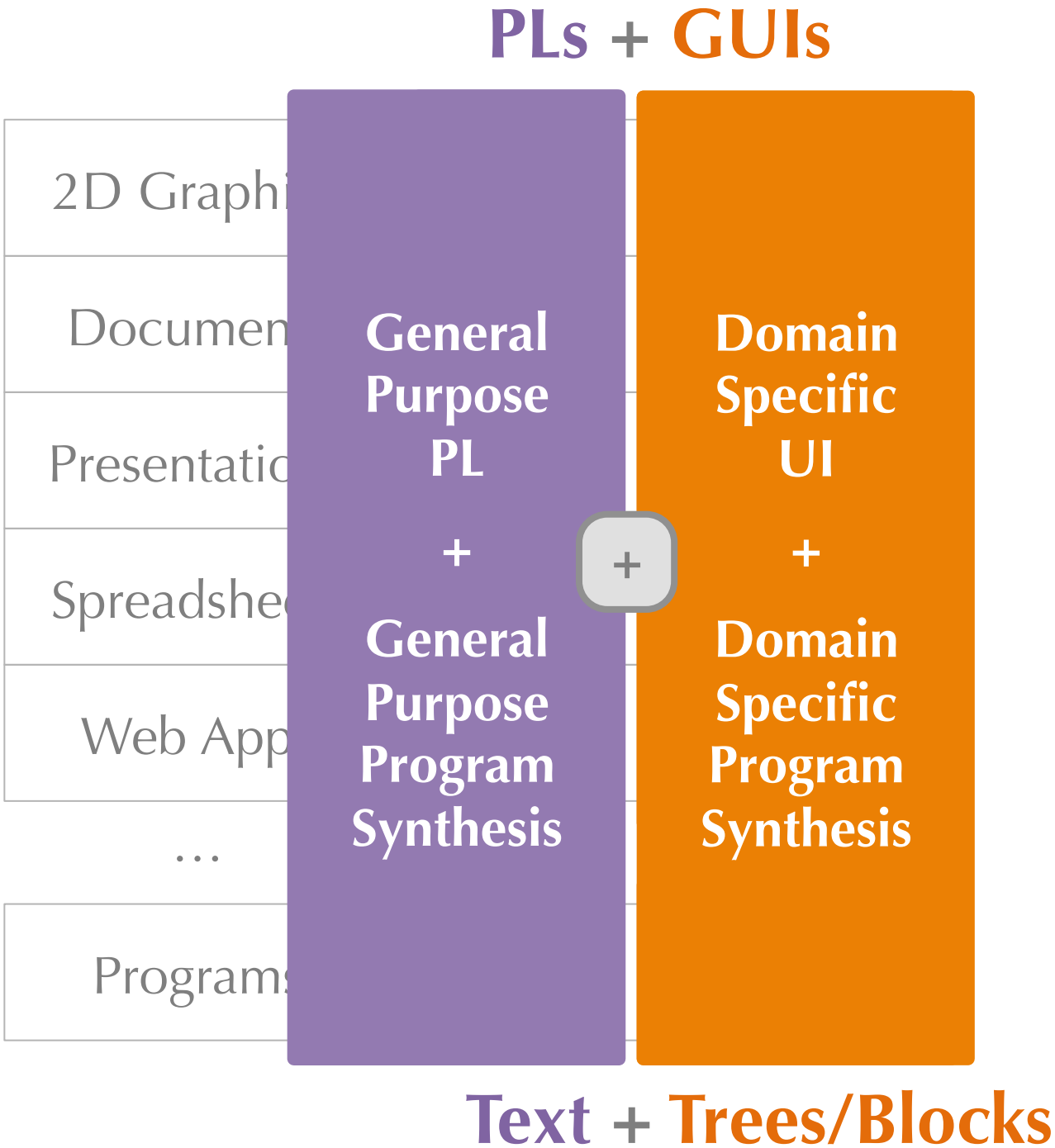
I
PLs

I
GUIs

	PLs	GUIs
2D Graphics		
Documents	LaTeX, Scholarly Markdown	
Presentations	LaTeX	
Spreadsheets	D3, SQL	
Web Apps	JS	

...

	PLs	GUIs
2D Graphics		
Documents		
Presentations		
Spreadsheets		
Web Apps		
…		
Programs		
	Text	Trees/Blocks


PLs + GUIs
2D Graphi
Documen
Presentatic
Spreadshe
Web App
...
Program
General Purpose PL
+
General Purpose Program Synthesis
+
Domain Specific UI
+
Domain Specific Program Synthesis
Text + Trees/Blocks

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

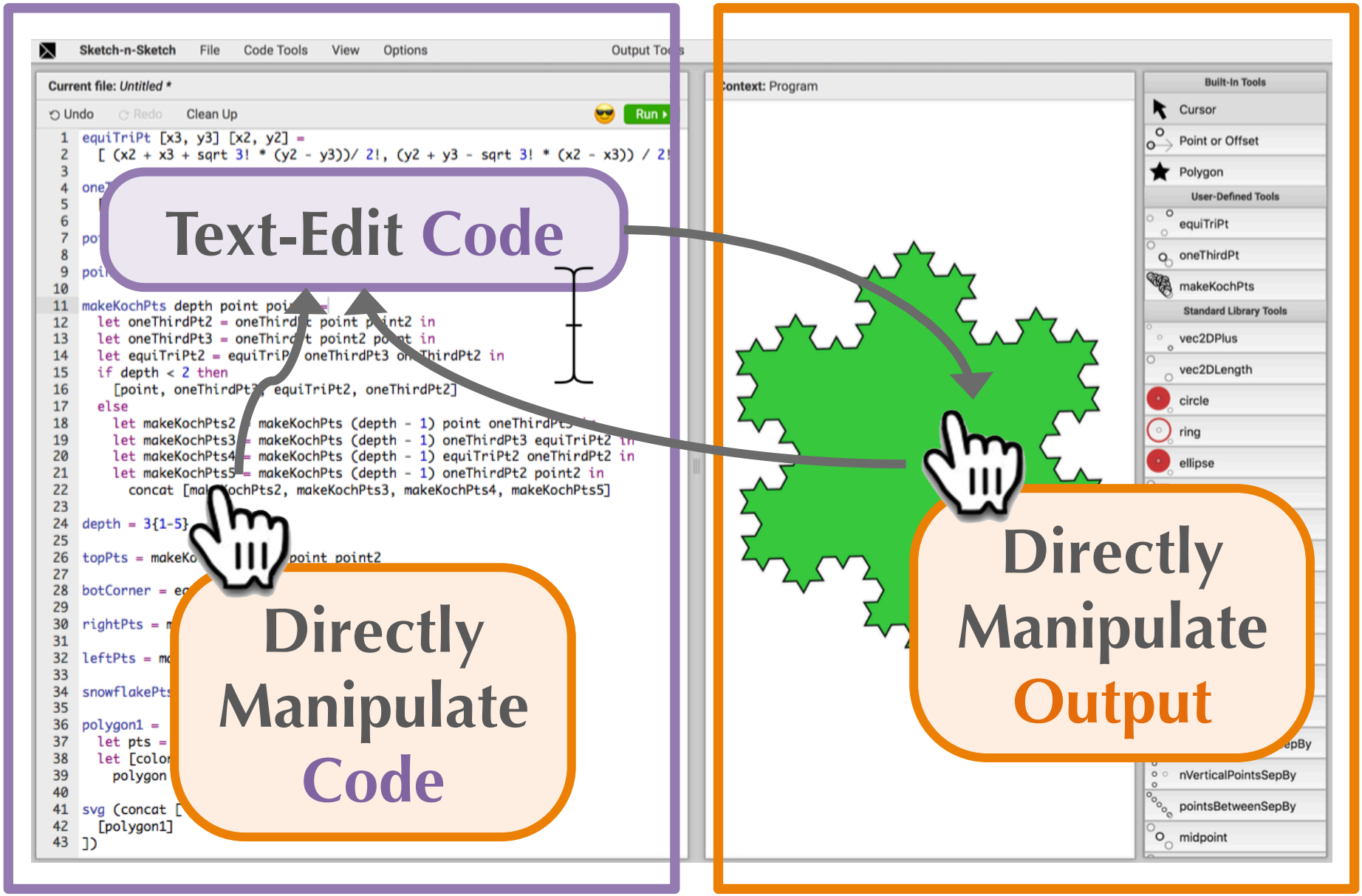
Hybrid Text-GUI Code Editors

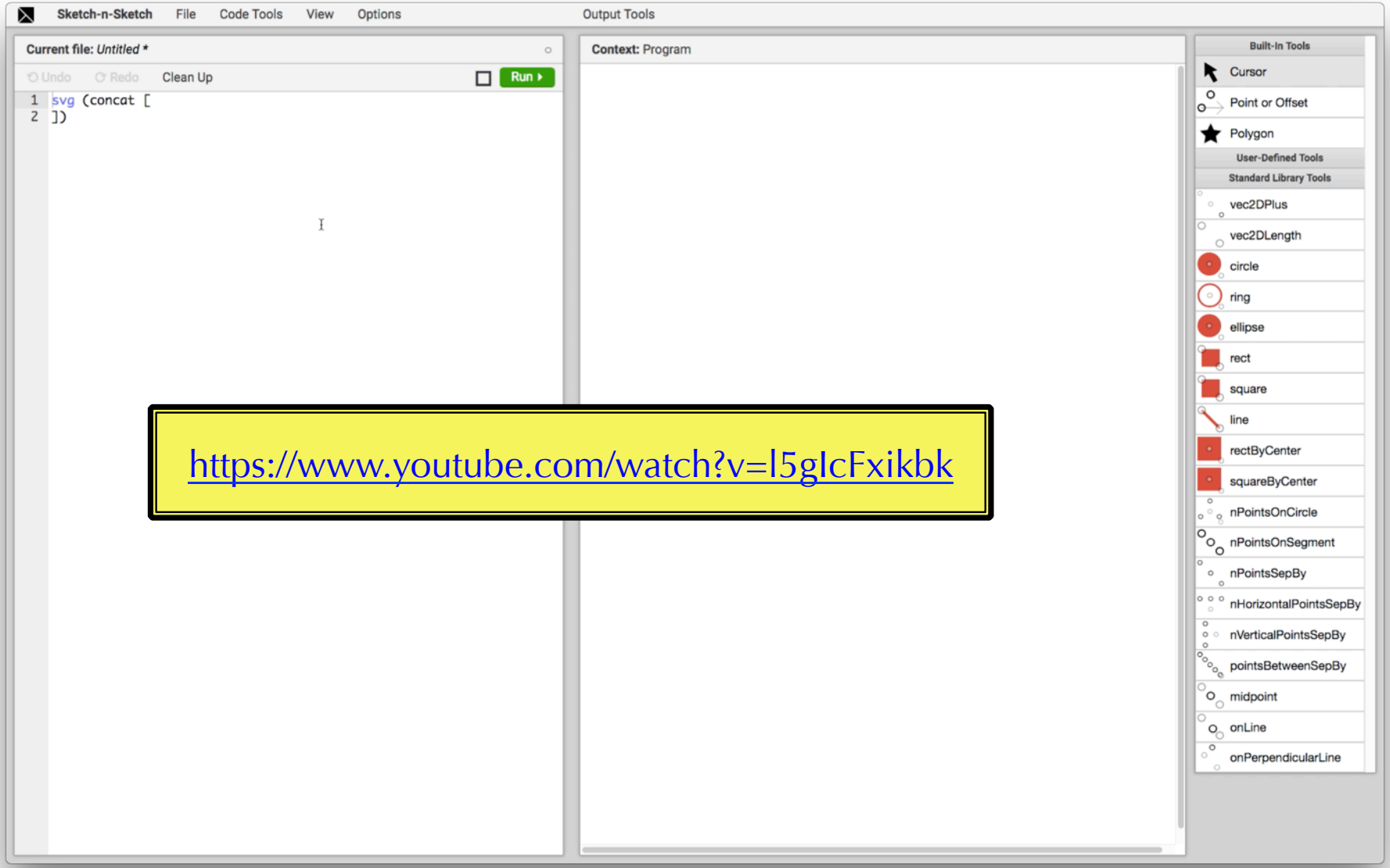
[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

Sketch the **Code** + Sketch the **Output**





https://www.youtube.com/watch?v=l5gIcFxikbk

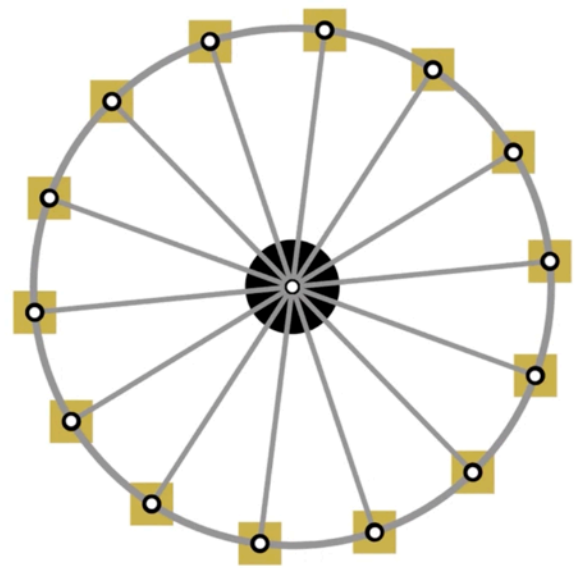
Sketch-n-Sketch File Examples Code Tools Output Tools View Options

Current file: *Untitled ((xi) Ferris Wheel) **

Undo Redo Clean Up Run

```
point = [307, 334]

r = 241

attachmentPts = nPointsOnCircle 14{0-20} 0.12560000000000002{ 3.1

color = 434

spokeFunc point2 =
  line color 5 point point2

spokes =
  map spokeFunc attachmentPts

carFunc center2 =
  squareByCenter 48 center2 20

cars =
  map carFunc attachmentPts

capFunc point2 =
  circle 364 point2 9

caps =
  map capFunc attachmentPts

ring1 = ring color 7 point r

hub = circle 362 point 44

svg (concat [
  [hub],
  cars,
  spokes,
  [ring1],
  caps
])
```

Context: Program



rot: 0.125..

n: 14

Built-In Tools

Cursor

Point or Offset

Polygon

User-Defined Tools

Standard Library Tools

vec2DPlus

vec2DLength

circle

ring

ellipse

rect

square

line

rectByCenter

squareByCenter

nPointsOnCircle

nPointsOnSegment

nPointsSepBy

nHorizontalPointsSepBy

nVerticalPointsSepBy

pointsBetweenSepBy

midpoint

onLine

onPerpendicularLine

https://www.youtube.com/watch?v=6TGBTuuaV3E

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

Current file: Untitled (1a: Table of States)

Undo Redo Run ▸

```
states =
  [ ["Alabama", "AL?", ""]
  , ["Alaska", "AL?", ""]
  , ["Arizona", "AR?", ""]
  , ["Arkansas", "AR?", ""]
  , ["California", "CA", ""]
  , ["Colorado", "CO?", ""]
  , ["Connecticut", "CO?", ""] ]

main =
  let headers = ["State", "Capital"] in
  let rows =
    List.map
      (\[state, abbrev, cap] -> [state, cap + ", " + abbrev])
      states
  in
  let padding = ["padding", "3px"] in
  let headerRow =
    let styles = [padding] in
    Html.tr [] [] (List.map (Html.th styles []) headers)
  in
  let stateRows =
    let colors = ["lightgray", "white"] in
    let drawRow i row =
      let color = List.nth colors (mod i (List.length colors)) in
      let columns =
        List.map
          (Html.
          row
      in
      Html.tr []
    in
    List.indexedMap drawRow rows
  in
  Html.table [padding] [] (headerRow :: stateRows)
```

State	Capital
Alabama	, AL?
Alaska	, AL?
Arizona	, AR?
Arkansas	, AR?
California	, CA
Colorado	, CO?
Connecticut	, CO?

GUI
HTML
Value
Auto Sync

https://www.youtube.com/watch?v=pp6yQPrd6bw

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

Current file: *Untitled (Get Started)* *

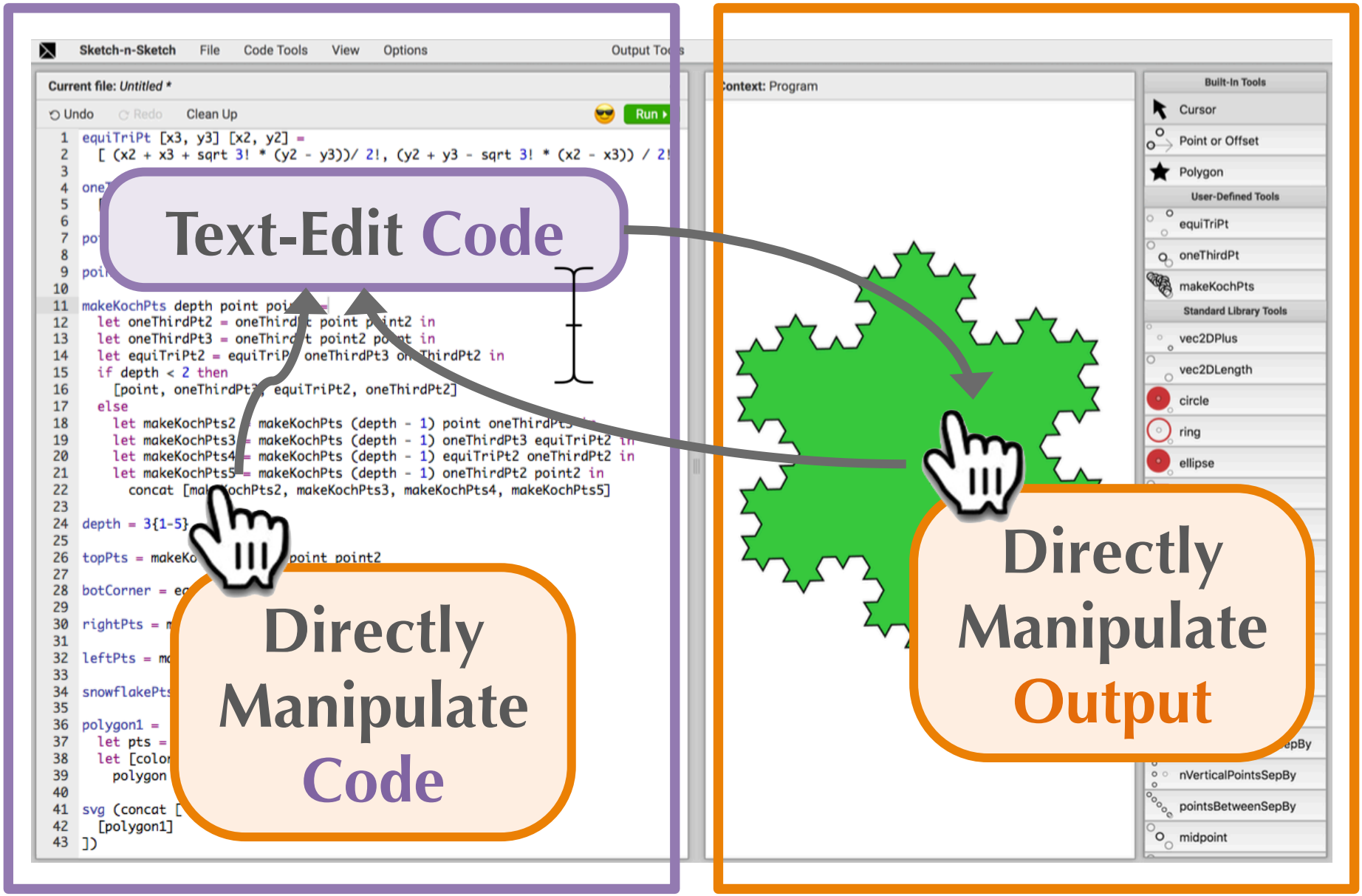
Undo Redo Run

```
(x, y, size) = (23, 30, 217)

rect1 =
  rect "slategray" x y size size

line1 =
  line "white" 10 x y (x + size) (y + size)

line2 =
  line "white" 10 x (y + size)

main = svg [rect1, line1, line2]
```

Raw
Stretchy
Sticky

https://www.youtube.com/watch?v=GE6KSG5Q__0

Sketch the **Code** + Sketch the **Output**



Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

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

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

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

$\Rightarrow$

```
"B 2018
@ PLW 2018"
```

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

```
"Bx 2019
@ PLW 2018"
```

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

```
"Bx 2019
@ PLW 2018"
```

⇐

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

```
"Bx 2019
@ PLW 2018"
```

⇐

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

```
"Bx 2019
@ PLW 2018"
```

⇐

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

⇒

```
"Bx 2019
@ PLW 2019"
```

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

```
"Bx 2019
@ PLW 2018"
```

⇐

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

⇒

```
"Bx 2019
@ PLW 2019"
```

Unrestricted function

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

```
"Bx 2019
@ PLW 2018"
```

⇐

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

⇒

```
"Bx 2019
@ PLW 2019"
```

Unrestricted function

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

Repair "code" + "data"

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

⇐

```
"Bx 2019
@ PLW 2018"
```

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

⇒

```
"Bx 2019
@ PLW 2019"
```

Unrestricted function

```
let f x = x ++ " 2018" in
f "B" ++ " @ " ++ f "PLW"
```

⇒

```
"B 2018
@ PLW 2018"
```

⇝

Repair "code" + "data"

```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

⇐

```
"Bx 2019
@ PLW 2018"
```

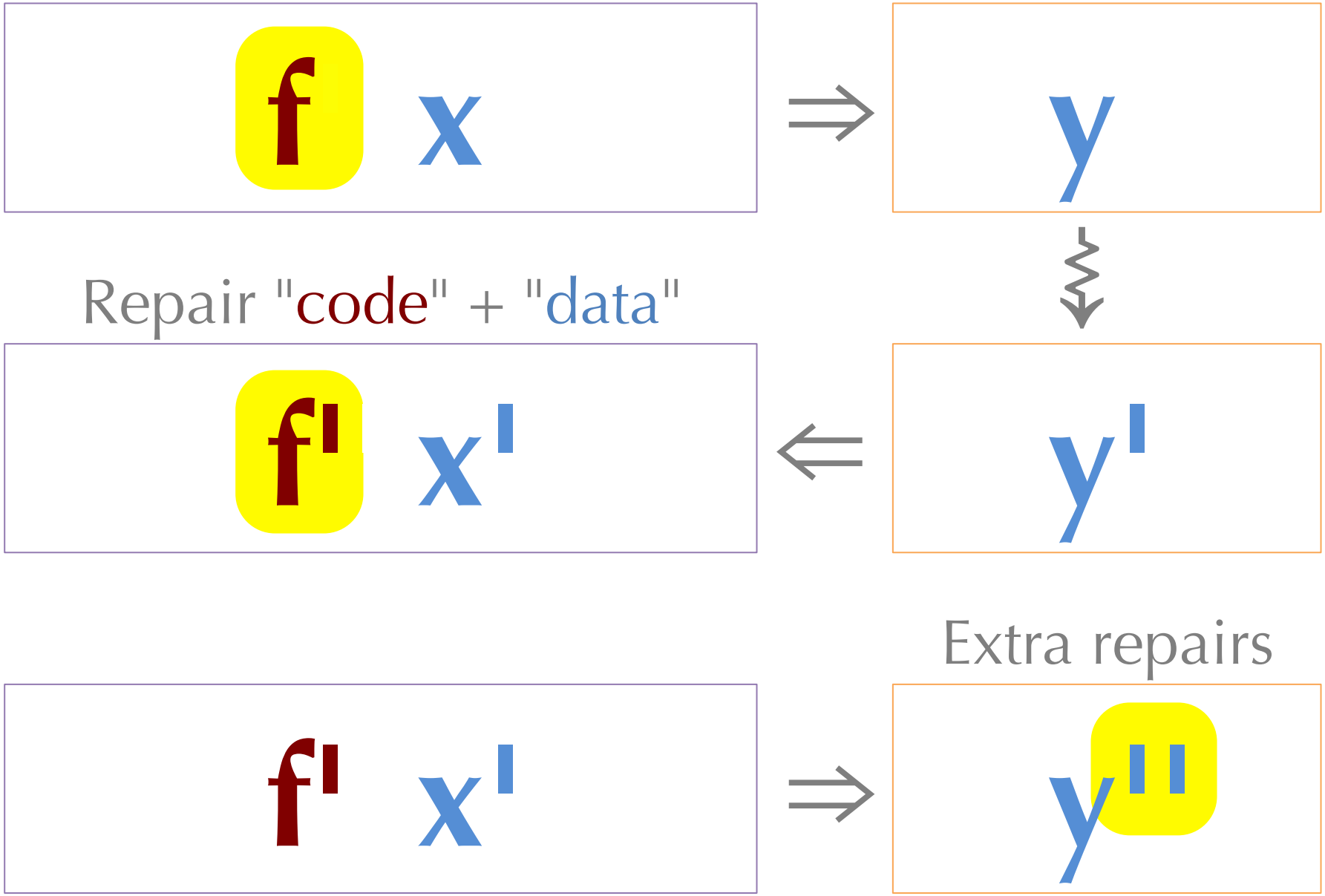
Extra repairs

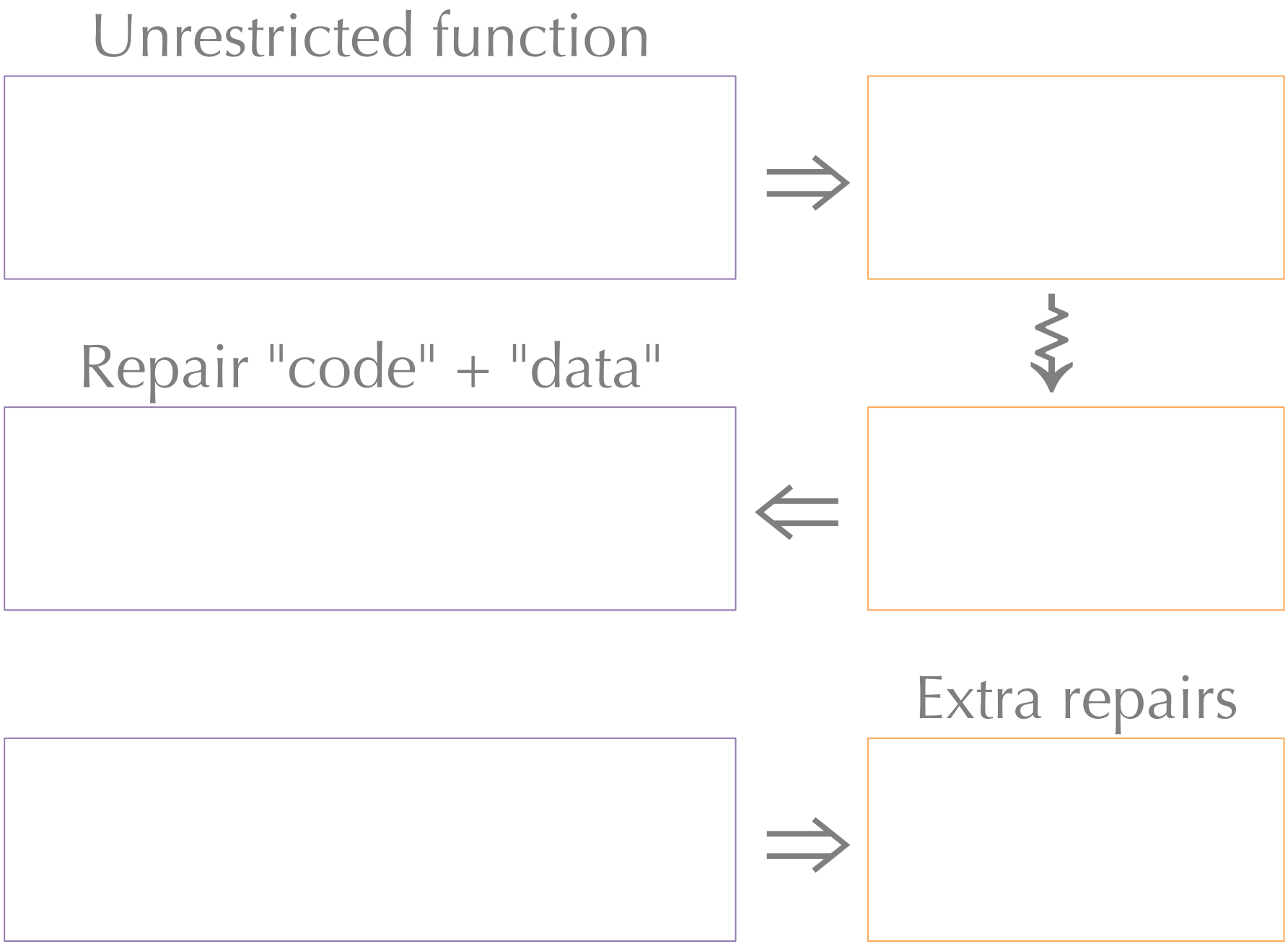
```
let f x = x ++ " 2019" in
f "Bx" ++ " @ " ++ f "PLW"
```

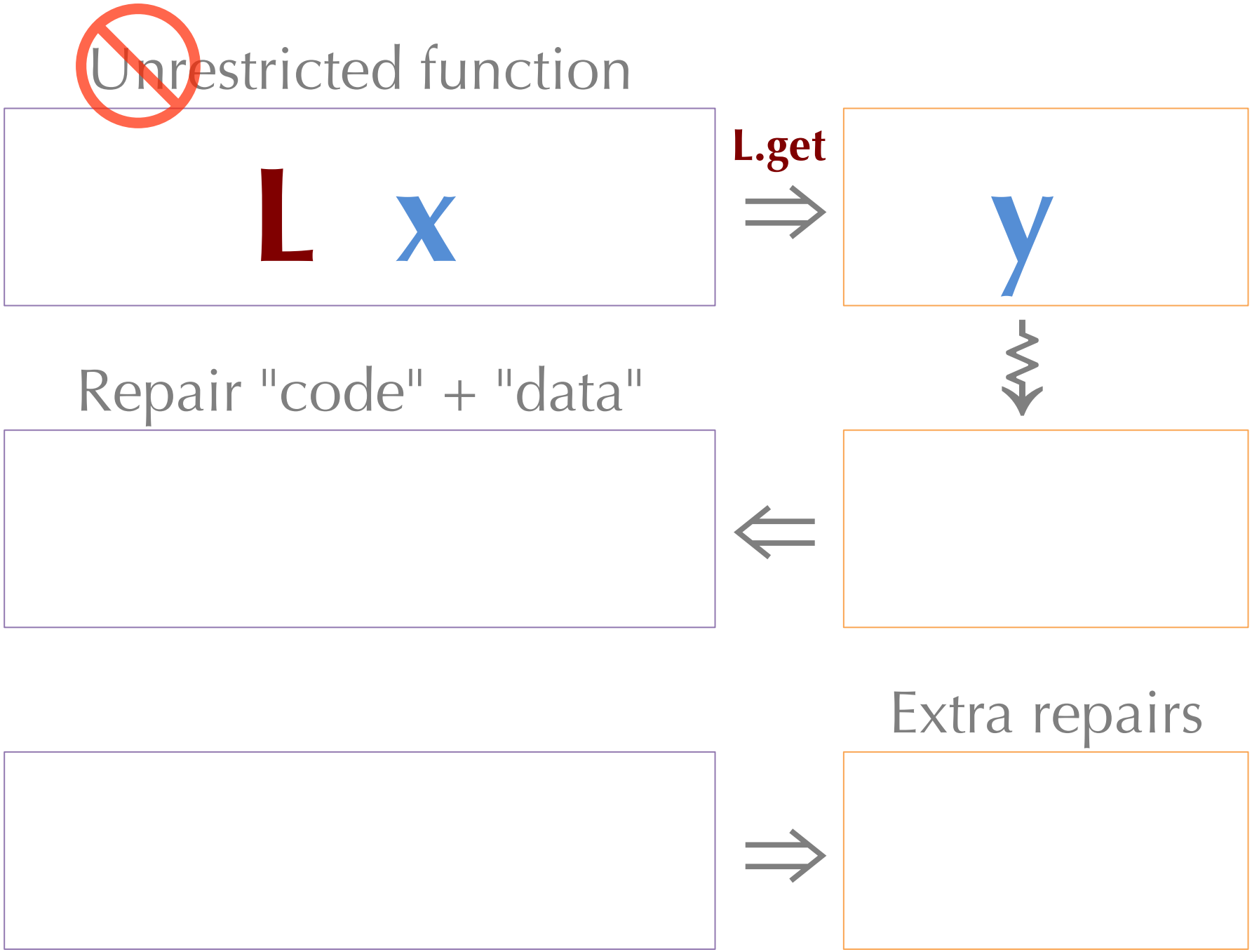
⇒

```
"Bx 2019
@ PLW 2019"
```

Unrestricted function




Unrestricted function
Repair "code" + "data"
Extra repairs


Unrestricted function
L x
L.get
y
Repair "code" + "data"
Extra repairs

Unrestricted function

L x

L.get ⇒

y

Repair "code" + "data"

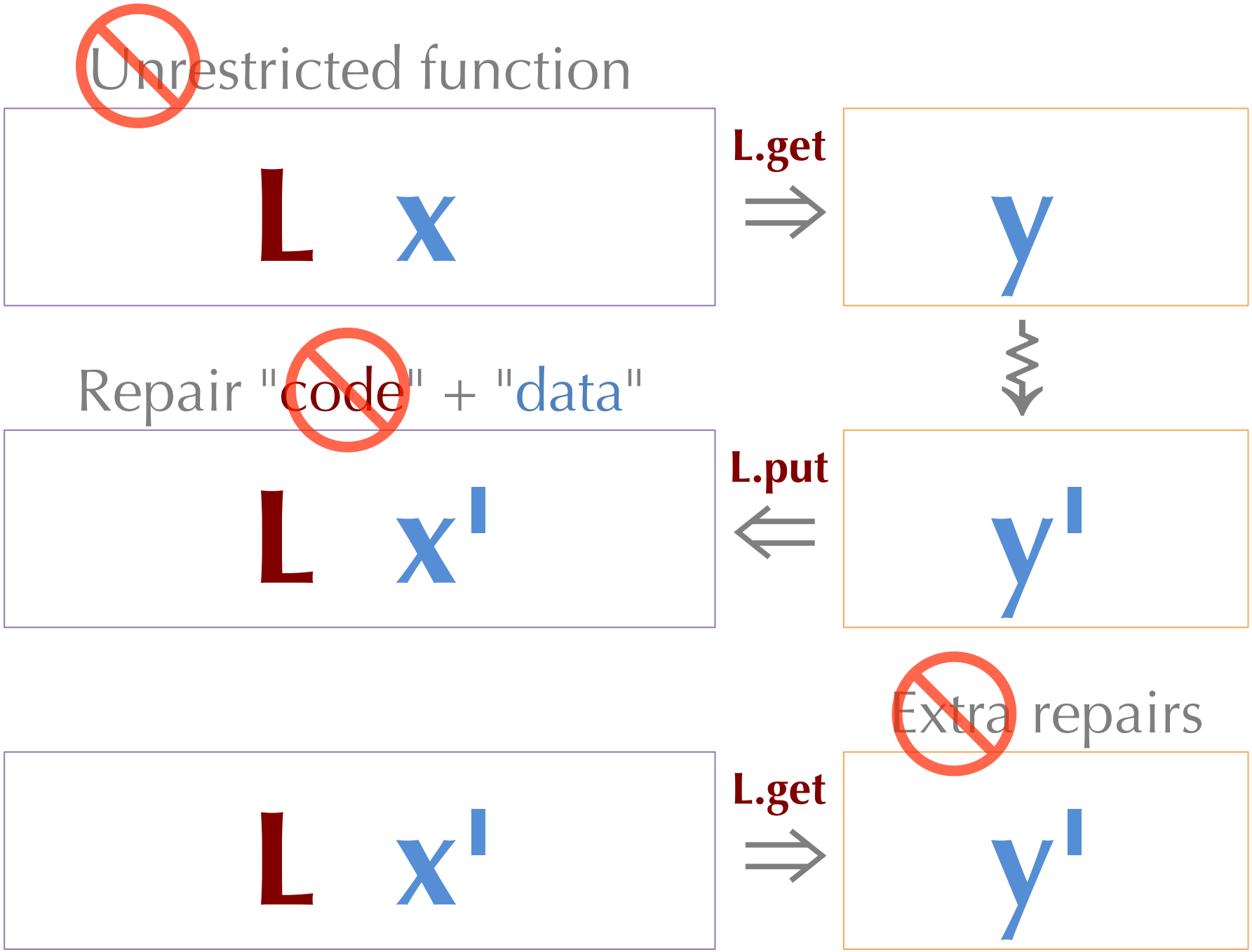
L x'

L.put ⇐

y'

Extra repairs

⇒


Unrestricted function
L x
L.get
y
Repair "code" + "data"
L x'
L.put
y'
Extra repairs
L x'
L.get
y'

Bidirectional Evaluation

Sketch-n-Sketch

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 [lightgray] 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

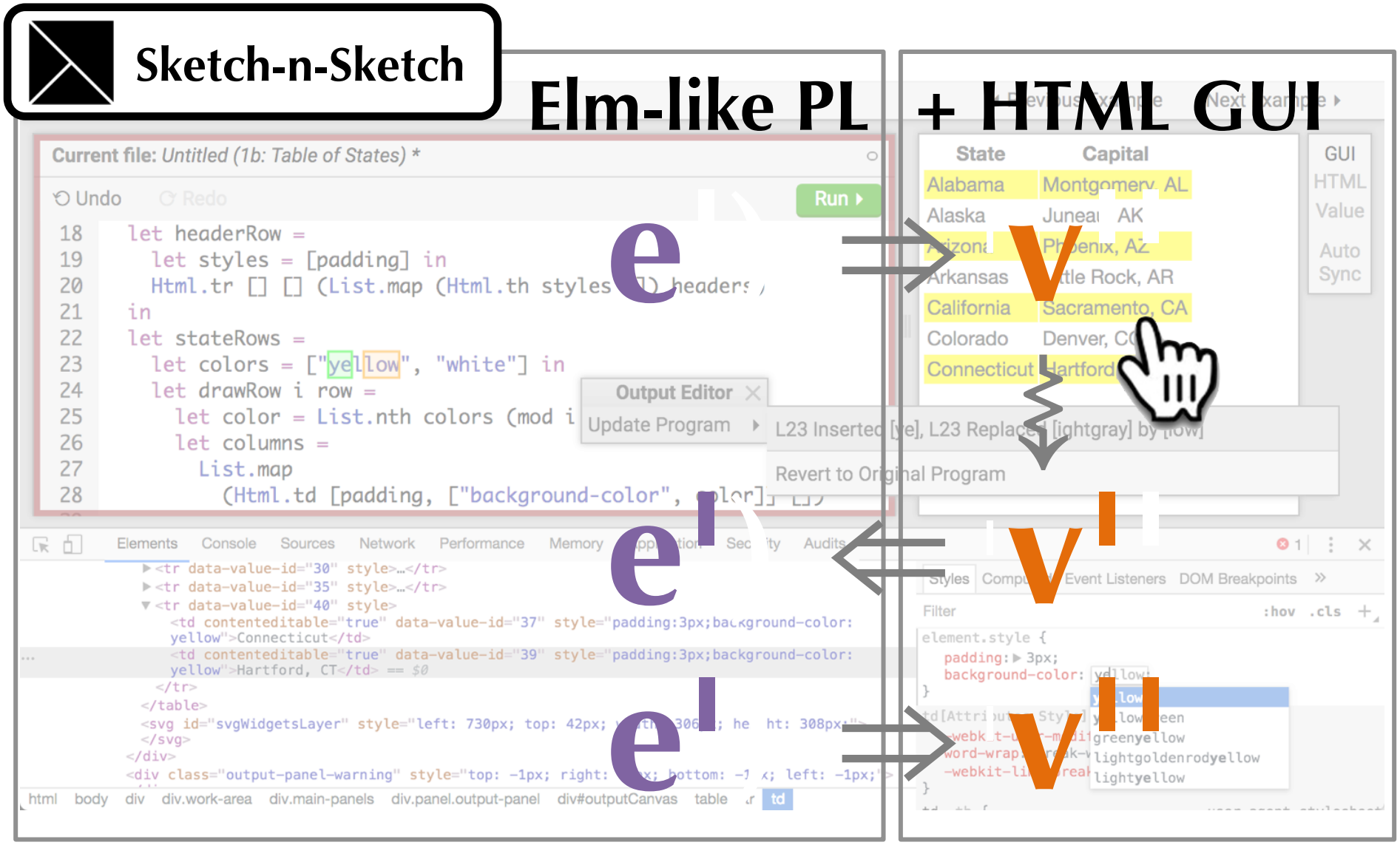
Styles Computed Event Listeners DOM Breakpoints

Filter :hov .cls

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

yellow
yellowgreen
greenyellow
lightgoldenrodyellow
lightyellow

Bidirectional Evaluation



Bidirectional Evaluation

$$
\begin{tikzcd}
e \arrow[r, Rightarrow, "\text{eval}"] & v \arrow[d, squiggly] \\
e' & v' \arrow[l, Leftarrow, "\text{uneval}"']
\end{tikzcd}
$$

[>>> Skip to Benchmarks >>>]

Bidirectional Evaluation

$$(E \vdash e) \overset{\text{eval}}{\Rightarrow} v$$

$$\rightsquigarrow$$

$$(E' \vdash e') \overset{\text{uneval}}{\Leftarrow} v'$$

Environment-style
simpler than
substitution-style

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

$$
\begin{tikzcd}
(E \vdash e) \arrow[r, Rightarrow, "\text{eval}"] & v \arrow[d, squiggly] \\
(E' \vdash e') & v' \arrow[l, Leftarrow, "\text{uneval}"']
\end{tikzcd}
$$

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

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

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

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

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

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

$$
\begin{array}{c}
(- \vdash 2018) \Rightarrow 2018 \\
\rightsquigarrow \\
2019
\end{array}
$$

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

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

$$
\begin{array}{c}
(- \vdash 2018) \Rightarrow 2018 \\
\rightsquigarrow \\
(- \vdash 2019) \Leftarrow 2019
\end{array}
$$

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

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

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

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

$$
\begin{array}{c}
(- \vdash \text{let } x = 2018 \text{ in } x) \Rightarrow 2018 \\
\rightsquigarrow \\
2019
\end{array}
$$

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

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

$$
\begin{array}{c}
((x \mapsto 2018) \vdash x) \Rightarrow 2018 \\
\rightsquigarrow \\
2019
\end{array}
$$

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

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

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

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

$$\begin{array}{c} ((x \mapsto 2018) \vdash x) \Rightarrow 2018 \\ \rightsquigarrow \\ ((x \mapsto 2019) \vdash x) \Leftarrow 2019 \end{array}$$

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

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

$$
\begin{array}{c}
(- \vdash \text{let } x = 2018 \text{ in } x) \Rightarrow 2018 \\
\rightsquigarrow \\
(- \vdash \text{let } x = 2019 \text{ in } x) \Leftarrow 2019
\end{array}
$$

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

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

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

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

$$
\begin{array}{c}
(- \vdash \text{let } x = 18 \text{ in } (x, x)) \Rightarrow (18, 18) \\
\rightsquigarrow \\
(19, 18)
\end{array}
$$

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

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

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

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

$$((x \mapsto 18) \vdash (x, x)) \Rightarrow (18, 18)$$

$$\rightsquigarrow$$

$$(19, 18)$$

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

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

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

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

$$((x \mapsto 18) \vdash (x, x)) \Rightarrow (18, 18)$$

$$\rotatebox{90}{$\rightsquigarrow$}$$

$$((x \mapsto \boxed{\begin{array}{c}18?\\19?\end{array}}) \vdash (x, x)) \Leftarrow (19, 18)$$

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

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

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

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

$$((x \mapsto 18) \vdash (x, x)) \Rightarrow (18, 18)$$

$$\rightsquigarrow \qquad \rightsquigarrow$$

$$((x \mapsto \boxed{\begin{array}{c}18?\\19?\\20?\end{array}}) \vdash (x, x)) \Leftarrow (19, \boxed{20})$$

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

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

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

$(x \mapsto 19) \oplus_{\text{conservative}} (x \mapsto 20)$ undefined

Enables a PutGet theorem

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

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

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

$(x \mapsto 19) \oplus_{\text{conservative}} (x \mapsto 20)$ undefined

Enables a PutGet theorem

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

$(x \mapsto 19) \oplus_{\text{optimistic}} (x \mapsto 20) = 19$ or 20 (or 18)

Enables many desirable interactions

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

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

$$((x \mapsto 18) \vdash (x, x)) \Rightarrow (18, 18)$$

$$\rightsquigarrow \quad \rightsquigarrow$$

$$((x \mapsto \boxed{\begin{array}{c}18?\\19?\\20?\end{array}}) \vdash (x, x)) \Leftarrow (19, 20)$$

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

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

$$(- \vdash \text{let } x = 18 \text{ in } (x, x)) \Rightarrow (18, 18)$$

$$\rightsquigarrow \quad \rightsquigarrow$$

$$(- \vdash \text{let } x = \boxed{\begin{array}{c} 18? \\ 19? \\ 20? \end{array}} \text{ in } (x, x)) \Leftarrow (19, 20)$$

Ask user to choose

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

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

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

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

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

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

$$\rightsquigarrow$$

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

$$(- \vdash \begin{array}{l}\text{let } f = \lambda x.\ x \mathbin{++} \text{"18"} \\ \text{in } f\ \text{"B "}\end{array}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$\text{"B}\boxed{\text{x 19}}\text{"}$$

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

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

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

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

$$((f \mapsto \lambda x.\ x \mathbin{++} \text{"18"}) \vdash f\ \text{"B "}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$\text{"Bx 19"}$$

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

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

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

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

$$((x \mapsto \text{"B "}) \vdash x \mathbin{++} \text{"18"}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$\text{"Bx 19"}$$

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

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

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

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

$$((x \mapsto \text{"B "}) \vdash x \mathbin{++} \text{"18"}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$((x \mapsto \text{"B}\boxed{x}\text{"}) \vdash x \mathbin{++} \text{"1}\boxed{9}\text{"}) \Leftarrow \text{"Bx 19"}$$

"data" "code"

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

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

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

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

$$((f \mapsto \lambda x.\ x \mathbin{++} \text{"18"}) \vdash f\ \text{"B "}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$((f \mapsto \lambda x.\ x \mathbin{++} \text{"1}\boxed{9}\text{"}) \vdash f\ \text{"B}\boxed{x}\text{"}) \Leftarrow \text{"Bx 19"}$$

"code" "data"

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

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

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

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

$$(- \vdash \begin{array}{l} \text{let } f = \lambda x.\ x \mathbin{++} \text{"18"} \\ \text{in } f\ \text{"B "} \end{array}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$(- \vdash \begin{array}{l} \text{let } f = \lambda x.\ x \mathbin{++} \text{"19"} \\ \text{in } f\ \text{"Bx "} \end{array}) \Leftarrow \text{"Bx 19"}$$

"code"

"data"

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

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

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

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

$$(- \vdash \text{"B "} \mathbin{++} \text{"18"}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$\text{"Bx 19"}$$

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

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

$$(- \vdash \text{"B "} ++ \text{"18"}) \Rightarrow \text{"B 18"}$$

$$\rightsquigarrow$$

$$(- \vdash \text{"Bx "} ++ \text{"19"}) \Leftarrow \text{"Bx 19"}$$

Primitive lens: $((++)_{get}, (++)_{put})$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \mid \ldots$$
$$\mid \boxed{\text{appLens}\ (e_{get}, e_{put})\ e_x}$$

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

Libraries and users can define custom lenses

$$(- \vdash \text{"B "} \mathbin{++} \text{"18"}) \Rightarrow \text{"B 18"}$$
$$\rightsquigarrow$$
$$(- \vdash \text{"Bx "} \mathbin{++} \text{"19"}) \Leftarrow \text{"Bx 19"}$$

Primitive lens: $((++)_{get}, (++)_{put})$

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \mid \ldots$$
$$\mid \text{appLens}\ (e_{get}, e_{put})\ e_x$$

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

Bidirectional Programs are Lenses

$$e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1\ e_2 \mid \ldots$$
$$\mid \ \mathbf{appLens}\ (e_{get}, e_{put})\ e_x$$

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

Bidirectional Programs *with* ~~are~~ Lenses

Example	LOC	Eval	#Upd	#Sol	Average Upd	
States Table A*	37	304±20	11	1.18	85±20	200×
States Table B*	126	774±70	7	1	331±80	700×
Recipe*	193	1455±80	17	1.05	1328±500	16×
Budgetting	37	328±11	7	2	9±2	80×
MVC	71	720±50	10	1	289±80	40×
Linked-Text	91	855±40	5	1.2	2025±200	5×
Markdown	128	1179±110	6	1	1607±200	13×
Dixit	130	705±40	15	1	417±1500	120×
Translation	122	357±20	8	2	415±200	50×
LaTeX in HTML	534	1648±200	6	1	943±1000	150×
Total / **Average**	*1469*	**833±400**	*92*	**1.18**	**(723±900)**	**(70×)**

Example	LOC	Eval	#Upd	#Sol	Average Upd	
States Table A*	37	304±20	11	1.18	85±20	200×
States Table B*	126	774±70	5	[illegible]	332±80	700×
Recipe*	193	1455±80	17	1.05	1328±500	16×
Budgetting	37	328±11	7	2	9±2	80×
MVC	71	720±50	10	1	289±80	40×
Linked-Text	91	855±40	5	1.2	2025±200	5×
Markdown	128	1179±110	6	1	1607±200	13×
Dixit	130	705±40	15	1	417±1500	120×
Translation	122	357±20	8	2	415±200	50×
LaTeX in HTML	534	1648±200	6	1	943±1000	150×
Total / **Average**	*1469*	**833±400**	*92*	**1.18**	**(723±900)**	**(70×)**

Variety of "bidirectional docs"

Example	LOC	Eval	#Upd	#Sol	Average Upd	
States Table A*	37	304±20	11	1.18	85±20	200×
States Table B*	12[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	700×
Recipe*	193	1455±80	17	1.05	1328±500	16×
Budgetting	37[illegible]	[illegible]	[illegible]	[illegible]	9±2	80×
MVC	71	720±50	10	1	289±80	40×
Linked-Text	91	855±40	5	1.2	2025±200	5×
Markdown	128	1179±110	6	1	1607±200	13×
Dixit	130	705±40	15	1	417±1500	120×
Translation	122	357±20	8	2	415±200	50×
LaTeX in HTML	534	1648±200	6	1	943±1000	150×
Total / **Average**	*1469*	**833±400**	*92*	**1.18**	**(723±900)**	**(70×)**

Variety of "bidirectional docs"

Variety of updates with relatively little ambiguity

Example	LOC	Eval	#Upd	#Sol	Average Upd	
States Table A*	37	304±20	11	1.18	85±20	200×
States Table B*	126	774±70	7	1	331±80	700×
Recipe*	193	1455±80	17	1.05	1328±500	16×
Budgetting	37	328±11	5	8	9±2	80×
MVC	71	720±50	10	1	289±80	40×
Linked-Text	91	855±40	5	1.2	2025±200	5×
Markdown	128	1179±110	6	1	1607±200	13×
Dixit	130	705±40	15	1	417±1500	120×
Translation	120	357±20	8	1	415±200	50×
LaTeX in HTML	534	1648±200	6	1	943±1000	150×
Total / **Average**	*1469*	**833±400**	*92*	**1.18**	**(723±900)**	**(70×)**

Variety of "bidirectional docs"

Variety of updates with relatively little ambiguity

Uneval time ≈ Eval time (with optimizations)

[>>> Skip Related Work >>>]

Bidirectional Evaluation

Related Work

Lenses [Foster et al. 2007, et al.]

BX DSLs for relational data, trees, strings, graphs

Point-free programming style

Applicative and Monadic Lenses

[Matsuda and Wang, ICFP 2015]
[Xia, Orchard, and Wang, ESOP 2019]

Bidirectional Evaluation
Related Work

Func. Program Slicing [Perera et al., ICFP 2012]

Forward evaluator records full trace for output value

Backward evaluator slices exp. based on "don't-care" sub-values

HOBiT [Matsuda and Wang, ESOP 2018]

Staged evaluator for λ-calculus 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

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

```
length : List a -> Int
length =
  ??

assert (length [2,0,1,6] == 4)
assert (length       [] == 0)
```

Program Synthesizer

```
length xs =
  case xs of
    []     -> 0
    x::xs' -> 1 + length xs'
```

```
stutter : List a -> List a
stutter =
  ??

assert (stutter [1,2,3] == [1,1,2,2,3,3])
assert (stutter      [] == []           )
```

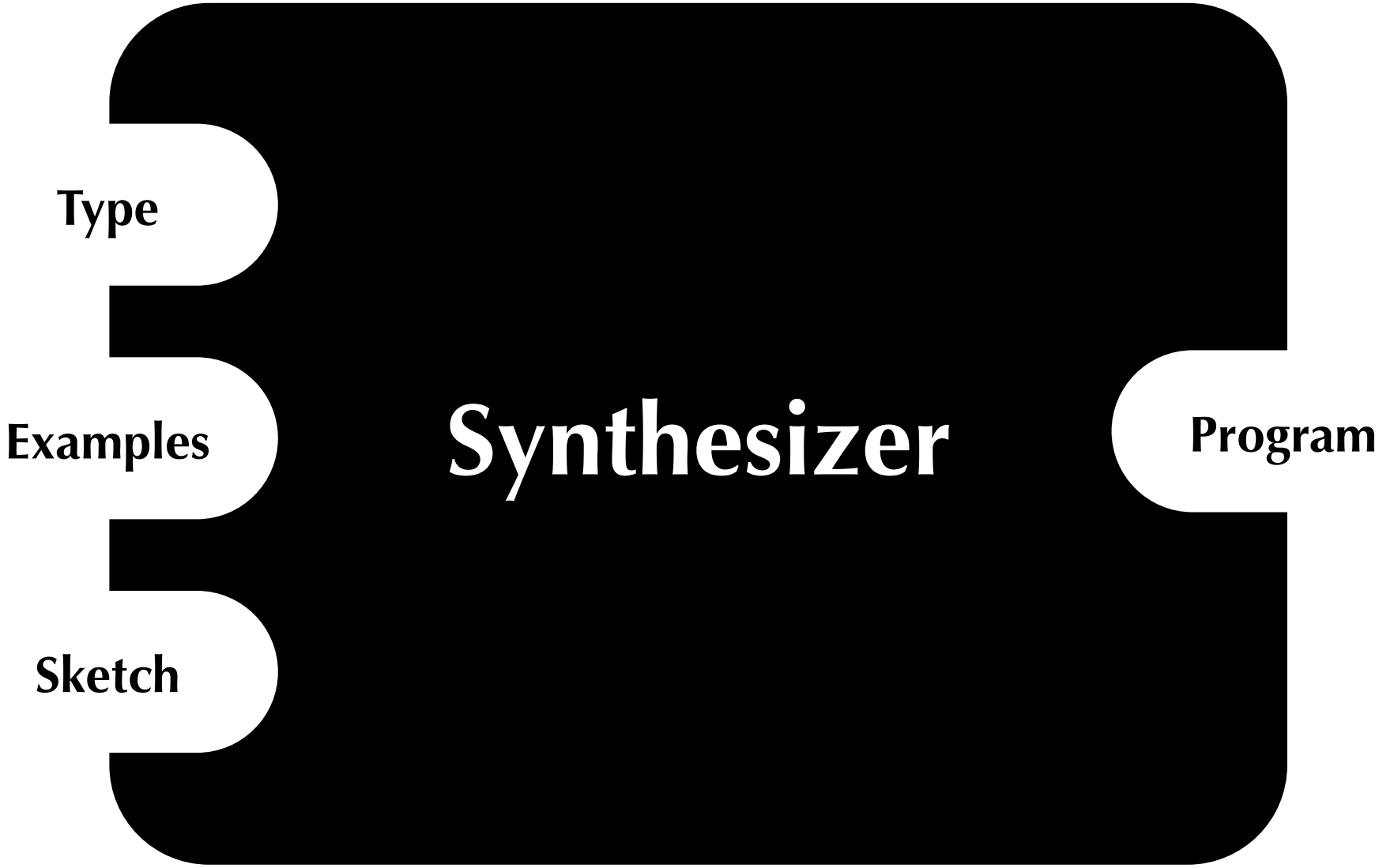
Program Synthesizer

```
stutter xs =
  case xs of
    []     -> []
    x::xs' -> x :: x :: stutter xs'
```

```
max : Nat -> Nat -> Nat
max n m =
  case n of
    Zero    -> m
    Succ n' -> ??

assert (max 1 1 == 1)
assert (max 1 2 == 2)
assert (max 2 0 == 2)
assert (max 2 1 == 2)
assert (max 2 2 == 2)
```

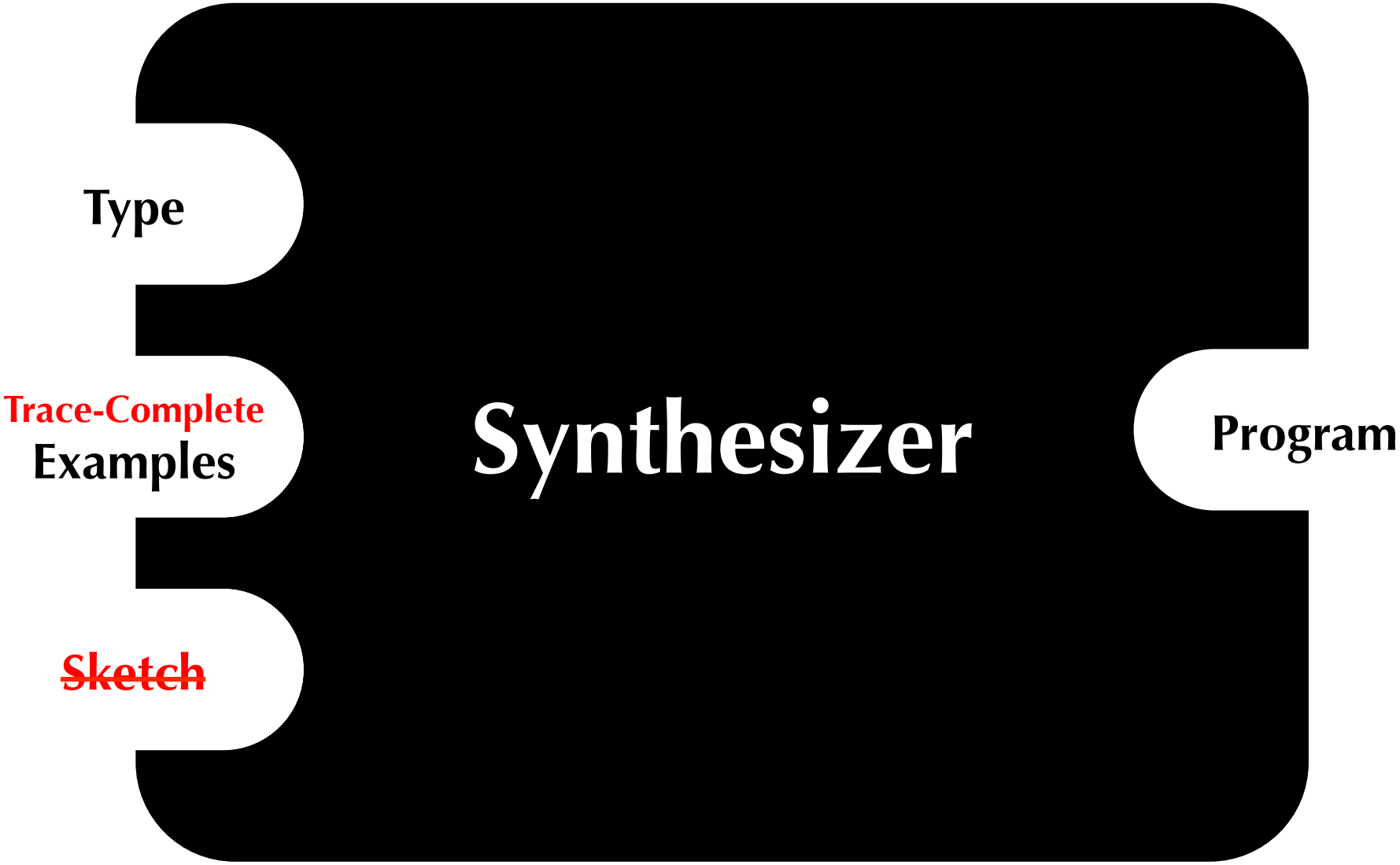
Program Synthesizer

```
max n m =
  case n of
    Zero    -> m
    Succ n' -> case m of
                 Zero    -> n
                 Succ m' -> Succ (max n' m')
```


Type
Examples
Sketch
Synthesizer
Program

Sketch, Rosette, … Leon, Synquid, … FlashFill, PROSE, …

Myth [Osera et al. PLDI 2015, POPL 2016]



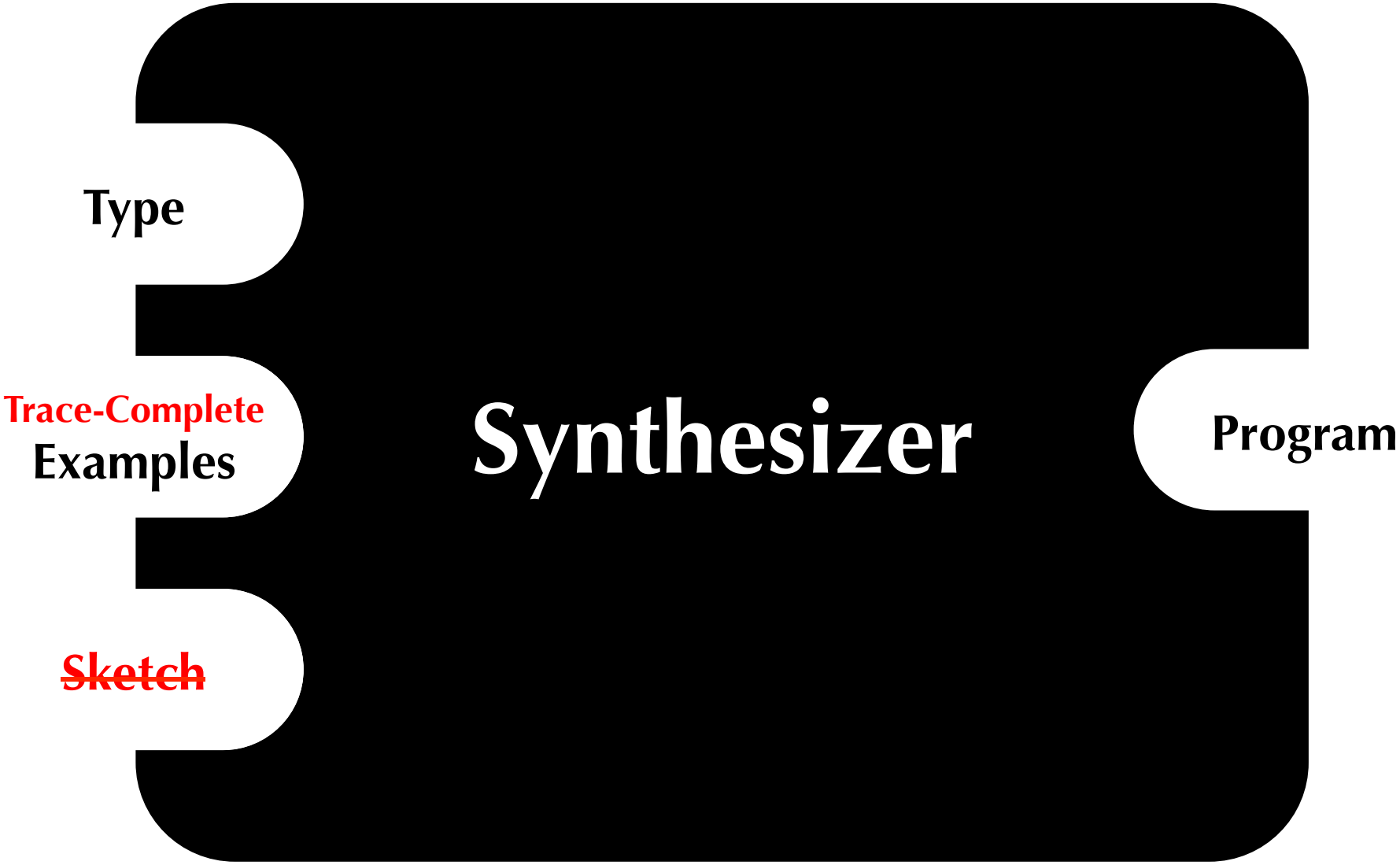
Trace-Complete
Examples

```
length : List a -> Int
length =
  ??

assert (length [2,0,1,6] == 4)
assert (length        [] == 0)
```

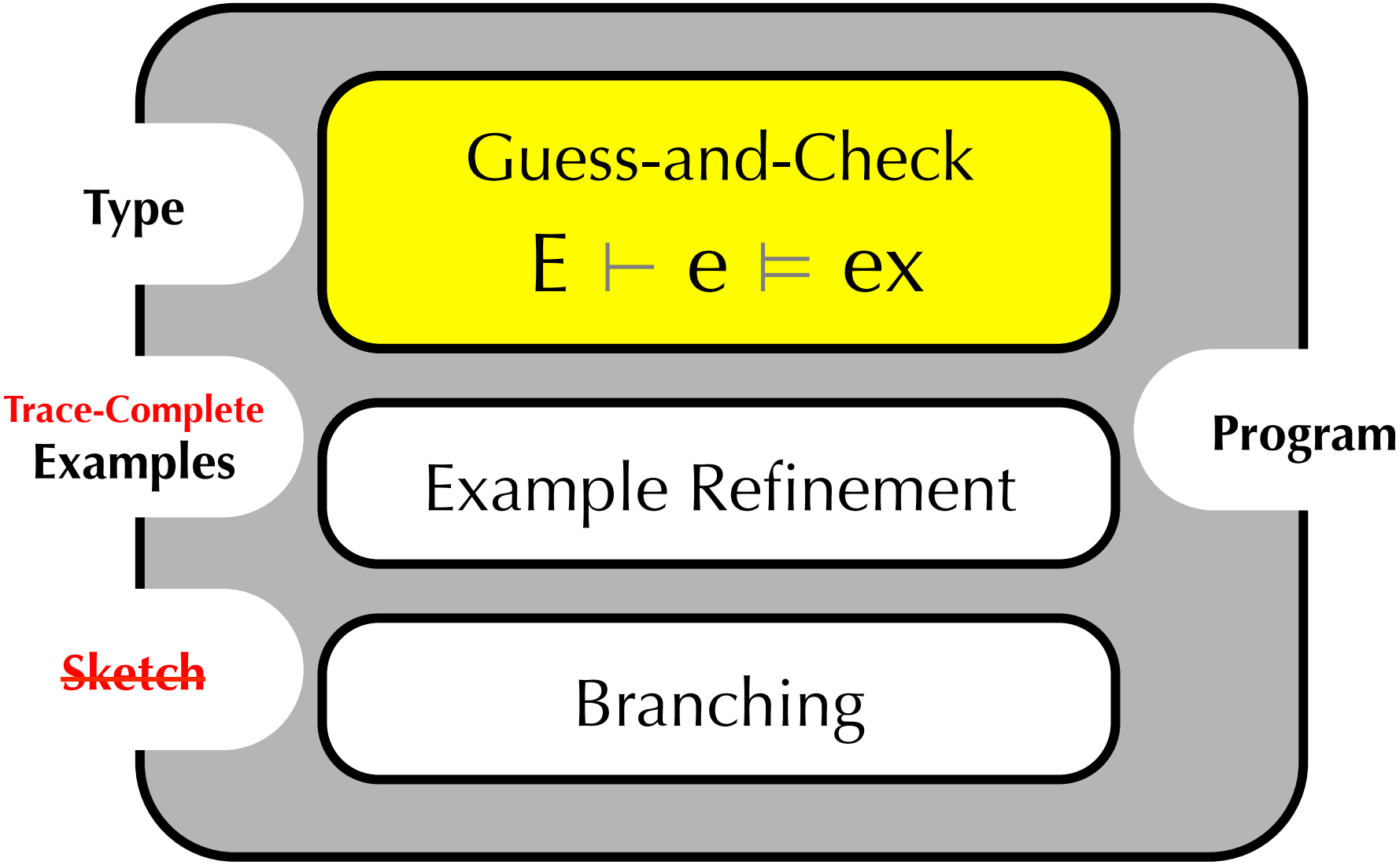
Trace-Complete
Examples

```
length : List a -> Int
length =
  ??

assert (length [2,0,1,6] == 4)
assert (length   [0,1,6] == 3)
assert (length     [1,6] == 2)
assert (length       [6] == 1)
assert (length        [] == 0)
```

Myth [Osera et al. PLDI 2015, POPL 2016]



Myth [Osera et al. PLDI 2015, POPL 2016]



Example Satisfaction in Myth

$$E \vdash e \vDash ex$$

Eval

$$E \vdash e \Rightarrow v$$

$+$

Check

$$v \vDash ex$$

$$\mathbf{v} ::= \ldots \mid \lambda x.e$$

$$\mathbf{ex} ::= \ldots \mid \{ \mathbf{v_{in}} \rightarrow \mathbf{ex_{out}} \}$$

Input-output examples for functions

$$E \vdash e \models ex$$

How to **Eval** and **Check** Sketches ??

Eval

$$E \vdash e \Rightarrow v$$

$+$

Check

$$v \models ex$$

$$E \vdash e \vDash ex$$

How to **Eval** and **Check** Sketches ??

Live Eval

$$E \vdash e \Rightarrow r$$

+

Check

$$v \vDash ex$$

Partially evaluate sketch to result r (value or "paused")

$$E \vdash e \models ex$$

How to **Eval** and **Check** Sketches ??

Live Eval

$$E \vdash e \Rightarrow r$$

+

Live Uneval

$$r \Leftarrow ex \dashv X$$

Partially evaluate sketch to result r (value or "paused")

Constraints X s.t. r will resume and satisfy ex

$$E \vdash e \Rightarrow r \Leftarrow ex \dashv X$$

Live Bidirectional Evaluation

Live Eval

$$E \vdash e \Rightarrow r$$

+

Live Uneval

$$r \Leftarrow ex \dashv X$$

Partially evaluate sketch to result r (value or "paused")

Constraints X s.t. r will resume and satisfy ex

>>> Skip Examples >>>

Live Eval

$$E \vdash e \Rightarrow r$$

Partially evaluate sketch to result r (value or "paused")

Eval

$e \Rightarrow v$

Expressions

Values **Errors**

2019 + 1 ⇒ **2020**

2019 ++ "!" ⇒ **crash**

(??, 2019 + 1) ⇒ **crash**

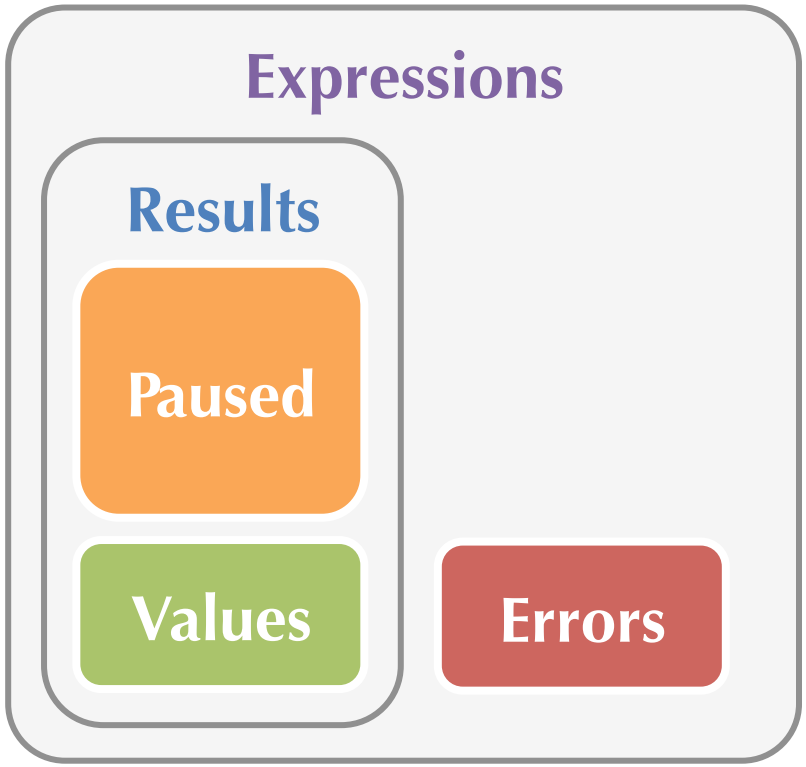
Holes typically represented as

`raise "Not yet implemented"`

Live Eval

$$e \Rightarrow r$$

Partially evaluate "around" holes



(??, 2019 + 1) ⇒ (??, 2020)

snd (??, 2019 + 1) ⇒ 2020

(snd ??, 2019 + 1) ⇒ (snd ??, 2020)

Live Eval

$$e \Rightarrow r$$

$$r ::= \ldots \mid \lambda x.e \mid [E]\ ??_h \mid \text{snd } r \mid r_1 + r_2 \mid \ldots$$

Values

Paused Expressions

Live Eval

$$e \Rightarrow r$$

Live Uneval

$$r \Leftarrow ex \dashv X$$

$$r ::= \ldots \mid \lambda x.e \mid [E]\ ??_h \mid \mathbf{snd}\ r \mid r_1 + r_2 \mid \ldots$$

$$ex ::= \ldots \mid \{\, v_{in} \rightarrow ex_{out} \,\}$$

$$X ::= \ldots \mid X, (E \vdash ??_h \vDash ex)$$

Example constraints required by Myth

Live Uneval

$$r \Leftarrow ex \dashv X$$

$$
\begin{array}{rcl}
r & ::= & \ldots \mid \lambda x.e \mid [E]\ ??_h \mid \mathbf{snd}\ r \mid r_1 + r_2 \mid \ldots \\
ex & ::= & \ldots \mid \{\, v_{in} \rightarrow ex_{out} \,\} \\
X & ::= & \ldots \mid X, (E \vdash ??_h \vDash ex)
\end{array}
$$

$$
\begin{array}{rcl}
2020 & \Leftarrow & 2020 \dashv \emptyset \\
[E]\ ??_1 & \Leftarrow & 2020 \dashv (E \vdash ??_1 \vDash 2020) \\
4 + [E]\ ??_2 & \Leftarrow & 2020 \dashv (E \vdash ??_2 \vDash 2016) \\
\mathbf{ite}\ ([E]\ ??_3)\ 2016\ 2020 & \Leftarrow & 2020 \dashv (E \vdash ??_3 \vDash \mathbf{False})
\end{array}
$$

Live Eval

$$e \Rightarrow r$$

Live Uneval

$$r \Leftarrow ex \dashv X$$

$$
\begin{array}{rcl}
r & ::= & \ldots \mid \lambda x.e \mid [E]\ ??_h \mid \mathbf{snd}\ r \mid r_1 + r_2 \mid \ldots \\
ex & ::= & \ldots \mid \{\, v_{in} \rightarrow ex_{out} \,\} \\
X & ::= & \ldots \mid X, (E \vdash ??_h \vDash ex)
\end{array}
$$

$$E \vdash e \Rightarrow r \Leftarrow ex \dashv X$$

Live Bidirectional Evaluation

Live Eval

$$E \vdash e \Rightarrow r$$

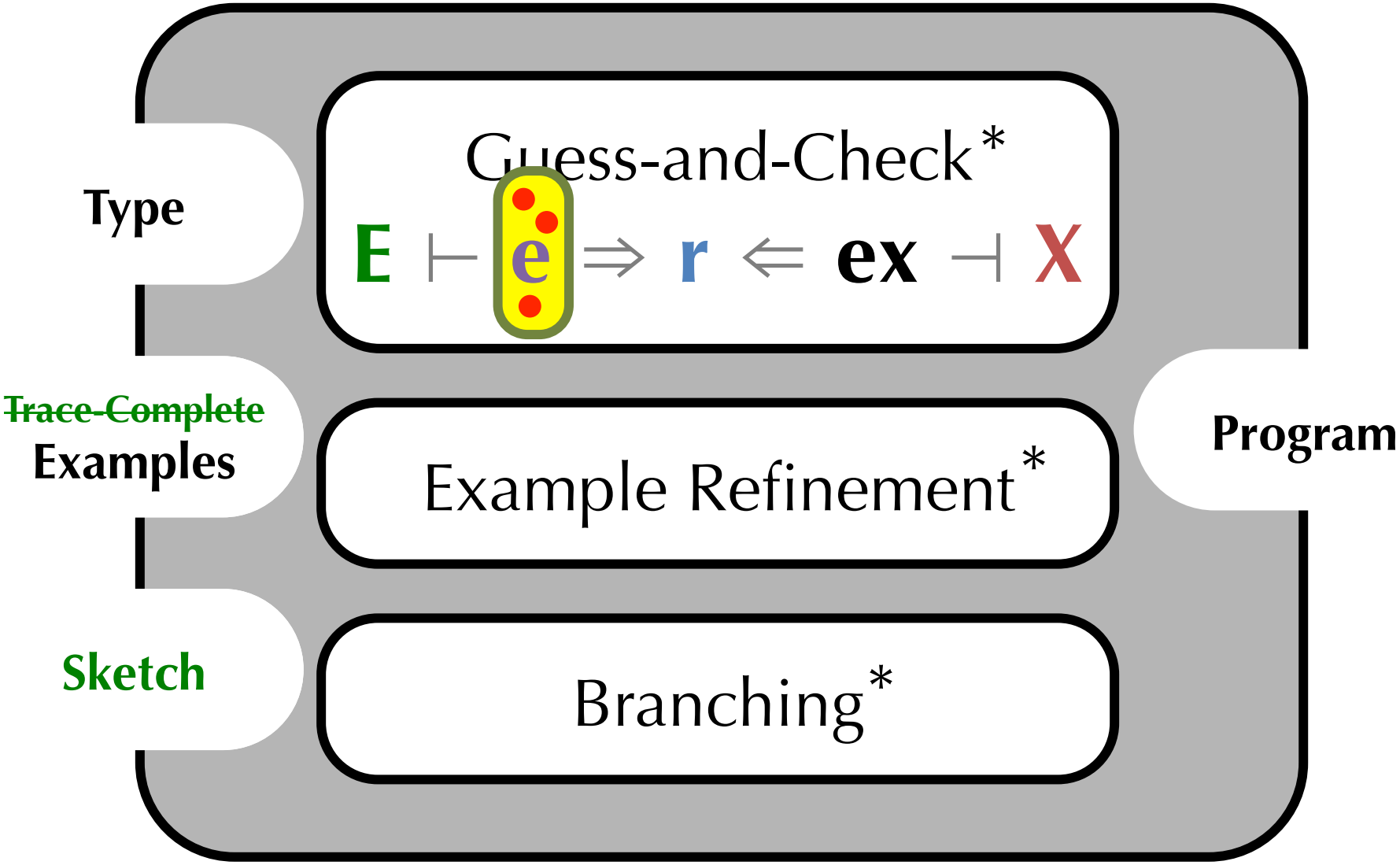
+

Live Uneval

$$r \Leftarrow ex \dashv X$$

Partially evaluate sketch to result r (value or "paused")

Constraints X s.t. r will resume and satisfy ex


Type
Guess-and-Check*
E ⊢ e ⇒ r ⇐ ex ⊣ X
~~Trace-Complete~~
Examples
Example Refinement*
Program
Sketch
Branching*

	Smyth					
Experiment	**1**		**2a**	**2b**	**3a**	**3b**
Sketch / Objective	*None / Top-1*		*None / Top-1*		*Base Case / Top-1-R*	
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]
list_nth	13	0.124	5 (38%)	(7,14)	1+2 (23%)	(1+7,1+15)
list_pairwise_swap	7	0.634	5 (71%)	timeout	overspec	timeout
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[3]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)
Averages			61%*		46%	

	Smyth					
Experiment	**1**		**2a**	**2b**	**3a**	**3b**
Sketch / Objective	*None / Top-1*		*None / Top-1*		*Base Case / Top-1-R*	
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]
list_nth	13	0.12	3 (38%)	timeout	1+2 (23%)	timeout
list_pairwise_swap	7	0.834	5 (71%)	timeout	overspec	timeout
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[3]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)
Averages			61%*		46%	

Compared to prior state-of-the-art:

Trace-complete examples (without a sketch)

	Smyth					
Experiment	**1**		**2a**	**2b**	**3a**	**3b**
Sketch / Objective	*None / Top-1*		*None / Top-1*		*Base Case / Top-1-R*	
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]
list_nth	13	0.124	5 (38%)	(7,14)	1+2 (23%)	(1+7,1+15)
list_pairwise_swap	7	0.634	5 (71%)	timeout	overspec	timeout
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[3]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)
Averages			61%*		46%	

Compared to prior state-of-the-art:

Non-trace-complete examples reduce specification size

	Smyth					
Experiment	**1**		**2a**	**2b**	**3a**	**3b**
Sketch / Objective	*None / Top-1*		*None / Top-1*		*[illegible] / Top-1-R*	
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]
list_nth	13	0.124	5 (38%)	(7,14)	1+2 (23%)	(1+7,1+15)
list_pairwise_swap	7	0.634	5 (71%)	timeout	overspec	timeout
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[2]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)
Averages			61%*		46%	

Compared to prior state-of-the-art:

Non-trace-complete examples reduce specification size

Sketches reduce specification size

	Smyth					
Experiment	**1**		**2a**	**2b**	**3a**	**3b**
Sketch / Objective	*None / Top-1*		*None*		*Base Co*	
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]
list_nth	13	0.124	5 (38%)	(7,14)	1+2 (23%)	(1+7,1+15)
list_pairwise_swap	7	0.634	5 (71%)	timeout	overspec	timeout
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[3]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)
Averages			61%*		46%	

Compared to prior state-of-the-art:

Non-trace-complete examples reduce specification size

Sketches reduce specification size

Synthesis (relatively) ***robust*** to choice of examples

	Smyth					
Experiment	**1**		**2a**	**2b**	**3a**	**3b**
Sketch / Objective	*None / Top-1*		*None / Top-1*		*Base Case / Top-1-R*	
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]
list_nth	13	0.124	5 (38%)	(7,14)	1+2 (23%)	(1+7,1+15)
list_pairwise_swap	7	0.634	5 (71%)	timeout	overspec	timeout
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[3]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)
Averages			61%*		46%	

Compared to prior state-of-the-art:

Non-trace-complete examples reduce specification size

Sketches reduce specification size

Synthesis (relatively) ***robust*** to choice of examples

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

PL+UI

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

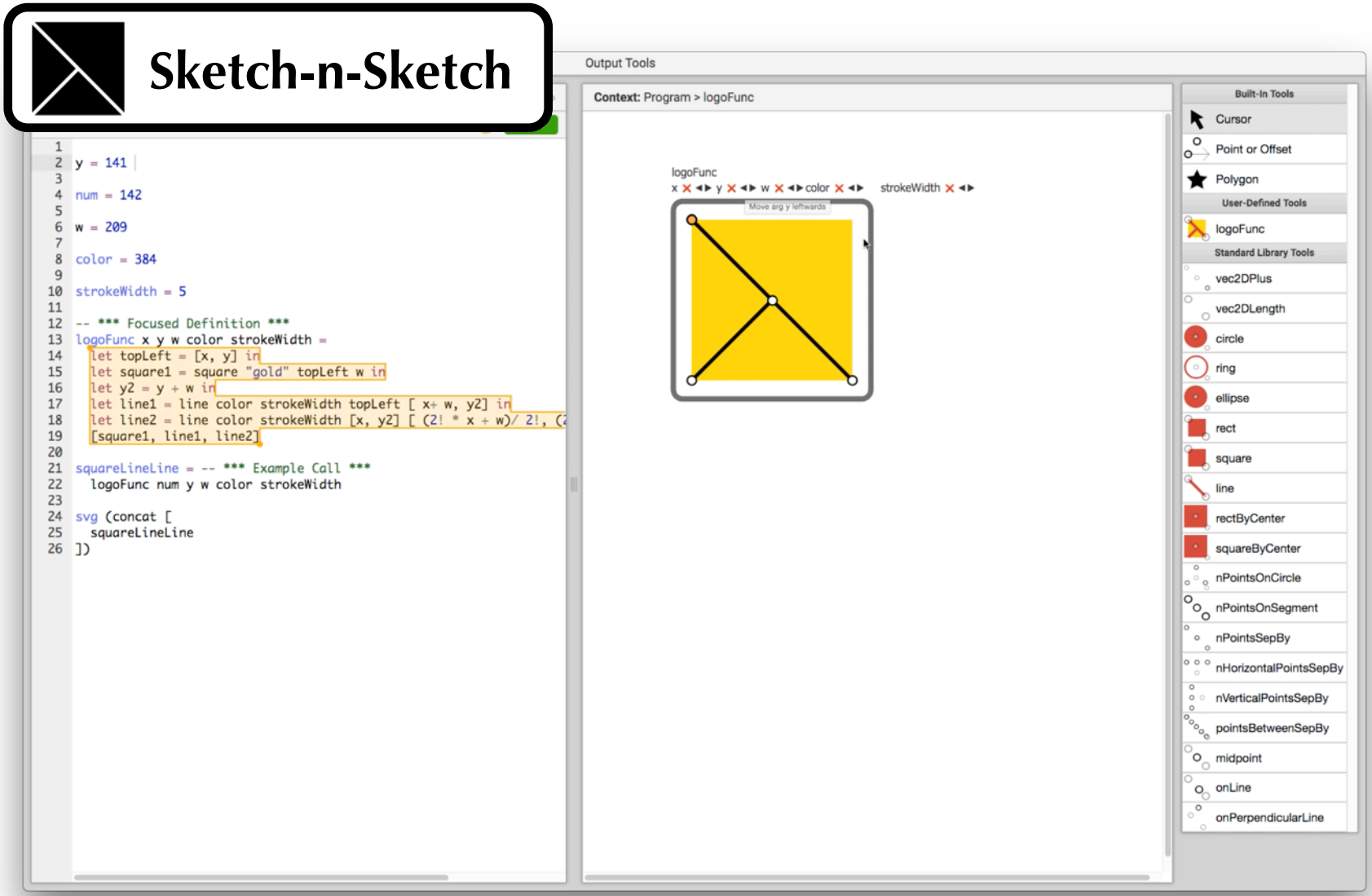
Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

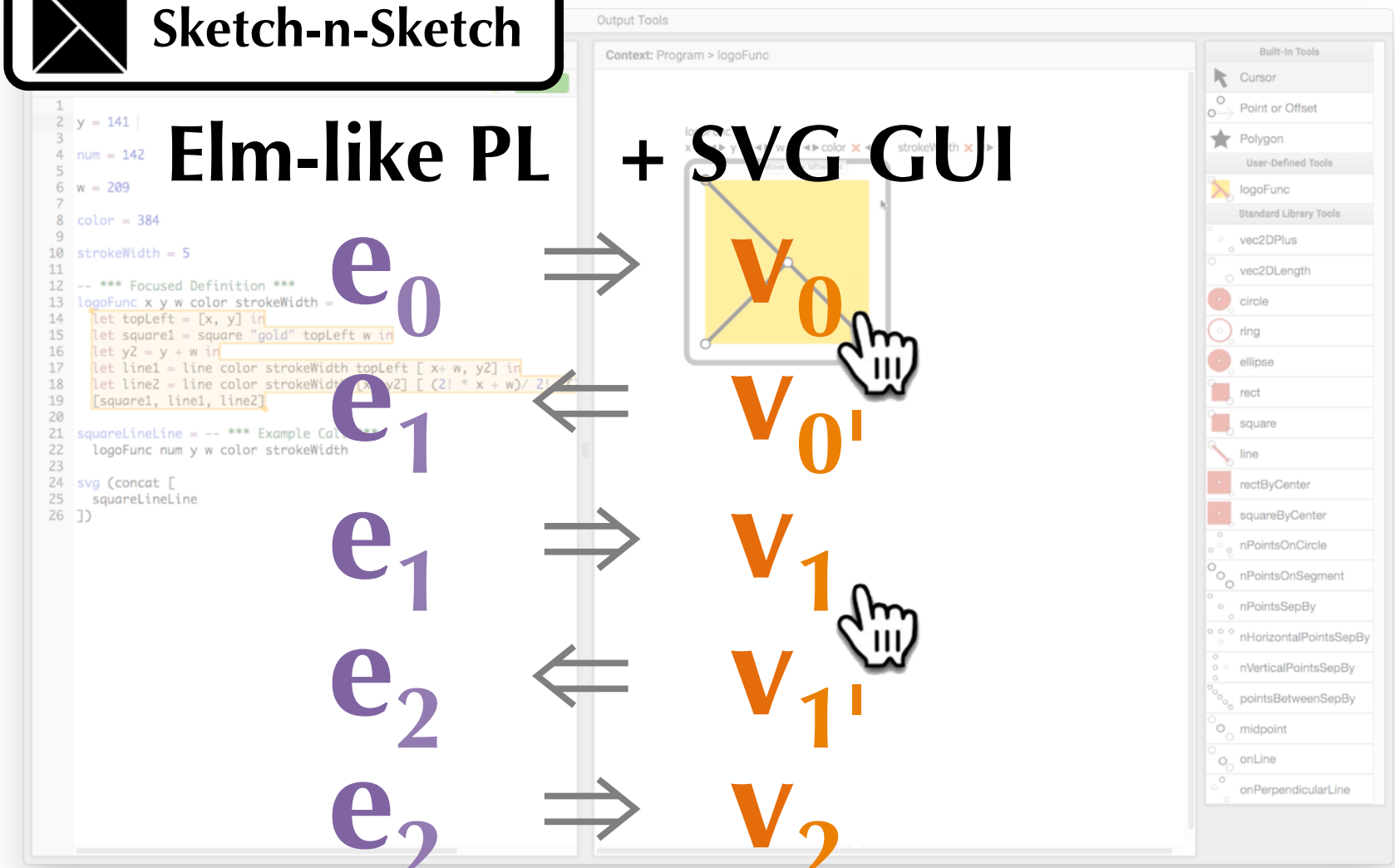
Output-Directed Programming



Output-Directed Programming

Sketch-n-Sketch

Elm-like PL + SVG GUI



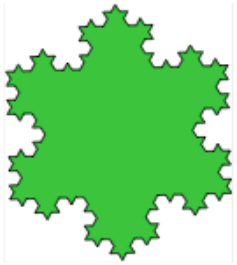
$e_0 \Rightarrow v_0$

$e_1 \Leftarrow v_{0'}$

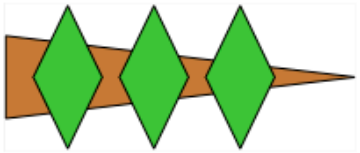
$e_1 \Rightarrow v_1$

$e_2 \Leftarrow v_{1'}$

$e_2 \Rightarrow v_2$



(i) Koch Snowflake ★
31 LOC



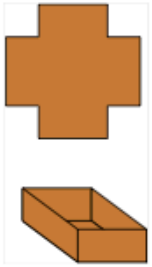
(ii) Tree Branch ★
31 LOC

(iii) Target ★
8 LOC

(iv) Precision Floor Plan ☆
9 LOC, 3 MO
(vs. 25 LOC, 27 MO)

(v) Mondrian Arch λ
15 LOC, 4 MO
(vs. 23 LOC, 9 MO)

(vi) Balance Scale ☆
34 LOC, 21 MO
(vs. 53 LOC, 40 MO)



(vii) Box Volume ★
81 LOC

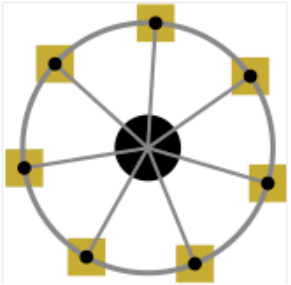
(viii) Xs λ★
44 LOC

(ix) Battery λ
13 LOC, 5 MO
(vs. 21 LOC, 2 MO)

(x) Ladder ★
18 LOC

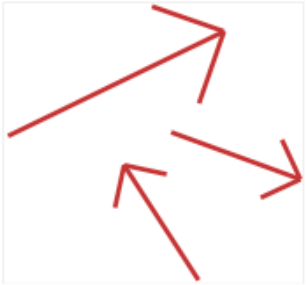
(xi) Logo (via Three Tris) λ★
40 LOC

(xii) N Boxes ★
7 LOC



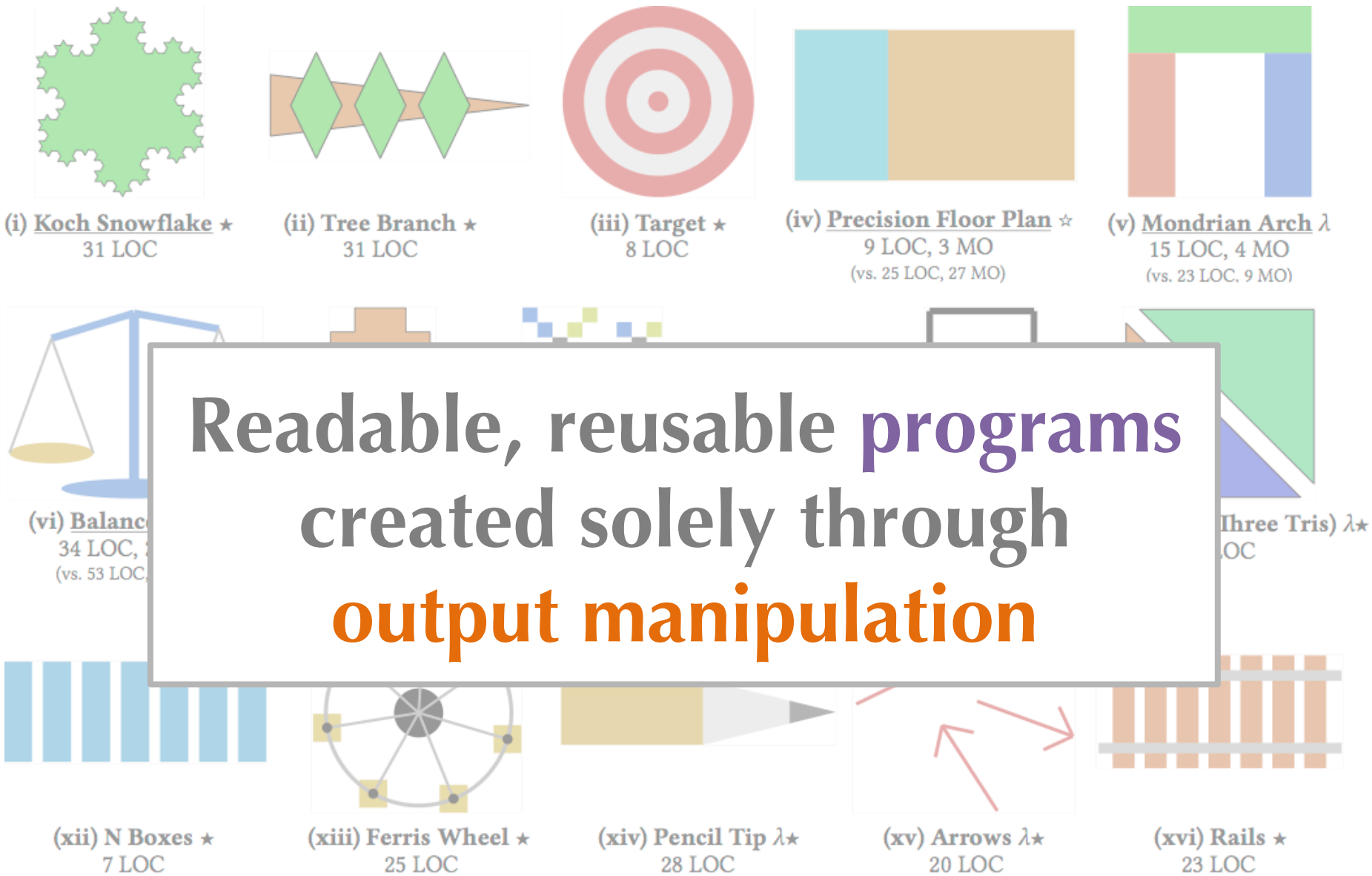
(xiii) Ferris Wheel ★
25 LOC

(xiv) Pencil Tip λ★
28 LOC



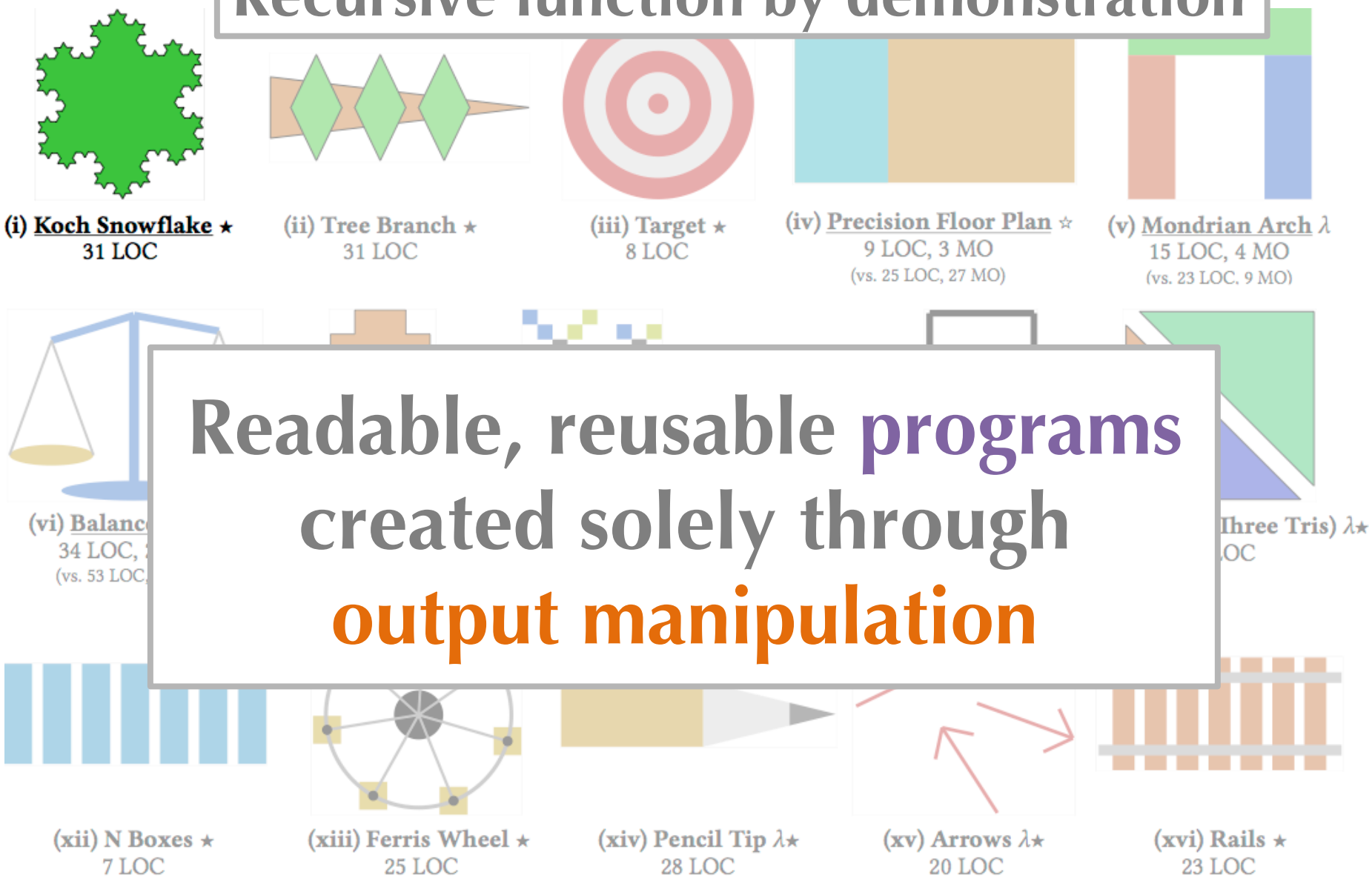
(xv) Arrows λ★
20 LOC

(xvi) Rails ★
23 LOC



Readable, reusable programs created solely through output manipulation

Recursive function by demonstration



Readable, reusable programs created solely through output manipulation

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

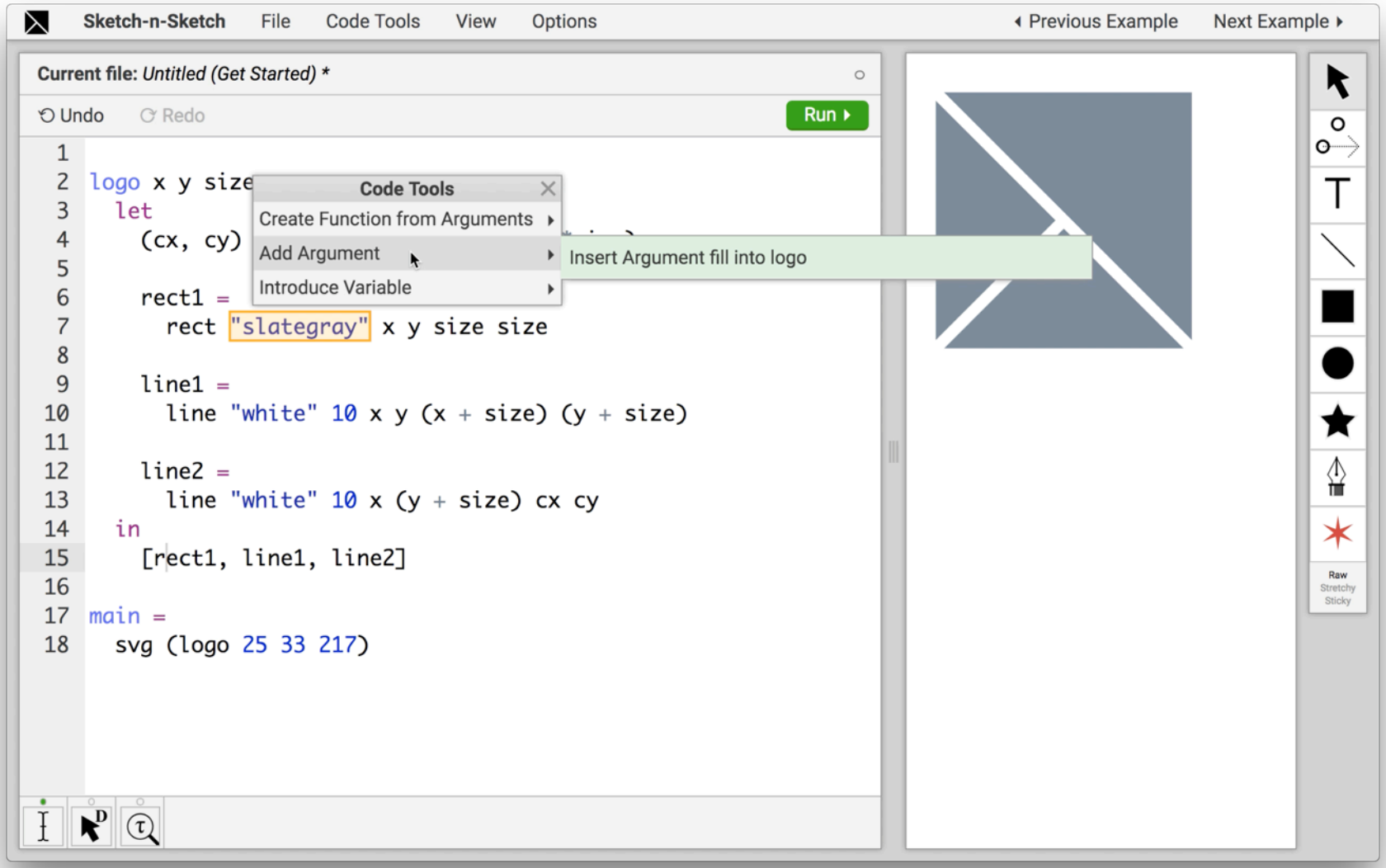
[ICSE 2018, VL/HCC 2020]

>>> Skip >>>

Output-Directed Programming

[UIST 2016, UIST 2019]

Hybrid Text-GUI Code Editors

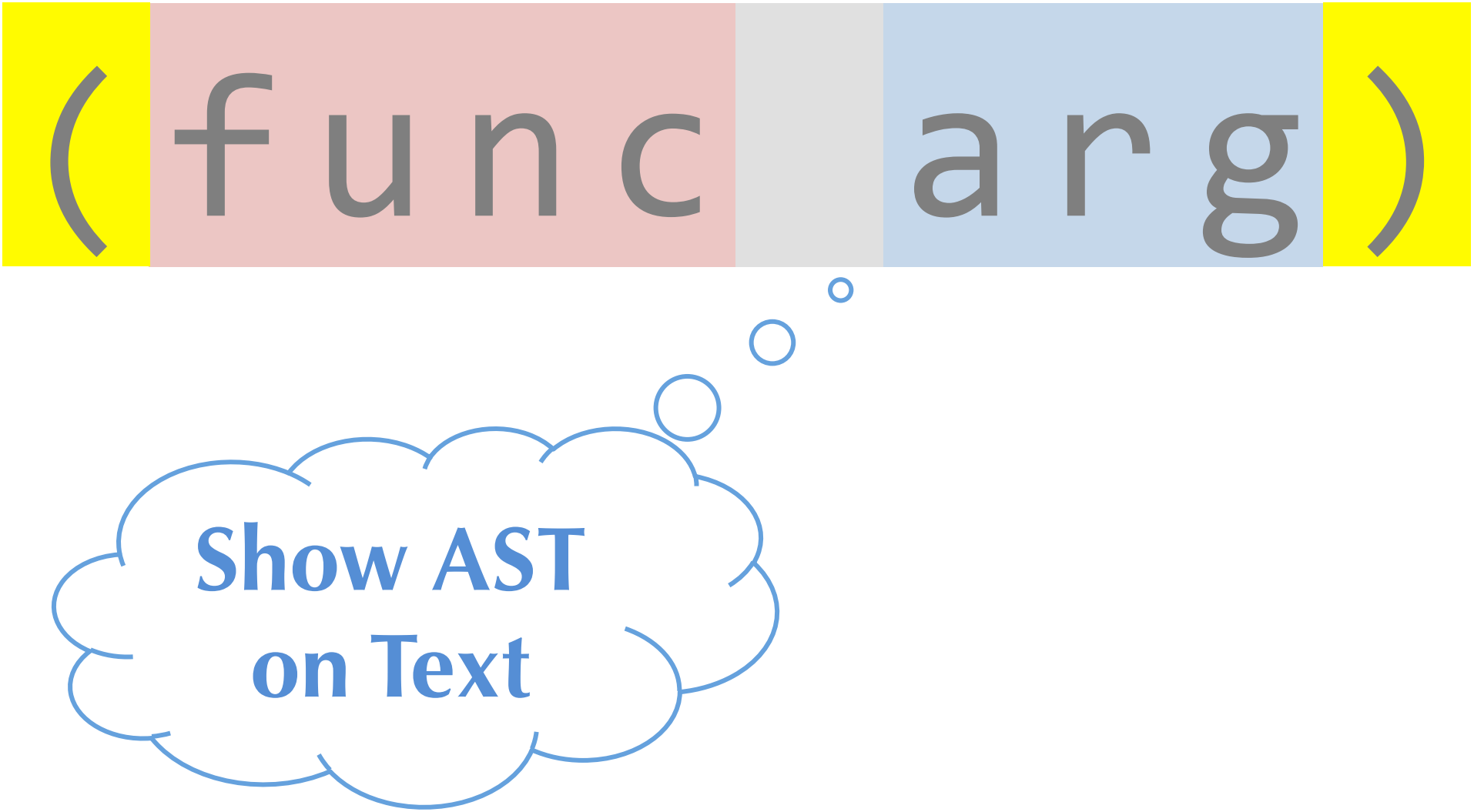


[>>> Skip >>>]

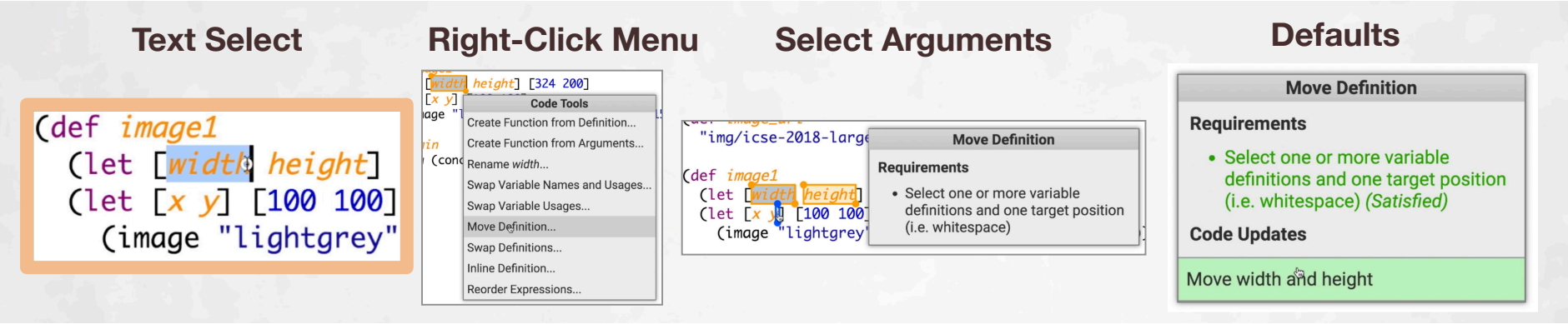
```
(func arg)
```

(	f	u	n	c		a	r	g	)

Show AST on Text

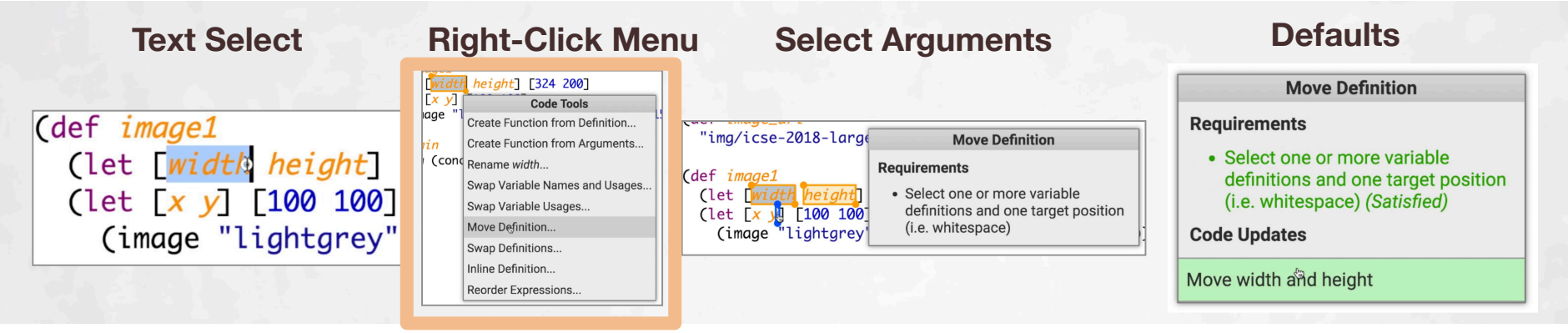

(func arg)
Show AST
on Text

Traditional Refactoring UI



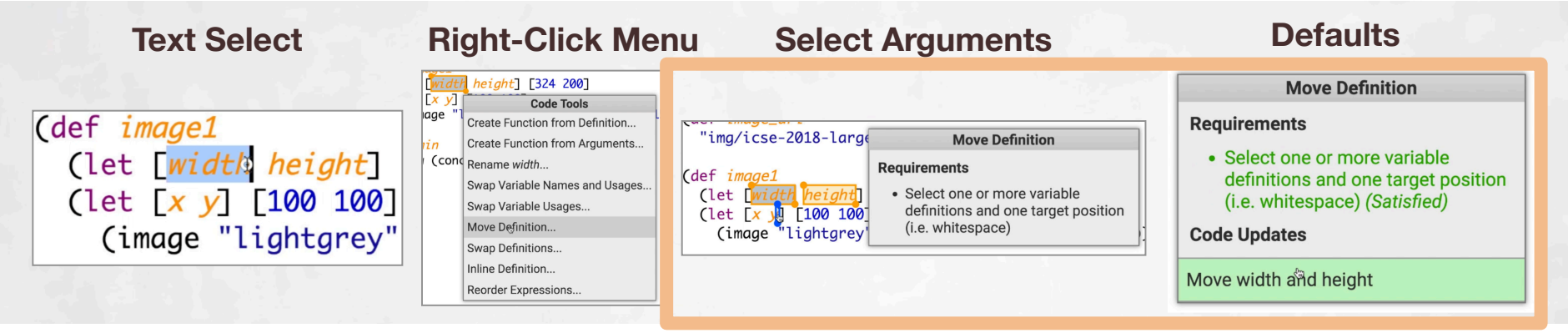
Text-Select

Traditional Refactoring UI



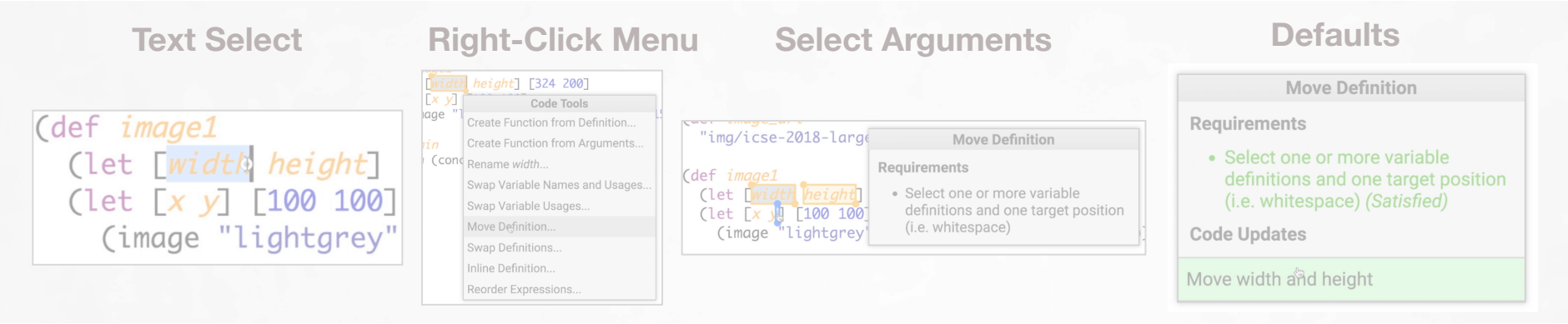
Text-Select + Long Menu

Traditional Refactoring UI



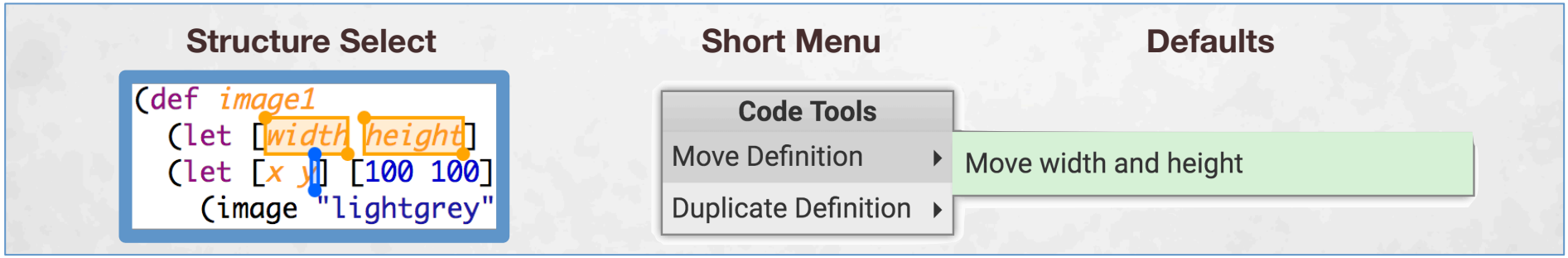
Text-Select + Long Menu + Configuration Choices

Traditional Refactoring UI



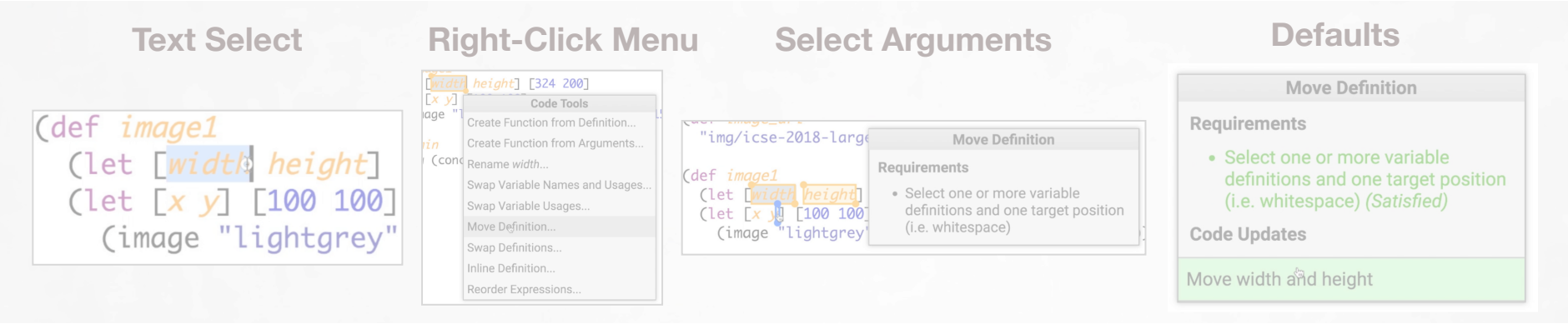
Text-Select + Long Menu + Configuration Choices

Deuce Refactoring UI



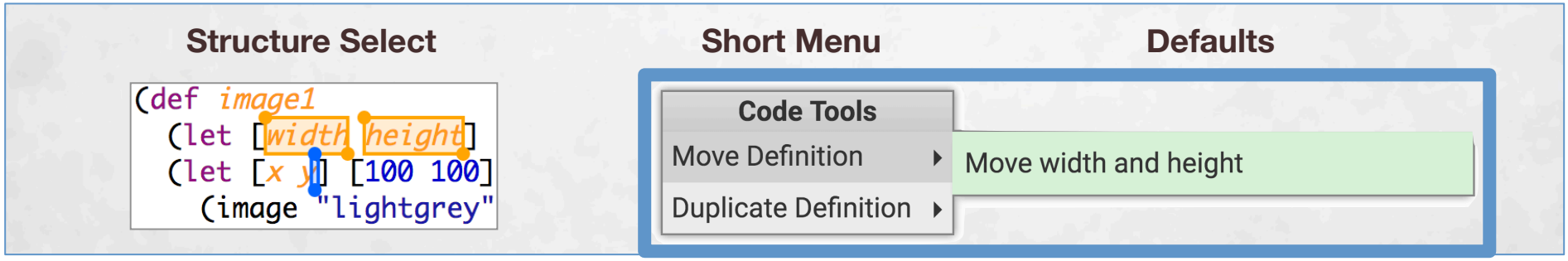
Structural Multi-Select

Traditional Refactoring UI

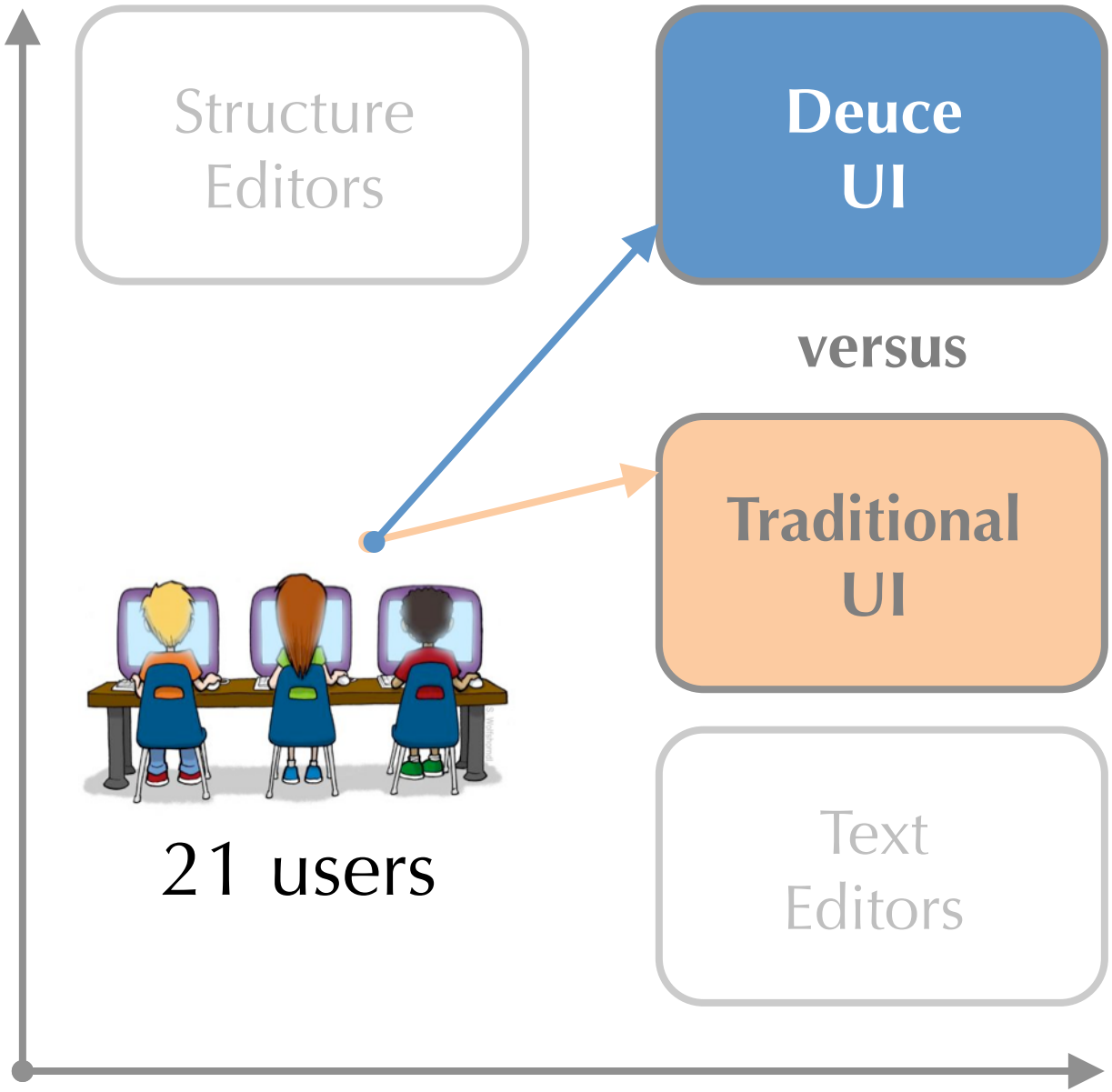


Text-Select + Long Menu + Configuration Choices

Deuce Refactoring UI



Structural Multi-Select + Context-Sensitive Transforms


Structure Editors
Deuce UI
versus
Traditional UI
21 users
Text Editors

Deuce UI versus **Traditional UI**

Traditional may be better for discovery

Deuce may be faster once learned

Deuce strongly preferred

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

PL + UI

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

>>> Skip 4 >>>

Bidirectional Evaluation

Future Work

Synthesis of larger structural repairs

More expressive value diffs

Automatically deriving unevaluators

Bi-directional, scriptable
Docs, Sheets, Slides, p5.js, D3, Vega, etc.

Live Programming by Example

Future Work

Unpredictability of examples

Synthesizing pattern and type holes

Integrating logic-based specifications

Propagating multiple candidate programs ("multi-sketches")

Output-Directed Programming

Future Work

Propagating multiple candidate programs ("multi-sketches")

Visualization of intermediate program states

Deriving structure editors from `toString` and `toSvg` functions

DSLs for defining new transformations

Hybrid Text-GUI Code Editors

Future Work

DSLs for defining new transformations

DSLs for defining custom, type-specific GUIs

UI concerns for larger programs

How to encourage tree transforms?

Usable Synthesis

Future Work

Explaining program differences

Statistics to rank solutions according to code corpus and edit history

Representation of search space and intermediate synthesis results

Interaction to resolve ambiguity

User Interfaces for Programming

Psychology of Programming Interest Group

LIVE 2020 · Workshop on Live Programming

LIVE 2020 | LIVE 2019 | LIVE 2018 | LIVE 2017 | LIVE 2016 | LIVE 2013

PLATEAU2020

11th annual workshop on the intersection of HCI and PL

Online Edition! November 2020

Co-located with CHICAGO SPLASH 2020

CHICAGO SPLASH2020

Sun 15 - Fri 20 November 2020 Online Conference

CHICAGO SPLASH 2020

Attending Tracks Organization Search Series Sign in Sign up

SPLASH 2020 (series) / HATRA 2020 (series) /

Human Aspects of Types and Reasoning Assistants

HATRA 2020

SHONAN MEETING

Seminars

NO.147

Self-supporting, Extensible Programming Languages and Environments for Exploratory, Live Software Development

‹Programming› 2020

User Interfaces for Programming

UIST 2016

Session 5-A: Viz Auditorium
Chair: Fanny Chealier

Eviza: A Natural Language Interface for Visual Analysis
Vidya Setlur, Sarah E Battersby, Melanie K Tory, Rich Gossweiler, Angel X Chang

Semi-Automated SVG Programming via Direct Manipulation
Brian Hempel, Ravi Chugh

Reading and Learning Smartfonts
Danielle Bragg, Shiri Azenkot, Adam Kalai

Interactive Volume Segmentation with Threshold Field Painting
Takeo Igarashi, Naoyuki Shono, Taichi Kin, Toki Saito

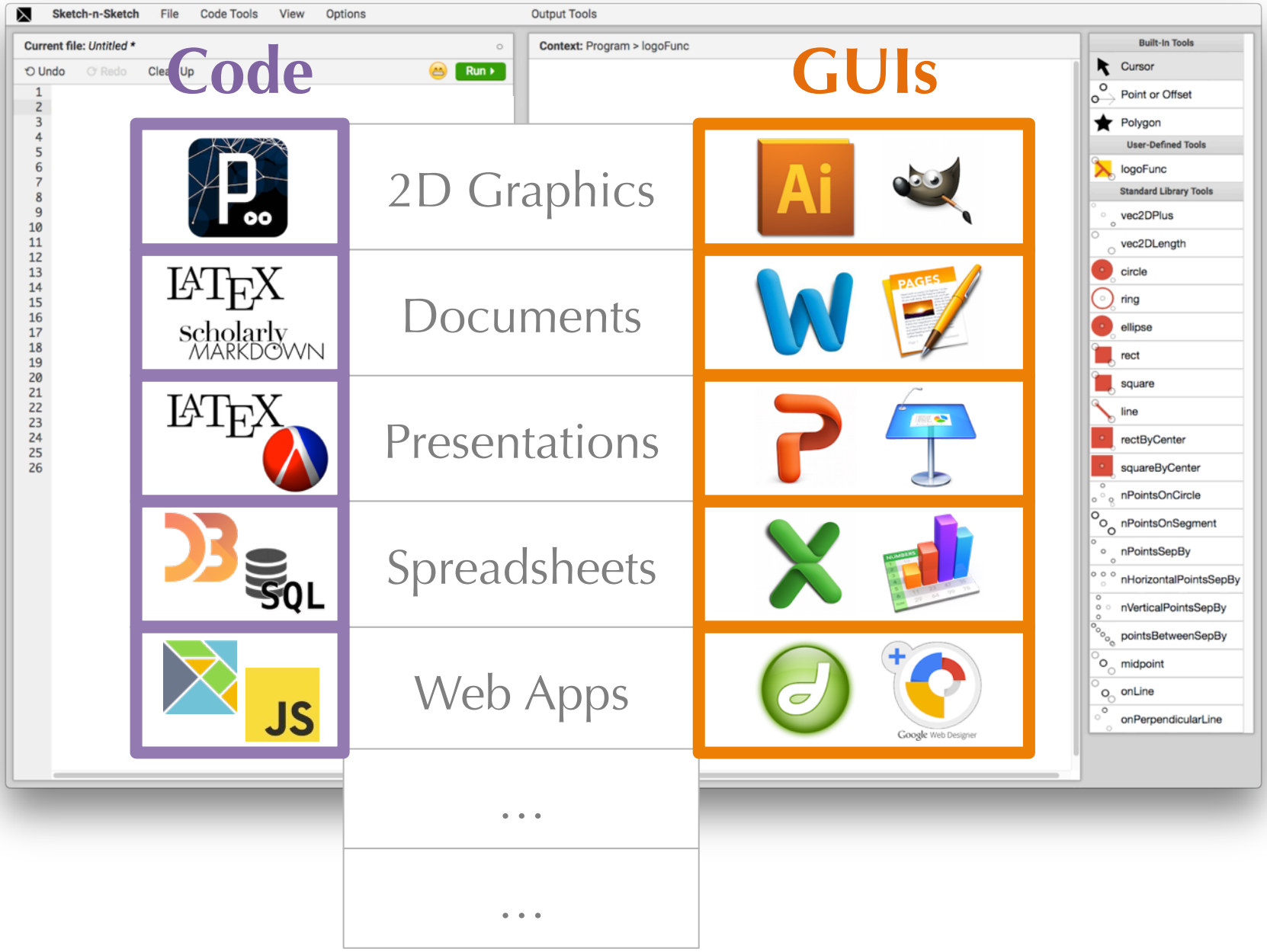
UIST 2020

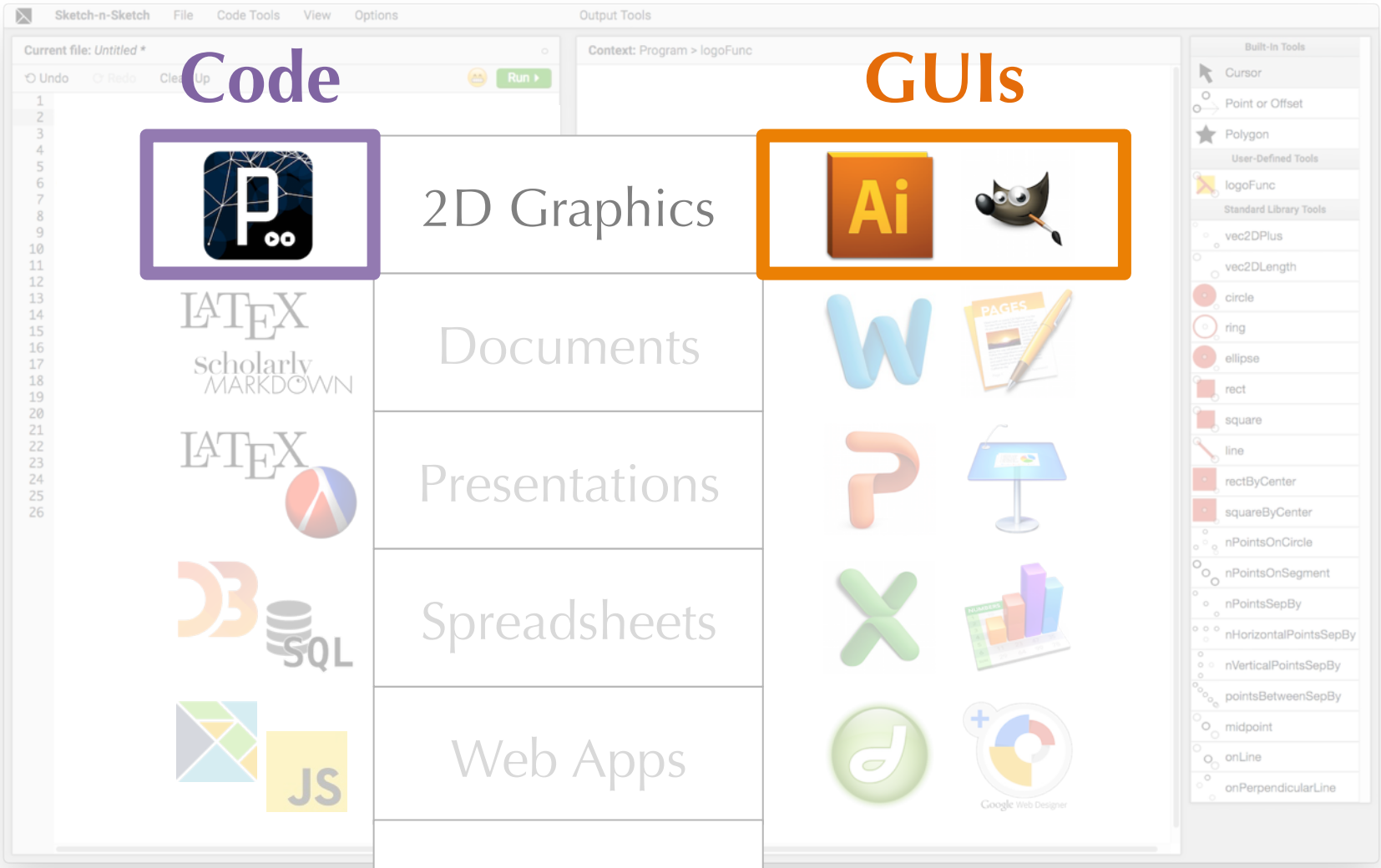
Session 7B: Live Programming and Synthesis

Focused Live Programming with Loop Seeds
Sorin Lerner

Small-Step Live Programming by Example
Kasra Ferdowsifard, Allen Ordookhanians, Hila Peleg, Sorin Lerner, Nadia Polikarpova

Interactive Program Synthesis by Augmented Examples
Tianyi Zhang, London Lowmanstone, Xinyu Wang, Elena Glassman


Sketch-n-Sketch
File
Code Tools
View
Options
Output Tools
Current file: Untitled *
Undo
Redo
Clear Up
Run
Context: Program > logoFunc
Code
GUIs
2D Graphics
Documents
Presentations
Spreadsheets
Web Apps
...
...
LaTeX
Scholarly MARKDOWN
LaTeX
SQL
JS
Ai
PAGES
Google Web Designer
Built-In Tools
Cursor
Point or Offset
Polygon
User-Defined Tools
logoFunc
Standard Library Tools
vec2DPlus
vec2DLength
circle
ring
ellipse
rect
square
line
rectByCenter
squareByCenter
nPointsOnCircle
nPointsOnSegment
nPointsSepBy
nHorizontalPointsSepBy
nVerticalPointsSepBy
pointsBetweenSepBy
midpoint
onLine
onPerpendicularLine


Code
GUIs
2D Graphics
Documents
Presentations
Spreadsheets
Web Apps
...
...
Sketch-n-Sketch
File
Code Tools
View
Options
Output Tools
Current file: Untitled *
Undo
Redo
Clean Up
Run
Context: Program > logoFunc
Built-In Tools
Cursor
Point or Offset
Polygon
User-Defined Tools
logoFunc
Standard Library Tools
vec2DPlus
vec2DLength
circle
ring
ellipse
rect
square
line
rectByCenter
squareByCenter
nPointsOnCircle
nPointsOnSegment
nPointsSepBy
nHorizontalPointsSepBy
nVerticalPointsSepBy
pointsBetweenSepBy
midpoint
onLine
onPerpendicularLine

Creative Coding

Khan Academy

Happy Coding

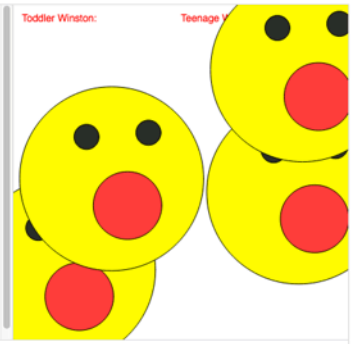
Intro to Variables

```
noStroke();

// face
fill(255, 255, 0);
ellipse(202, 208, 300, 300);

// eyes
fill(46, 46, 41);
ellipse(157, 151, 40, 40);
ellipse(304, 142, 40, 40);

// mouth
fill(252, 65, 65);
ellipse(257, 240, 120, 136);
```

0:00/3:52 Watch on YouTube Spin-off

Function Parameters

```
var drawWinston = function() {
    var faceX = random(50, 350);
    var faceY = random(50, 350);

    fill(255, 255, 0);
    ellipse(faceX, faceY, 220, 220); // face
    fill(46, 46, 41);
    ellipse(faceX - 30, faceY - 50, 30, 30); // eyes
    ellipse(faceX + 44, faceY - 55, 30, 30);
    fill(252, 65, 65);
    ellipse(faceX + 19, faceY + 32, 82, 81); // mouth
};

fill(255, 0, 0);
text("Toddler Winston:", 10, 20);
text("Teenage Winston:", 200, 20);
text("Twenty-Something Winston:", 10, 220);
text("Adult Winston:", 200, 220);
drawWinston();
drawWinston();
drawWinston();
```



0:00/5:03 Watch on YouTube Spin-off

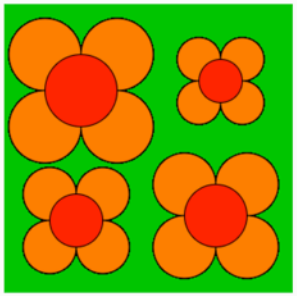
p5.js ▶ Play

```
function setup() {
  createCanvas(300, 300);
  background(0, 200, 0);
}

function draw() {
  drawFlower(150, 150, 100);
}

function drawFlower(flowerX, flowerY, petalSize) {
  let petalDistance = petalSize / 2;
```

This code calls the `drawFlower` function with parameters `150, 150, 100` which draws a flower with an X coordinate of `150`, a Y coordinate of `150`, and a petal size of `100`. Try changing the parameters to see what happens!

Now that you have a `drawFlower` function, you can call it as many times as you want, with whatever parameters you want!

```
function draw() {
  drawFlower(80, 90, 75);
  drawFlower(225, 80, 45);
  drawFlower(75, 225, 55);
  drawFlower(220, 220, 65);
}
```



Creative Coding

p5.js

p5* File Edit Sketch Help Log in or Sign up

Auto-refresh maquina by mecobi

sketch.js

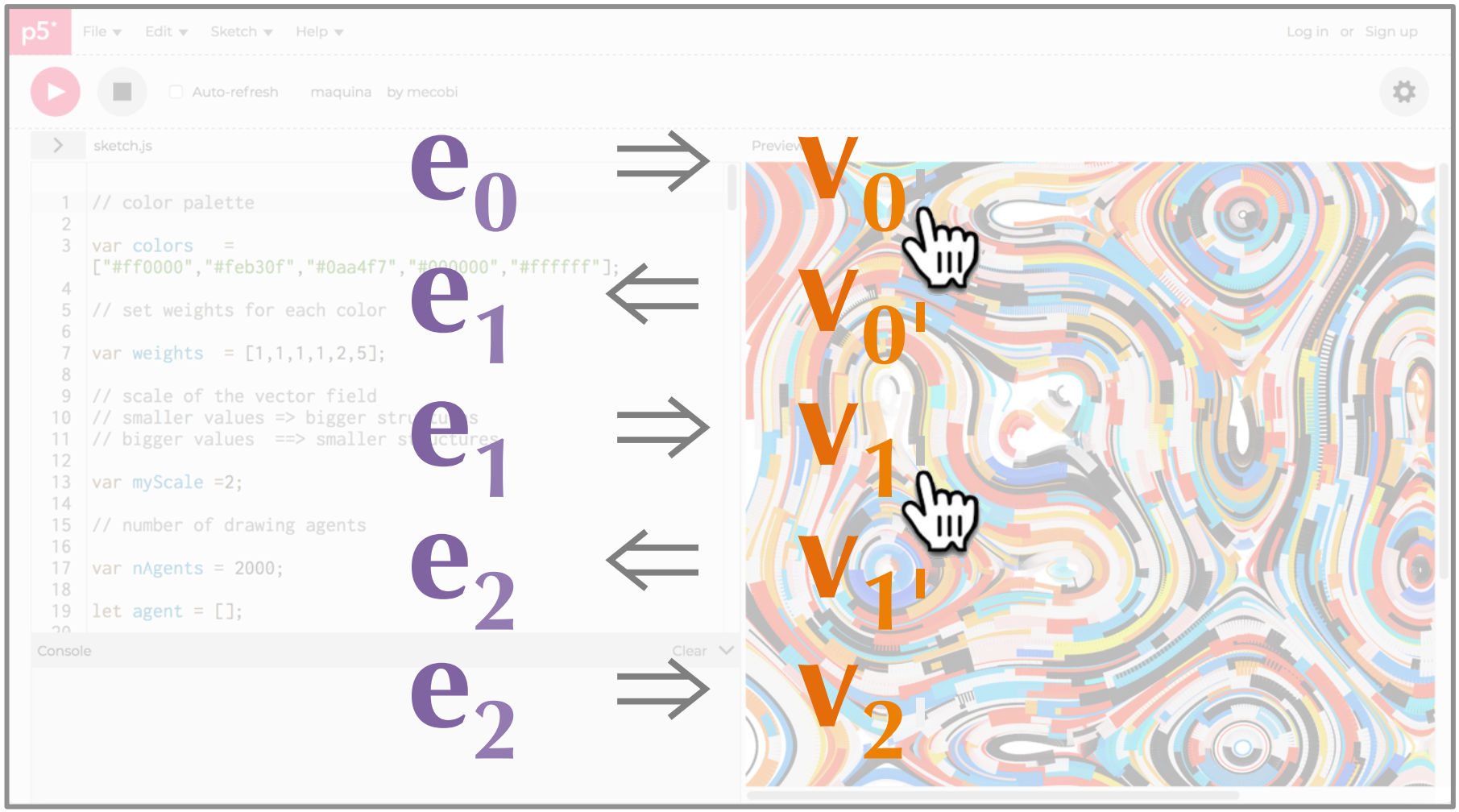
```
// color palette

var colors   =
["#ff0000","#feb30f","#0aa4f7","#000000","#ffffff"];

// set weights for each color

var weights  = [1,1,1,1,2,5];

// scale of the vector field
// smaller values => bigger structures
// bigger values  ==> smaller structures

var myScale =2;

// number of drawing agents

var nAgents = 2000;

let agent = [];
```

Console Clear

Preview

Creative Coding (CMSC 11111)

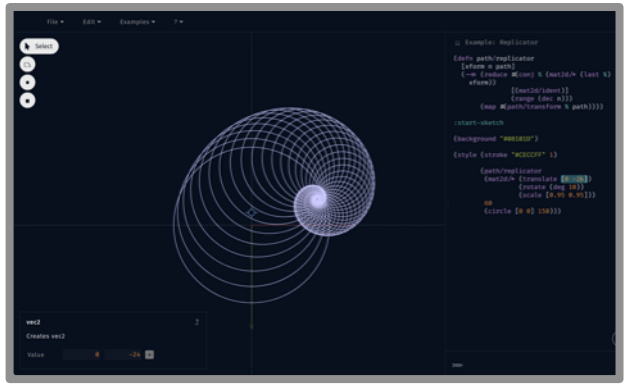
p5.js

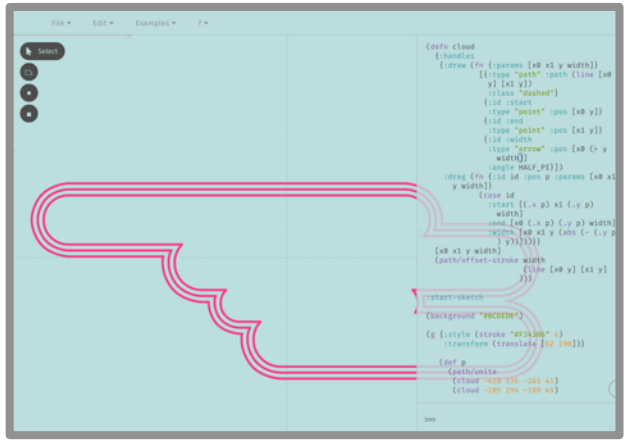


Creative Coding

Glisp

A Lisp-based Design Tool Bridging Graphic Design and Computational Arts.





Ravi Chugh @ravi_chugh · Jun 20

Cool! Backward evaluation is definitely an appropriate name, as is bidirectional programming. If you'd like to take a look, here are a couple ways we've gone about it in Sketch-n-Sketch:

arxiv.org/pdf/1507.02988… (cc @brianhempel)

arxiv.org/pdf/1809.04209… (cc @MikaelMayer)

1 3 5

Baku 麦 @_baku89 · Jun 21

Thank you for the papers, I'm just reading them and no confidence to understand...but I'll try anyway! (my math knowledge has stuck in the basic linear algebra haha) And I just wanted to tell you all that I'm so appreciated and influenced by what SnS is aiming for.

1 2

Ravi Chugh @ravi_chugh · Jun 21

Great, feel free to get in touch if you'd like to chat! Would be happy to talk about different examples and how things work. You may also find some of the videos here interesting:

ravichugh.github.io/sketch-n-sketc…

(My basic linear algebra knowledge is a distant memory...)

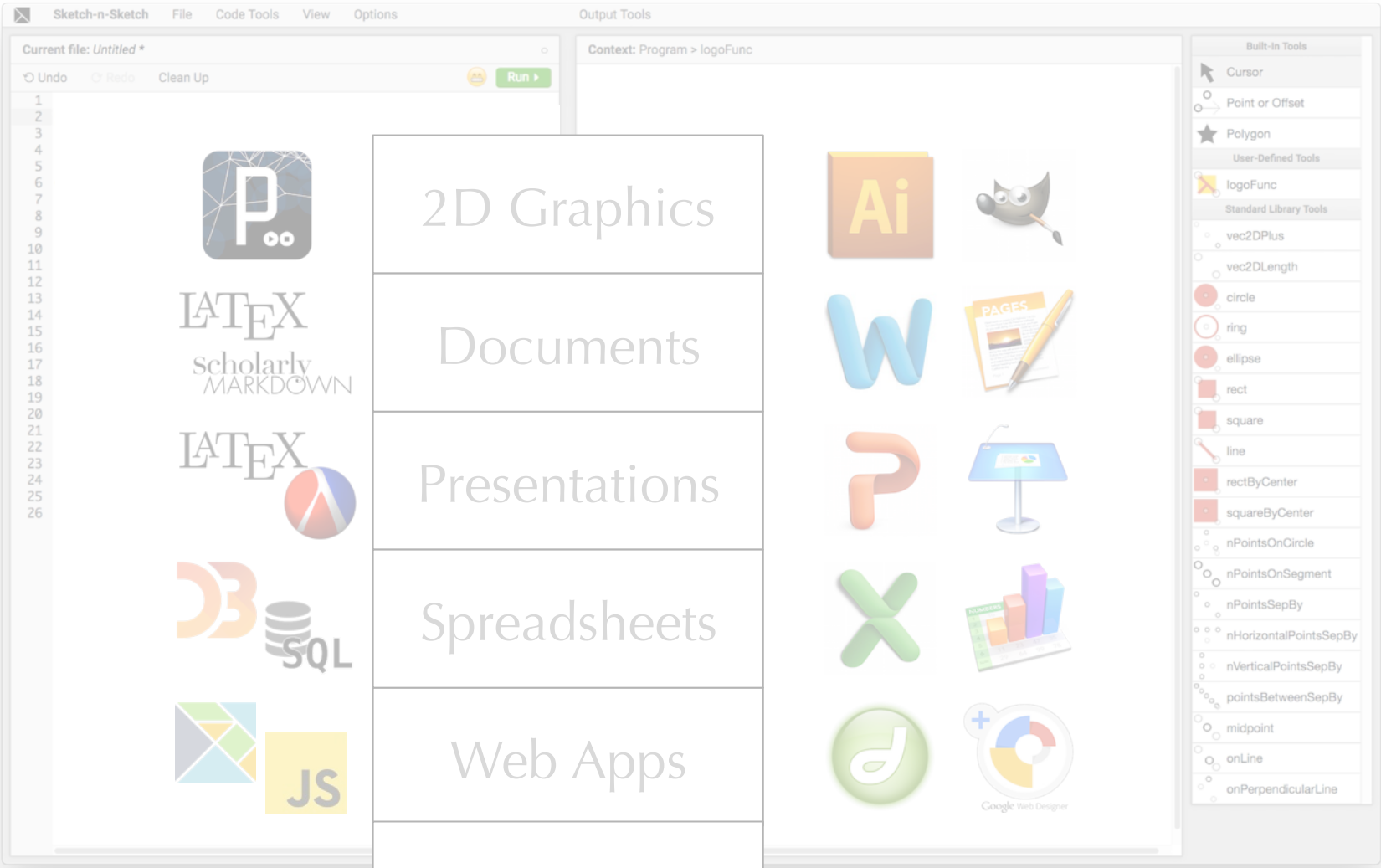
1 3

Baku 麦
@_baku89

Replying to @ravi_chugh @brianhempel and @MikaelMayer

Of course, I've been reading this website these days like a textbook for my prototype ;) Thank you for saying that.

8:52 AM · Jun 21, 2020 · Twitter Web App



2D Graphics

Documents

Presentations

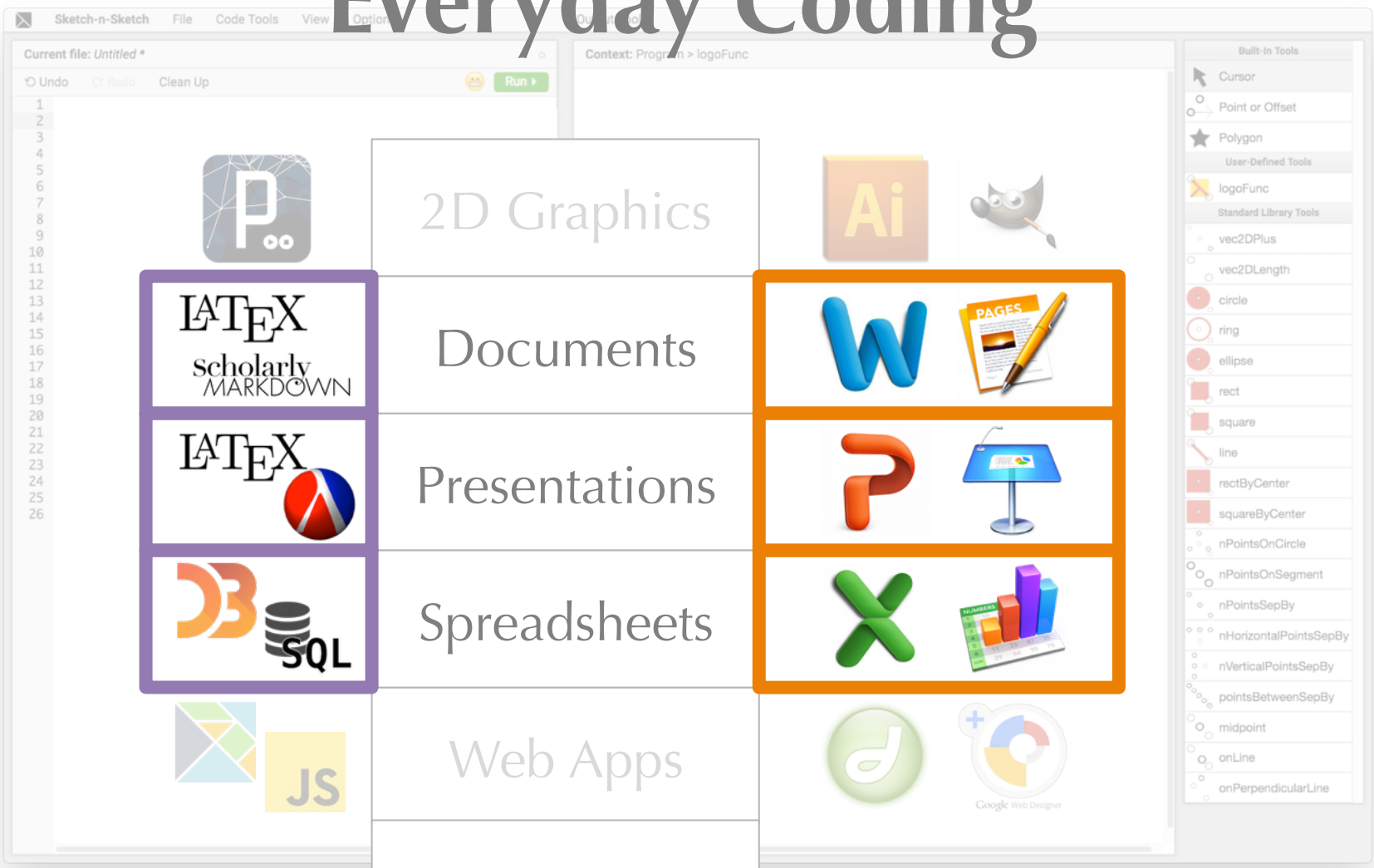
Spreadsheets

Web Apps

...

...

"Everyday Coding"



2D Graphics

Documents

Presentations

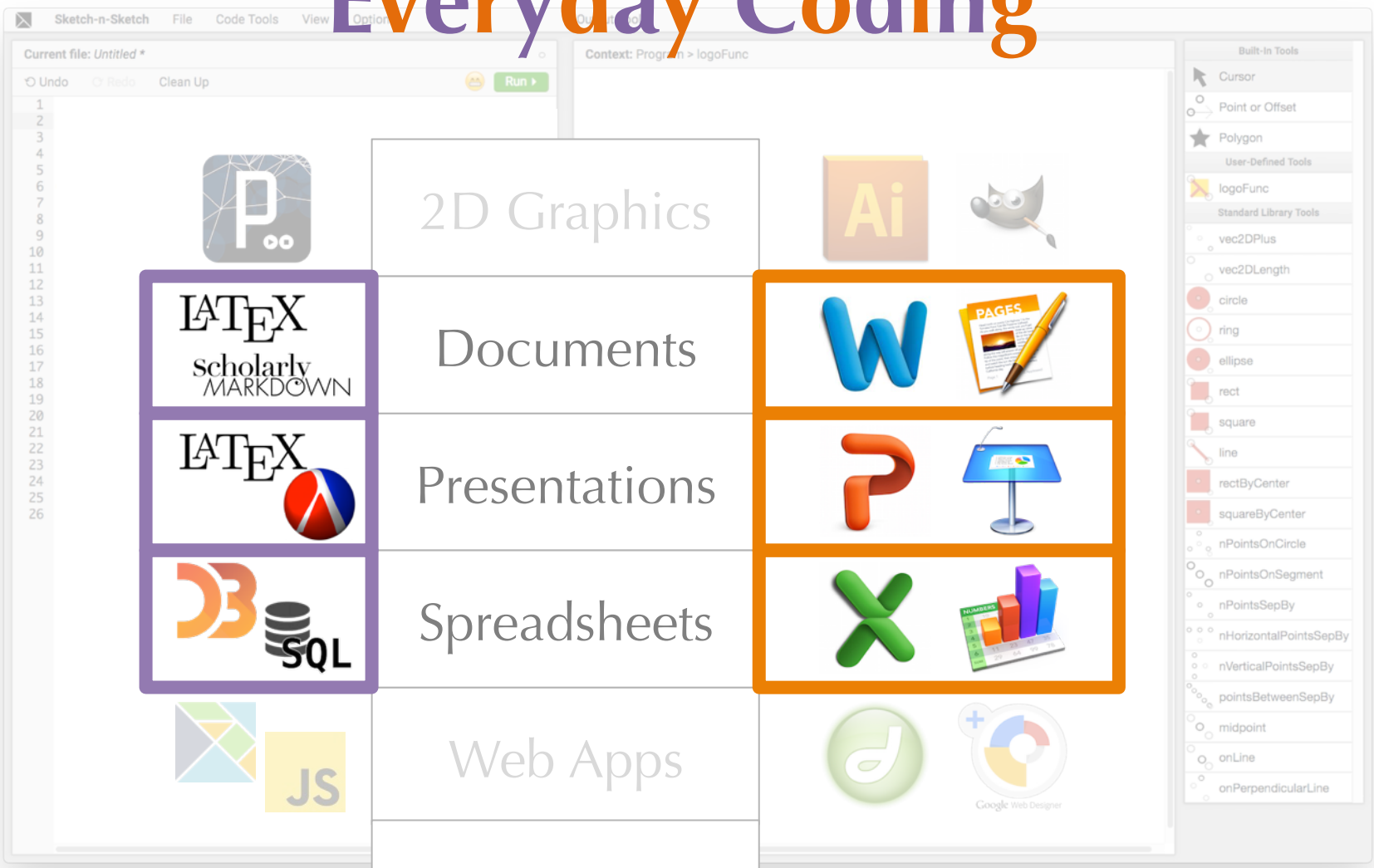
Spreadsheets

Web Apps

...

...

"Everyday Coding"



2D Graphics

Documents

Presentations

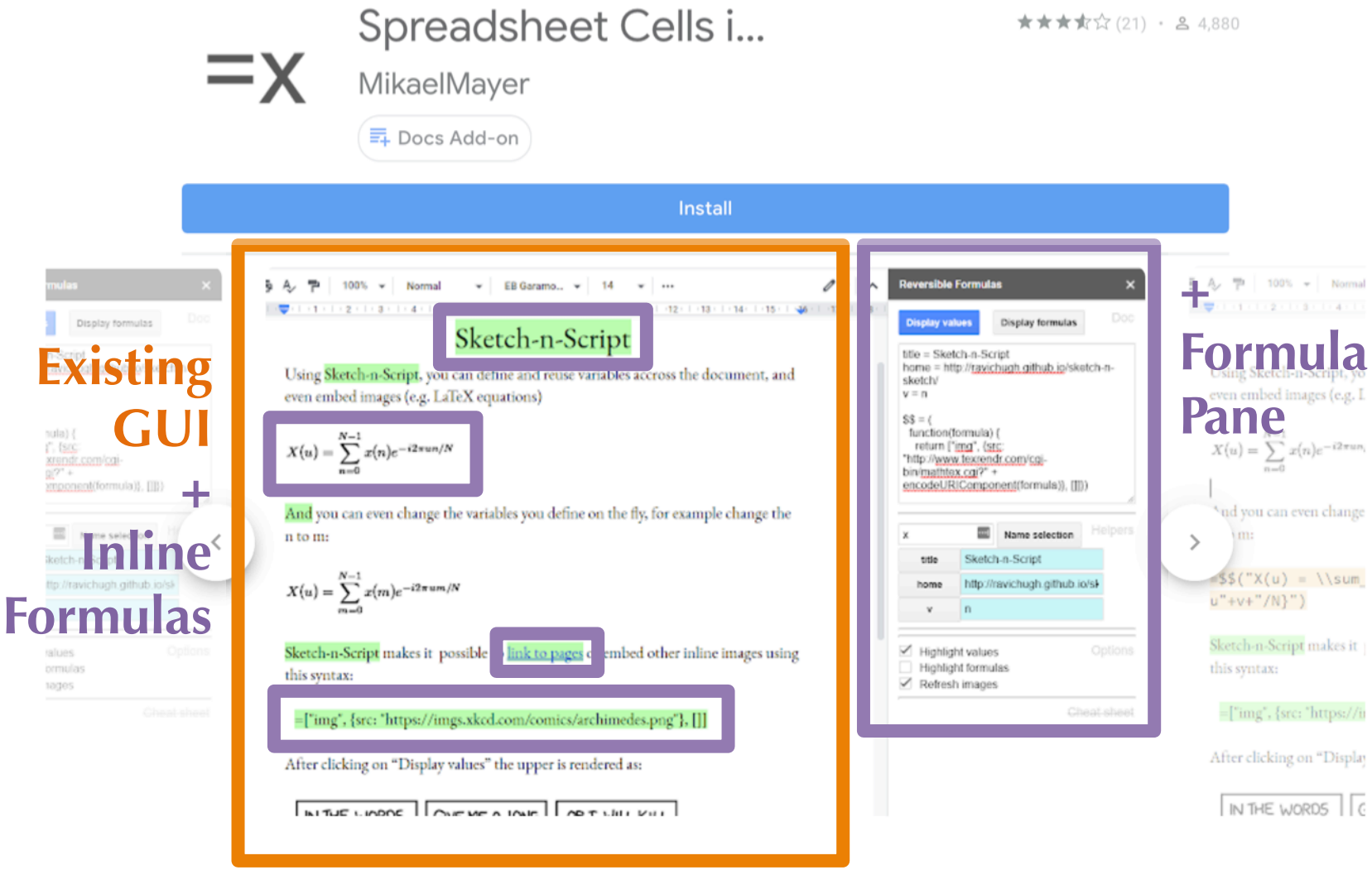
Spreadsheets

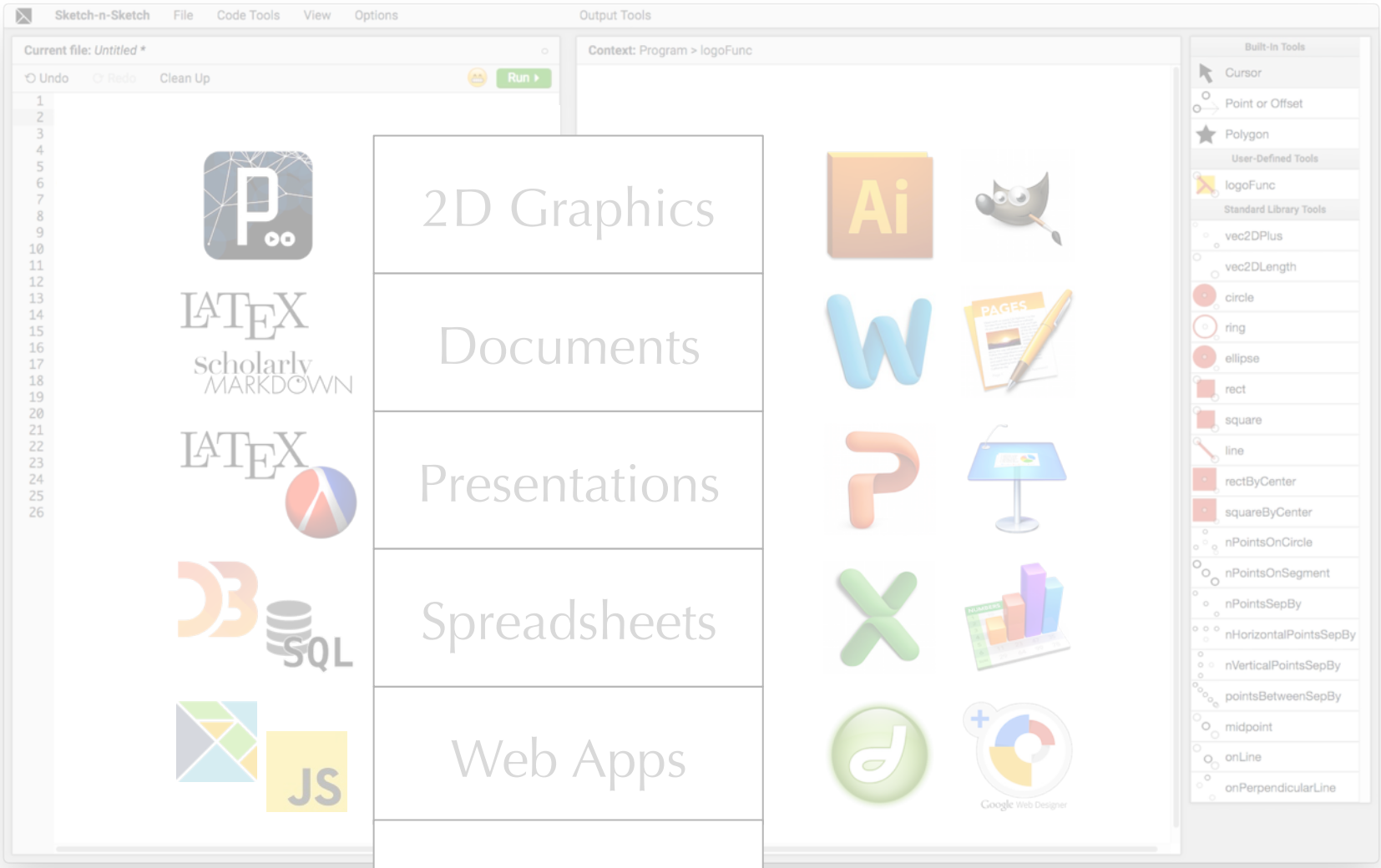
Web Apps

...

...

"Everyday Coding"





2D Graphics

Documents

Presentations

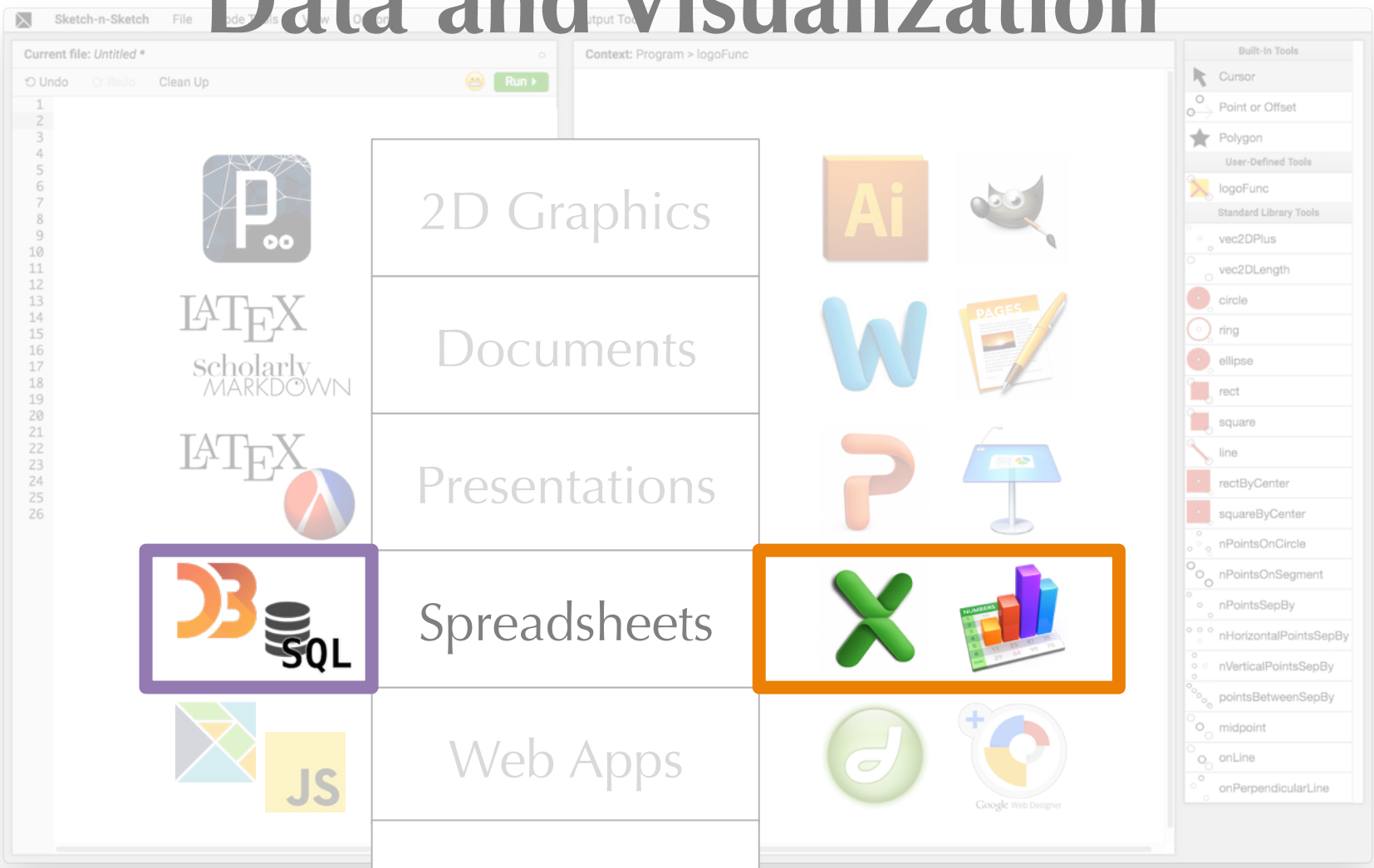
Spreadsheets

Web Apps

...

...

Data and Visualization



2D Graphics

Documents

Presentations

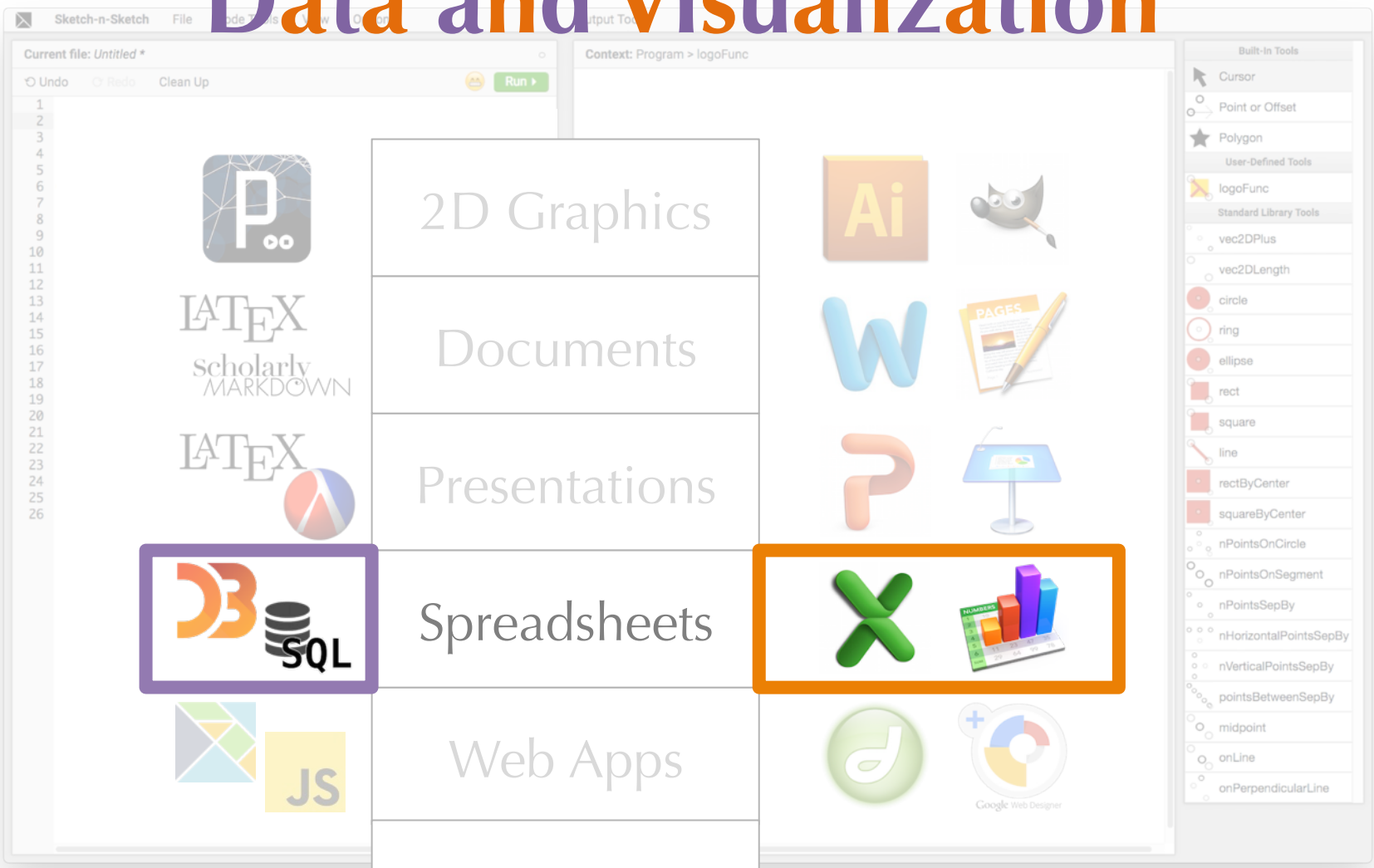
Spreadsheets

Web Apps

...

...

Data and Visualization



2D Graphics

Documents

Presentations

Spreadsheets

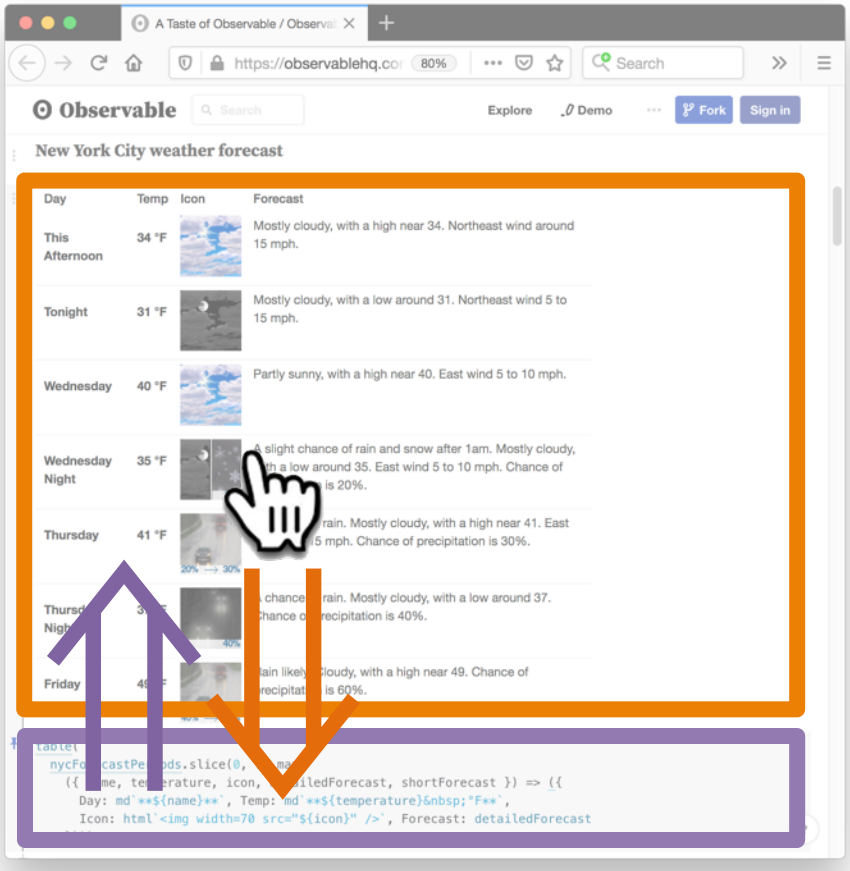
Web Apps

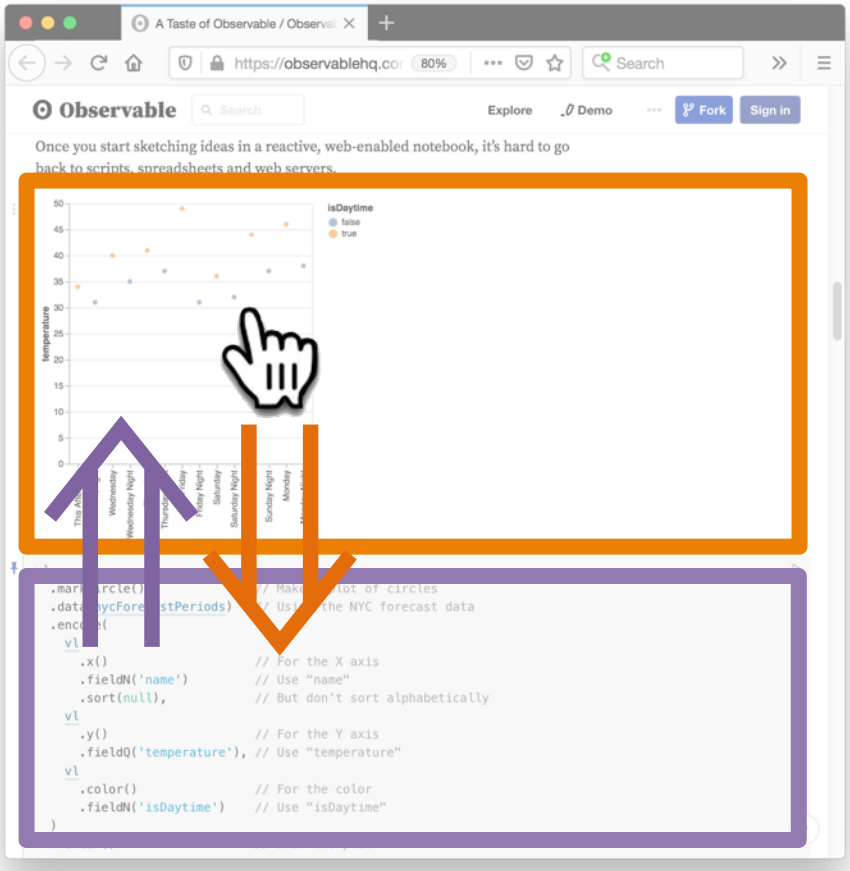
...

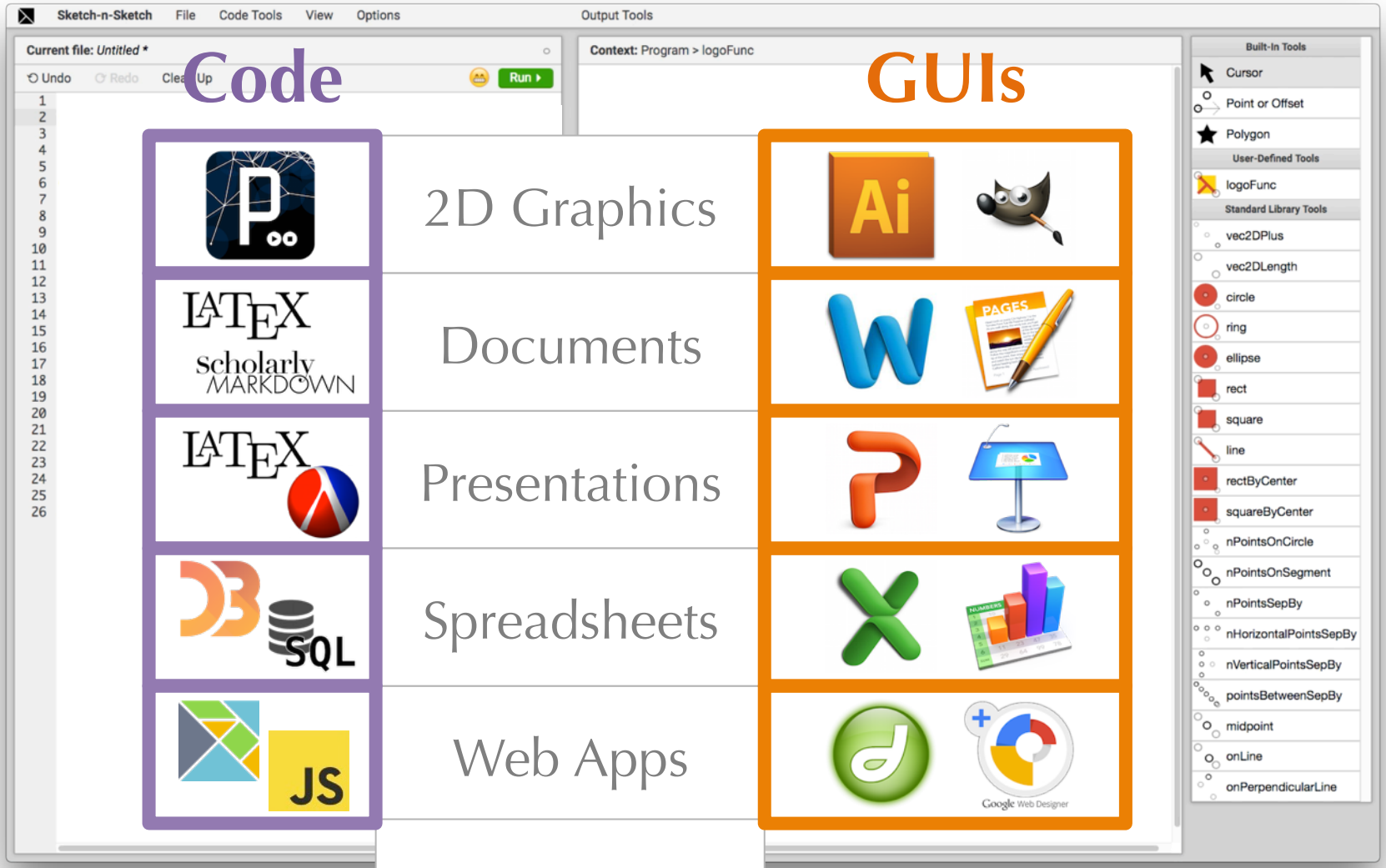
...

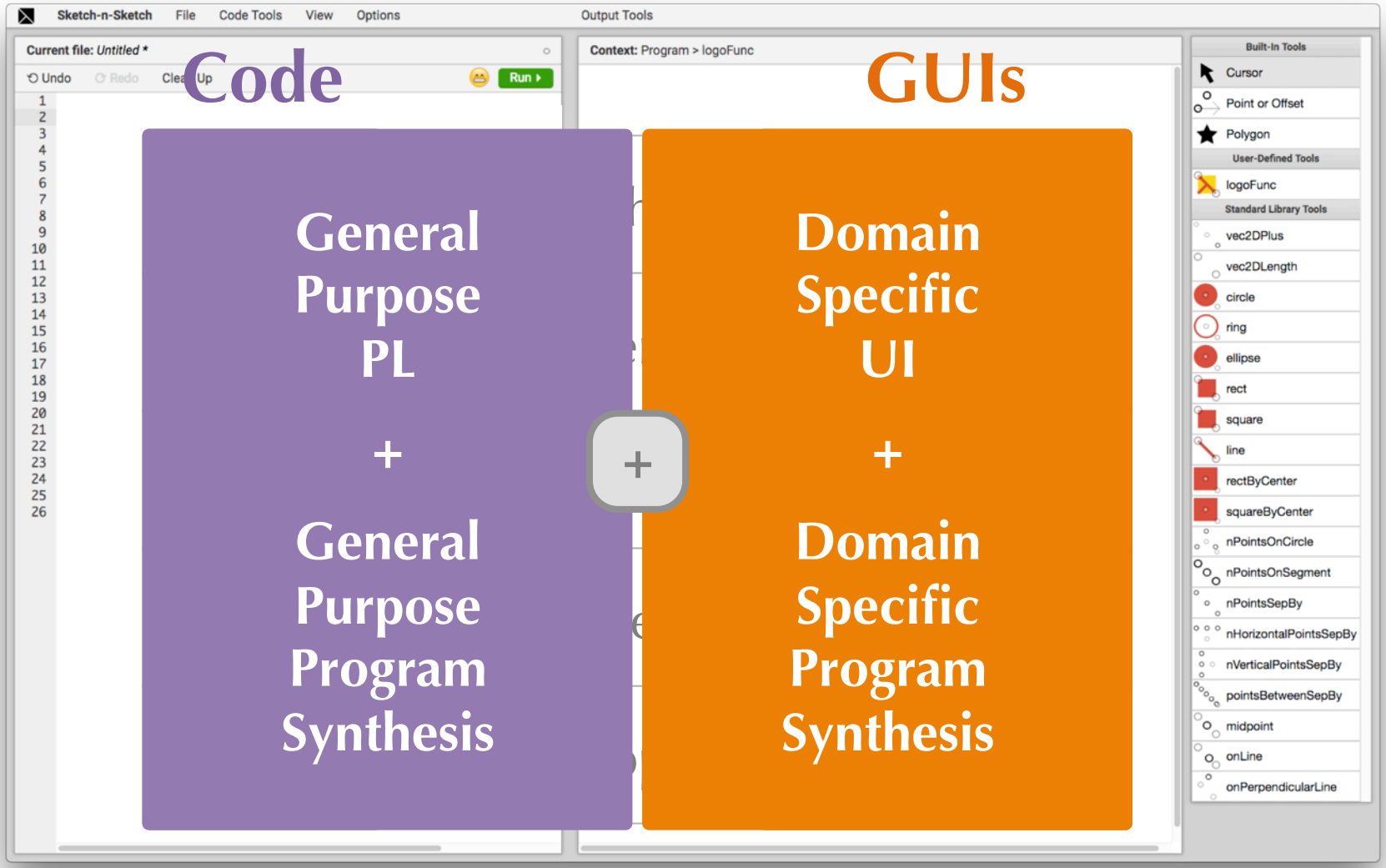
Data and Visualization

Computational Notebooks






Sketch-n-Sketch
File
Code Tools
View
Options
Output Tools
Current file: Untitled *
Undo
Redo
Clear Up
Run
Context: Program > logoFunc
Code
GUIs
Built-In Tools
Cursor
Point or Offset
Polygon
User-Defined Tools
logoFunc
Standard Library Tools
vec2DPlus
vec2DLength
circle
ring
ellipse
rect
square
line
rectByCenter
squareByCenter
nPointsOnCircle
nPointsOnSegment
nPointsSepBy
nHorizontalPointsSepBy
nVerticalPointsSepBy
pointsBetweenSepBy
midpoint
onLine
onPerpendicularLine
LaTeX
Scholarly MARKDOWN
LaTeX
SQL
JS
Ai
PAGES
Google Web Designer
2D Graphics
Documents
Presentations
Spreadsheets
Web Apps
...
...
Text
Trees/Blocks



Text

Trees/Blocks

Thanks!

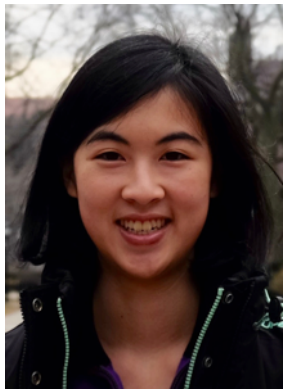
Nick Collins

Brian Hempel

Justin Lubin

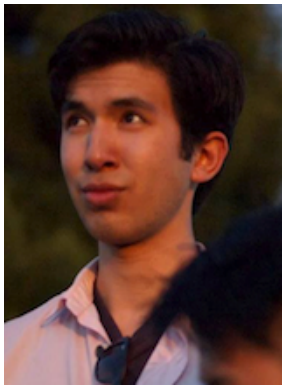
Mikaël Mayer

Cyrus Omar



Grace Lu

Mitch Spradlin



Jacob Albers

SWISS NATIONAL SCIENCE FOUNDATION

Bidirectional Evaluation

[PLDI 2016, OOPSLA 2018]

Live Programming by Example

[POPL 2019, ICFP 2020]

Sketch-n-Sketch

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

Hybrid Text-GUI Code Editors

[ICSE 2018, VL/HCC 2020]

Output-Directed Programming

[UIST 2016, UIST 2019]

EXTRA SLIDES

Bidirectional Evaluation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\rightsquigarrow$$

$$v'$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

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

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

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

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

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

Live Programming by Example

Live Programming by Example

Sketching with

Logic-Based Synthesis

Sketch
Leon
Rosette
Synquid

Our Goal:

Sketching with

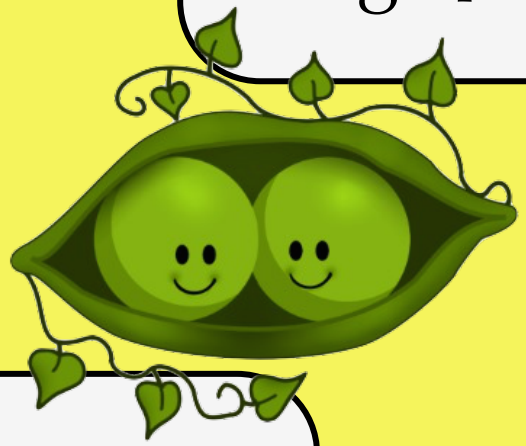
Example-Based Synthesis

Escher
Lambda^2
Myth
Myth2

Two Peas in a Pod

Program Sketching

e.g. [Sketch]



Logic-Based Synthesis

e.g. [Synquid]

Example-Based Synthesis

e.g. [Myth]

Program Sketching

[Sketch]

Sketch-n-Myth

Example-Based Synthesis

[Myth]

Examples from evaluating sketch

Solve multiple interdependent holes

Often requiring fewer user-provided examples

Myth

[Check]

$$\frac{E\,e \Rightarrow v \qquad v \models ex}{e \models (E \vdash \bullet \models ex)}$$

[Check-Input-Output]

$$\frac{e_f = \lambda x.e[\{v_i \rightarrow ex_i\}^{i \in [n]}/f] \quad \{\, e_f\, v_i \Rightarrow v_i' \qquad v_i' \models ex_i \,\}^{i \in [n]}}{\texttt{fix}\, f\, (\lambda x.e) \models \{v_i \rightarrow ex_i\}^{i \in [n]}}$$

Sketch-n-Myth

[Live-Check]

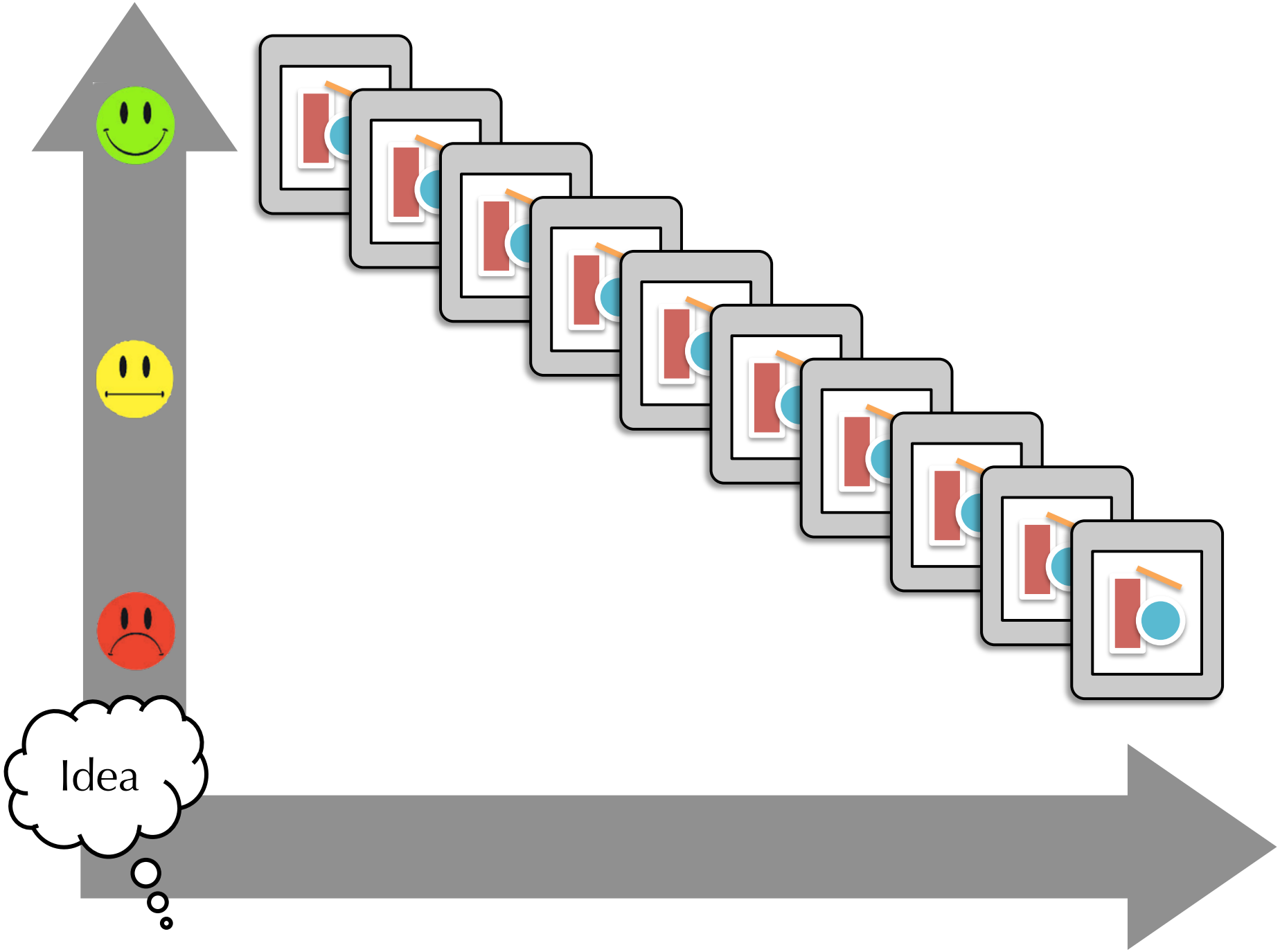
$$\frac{E\,e \Rightarrow r \qquad r \Leftarrow ex \dashv K}{e \models (E \vdash \bullet \models ex) \dashv K}$$

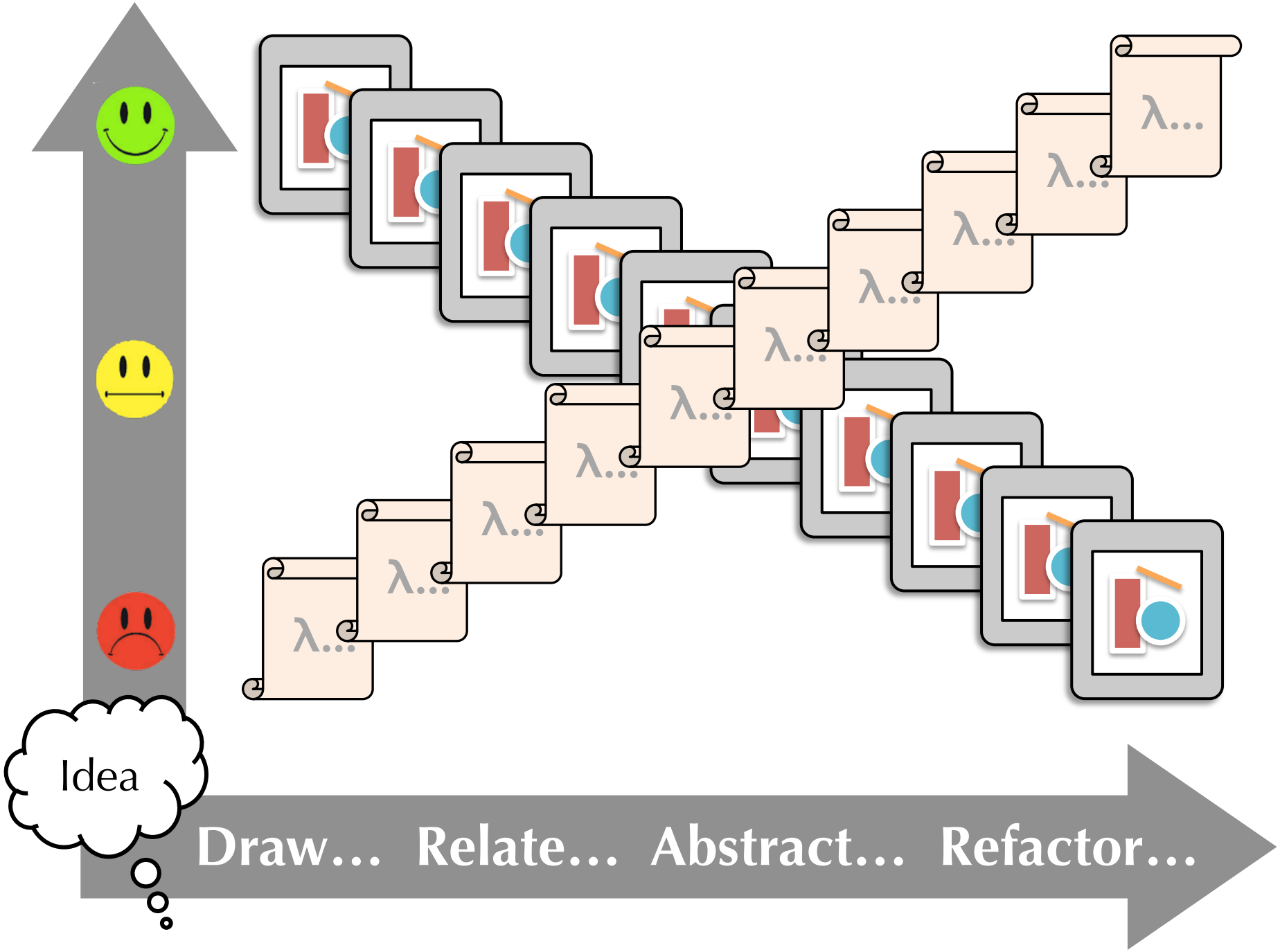
[Live-Check-Input-Output]

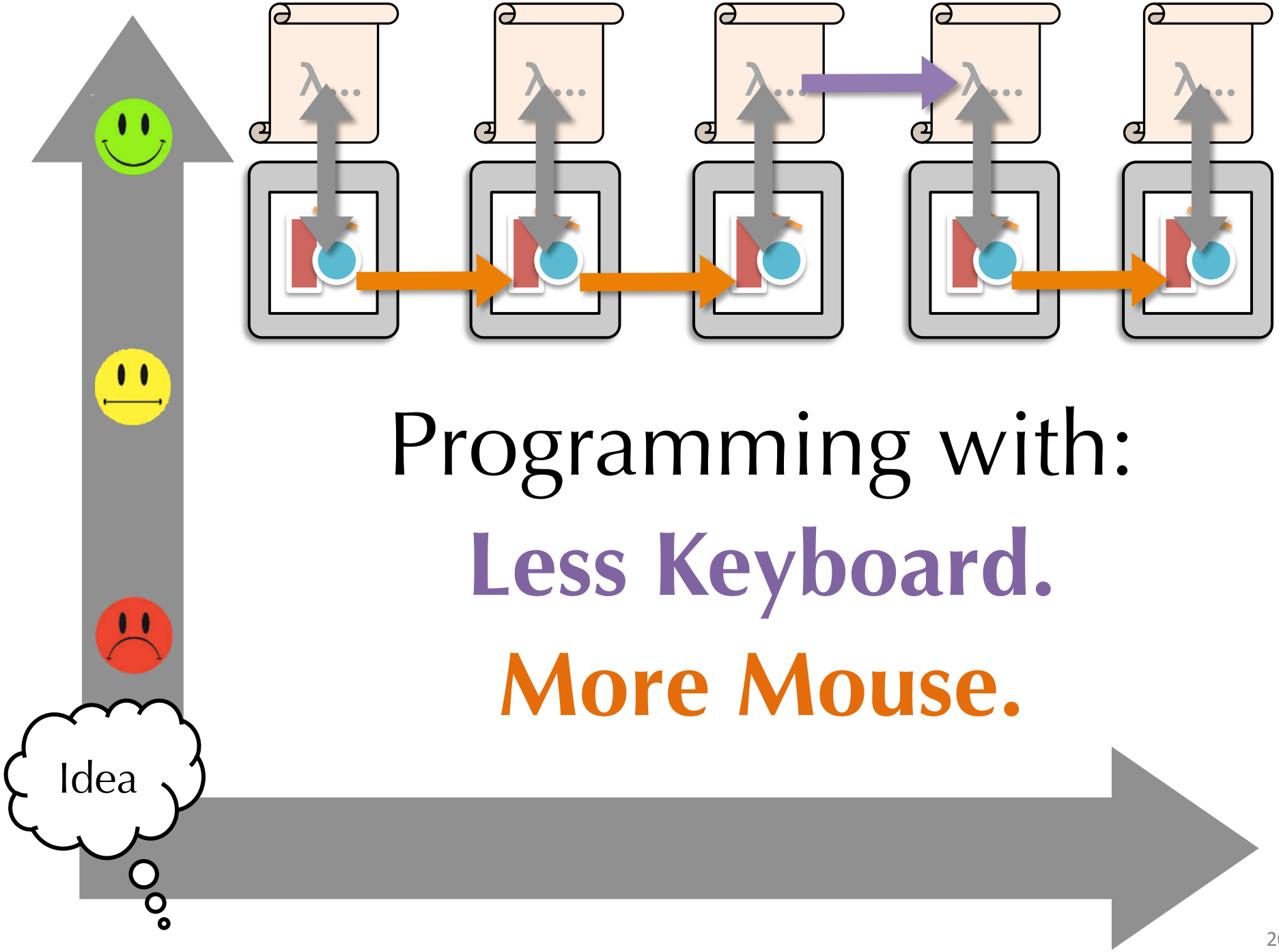
$$\frac{(\texttt{fix}\, f\, (\lambda x.e))\, v \Rightarrow r \qquad r \Leftarrow ex \dashv K}{\texttt{fix}\, f\, (\lambda x.e) \Leftarrow \{v \rightarrow ex\} \dashv K}$$

	Smyth						Leon		Synquid	
Experiment	**1**		**2a**	**2b**	**3a**	**3b**	**4**		**4**	
Sketch / Objective	*None / Top-1*		*None / Top-1*		*Base Case / Top-1-R*					
Name	**Expert**	**Time**	**Expert**	**Random** (50%, 90%)	**Expert**	**Random** (50%, 90%)	**1**	**2a**	**1**	**2a**
bool_band	4	0.004	3 (75%)	(4,4)	•[3]	•[3]	✓	✓	✓	✓
bool_bor	4	0.003	3 (75%)	(4,4)	•[3]	•[3]	✓	✓	✓	✓
bool_impl	4	0.004	3 (75%)	(4,4)	•[3]	•[3]	✓	✓	✓	✓
bool_neg	2	0.001	2 (100%)	(2,2)	•[3]	•[3]	✓	•[4]	✓	•[4]
bool_xor	4	0.009	4 (100%)	(4,4)	•[3]	•[3]	✓	•[4]	✓	•[4]
list_append	6	0.008	4 (67%)	(3,4)	1+1 (33%)	(1+3,1+4)	✓	✗[1]	✓	✗[1]
list_compress	13	timeout	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]
list_concat	6	0.010	3 (50%)	(2,4)	incorrect	(1+3,1+5)	✓	✗[1]	✗[1]	✗[1]
list_drop	11	0.092	5 (45%)	(6,9)	1+2 (27%)	(1+7,↓)	✓	✓	✓	✗[0]
list_even_parity	7	overspec	•[1]	(–,–)	•[1]	(–,–)	•[1]	•[1]	•[1]	•[1]
list_filter	9*	0.144	5 (56%)	•[2]	1+4 (56%)	•[2]	✗[2]	✗[2]	✗[2]	✗[2]
list_fold	9	0.838	3 (33%)	•[2]	1+3 (44%)	•[2]	✗[2]	✗[2]	✗[2]	✗[2]
list_hd	3	0.003	2 (67%)	(2,3)	•[3]	•[3]	✓	✓	✓	✓
list_inc	4	0.018	2 (50%)	(2,2)	•[3]	•[3]	✓	✓	✗[0]	✗[1]
list_last	6	0.007	4 (67%)	(5,9)	1+2 (50%)	(1+5,1+10)	✓	✓	✓	✗[0]
list_length	3	0.002	3 (100%)	(3,4)	1+1 (67%)	(1+2,1+2)	✓	•[4]	✓	•[4]
list_map	8	0.049	4 (50%)	•[2]	1+2 (38%)	•[2]	✗[2]	✗[2]	✗[2]	✗[2]
list_nth	13	0.124	5 (38%)	(7,14)	1+2 (23%)	(1+7,1+15)	✓	✓	✓	✗[0]
list_pairwise_swap	7	0.634	5 (71%)	timeout	overspec	timeout	✓	✓	✗[0]	✗[0]
list_rev_append	5	0.107	3 (60%)	(5,8)	1+2 (60%)	(1+3,1+4)	✓	✓	✗[0]	✗[0]
list_rev_fold	5	0.035	2 (40%)	(2,4)	•[3]	•[3]	✓	✓	✗[0]	✗[0]
list_rev_snoc	5	0.010	3 (60%)	(3,6)	1+1 (40%)	(1+2,1+4)	✓	✓	✗[1]	✗[0]
list_rev_tailcall	8	0.008	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+5)	✗[1]	✓	✓	✗[1]
list_snoc	8	0.012	3 (38%)	(3,4)	1+1 (25%)	(1+3,1+4)	✓	✓	✓	✗[0]
list_sort_sorted_insert	7	0.015	3 (43%)	(3,6)	1+1 (29%)	(1+2,1+4)	✓	✓	✗[0]	✗[1]
list_sorted_insert	12	2.902	7 (58%)	timeout	1+7 (67%)	timeout	✗[0]	✗[0]	✗[0]	✗[0]
list_stutter	3	0.003	2 (67%)	(3,3)	1+1 (67%)	(1+2,1+3)	✓	✓	✓	✗[1]
list_sum	3	0.029	2 (67%)	(2,2)	•[3]	•[3]	✓	✗[1]	✗[0]	✗[0]
list_take	12	0.065	5 (42%)	(6,9)	1+3 (33%)	(1+7,1+16)	✓	✓	✓	✗[0]
list_tl	3	0.002	2 (67%)	(2,3)	•[3]	•[3]	✓	✓	✓	✓
nat_add	9	0.006	4 (44%)	(5,6)	1+1 (22%)	(1+3,1+4)	✓	✓	✓	✗[1]
nat_iseven	4	0.003	3 (75%)	(4,4)	1+2 (75%)	(1+3,1+4)	✓	✓	✓	✗[0]
nat_max	9	0.041	9 (100%)	(8,12)	1+4 (56%)	(1+8,1+12)	✗[1]	•[4]	✓	•[4]
nat_pred	3	0.001	2 (67%)	(2,3)	•[3]	•[3]	✓	✓	✓	✓
tree_binsert	20	timeout	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]
tree_collect_leaves	6	0.074	3 (50%)	$(3,4)^{t=3}$	1+2 (50%)	(1+3,1+3)	✓	✓	✗[1]	✗[1]
tree_count_leaves	7	2.660	3 (43%)	timeout	1+1 (29%)	timeout	✓	✓	✗[0]	✗[0]
tree_count_nodes	6	0.351	3 (50%)	$(4,\downarrow)^{t=10}$	1+2 (50%)	$(1+3,1+5)^{t=3}$	✓	✓	✗[1]	✗[0]
tree_inorder	5	0.123	4 (80%)	(3,4)	1+2 (60%)	(1+3,1+4)	✓	✓	✗[1]	✗[0]
tree_map	7	0.061	4 (57%)	•[2]	1+3 (57%)	•[2]	✗[2]	✗[2]	✗[2]	✗[2]
tree_nodes_at_level	11	timeout	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]
tree_postorder	20	timeout	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]	•[1]
tree_preorder	5	0.153	3 (60%)	$(3,4)^{t=3}$	1+2 (60%)	(1+3,1+3)	✓	✓	✗[1]	✗[1]
Averages			61%*		46%					

Output-Directed Programming


Idea


Idea
Draw… Relate… Abstract… Refactor…

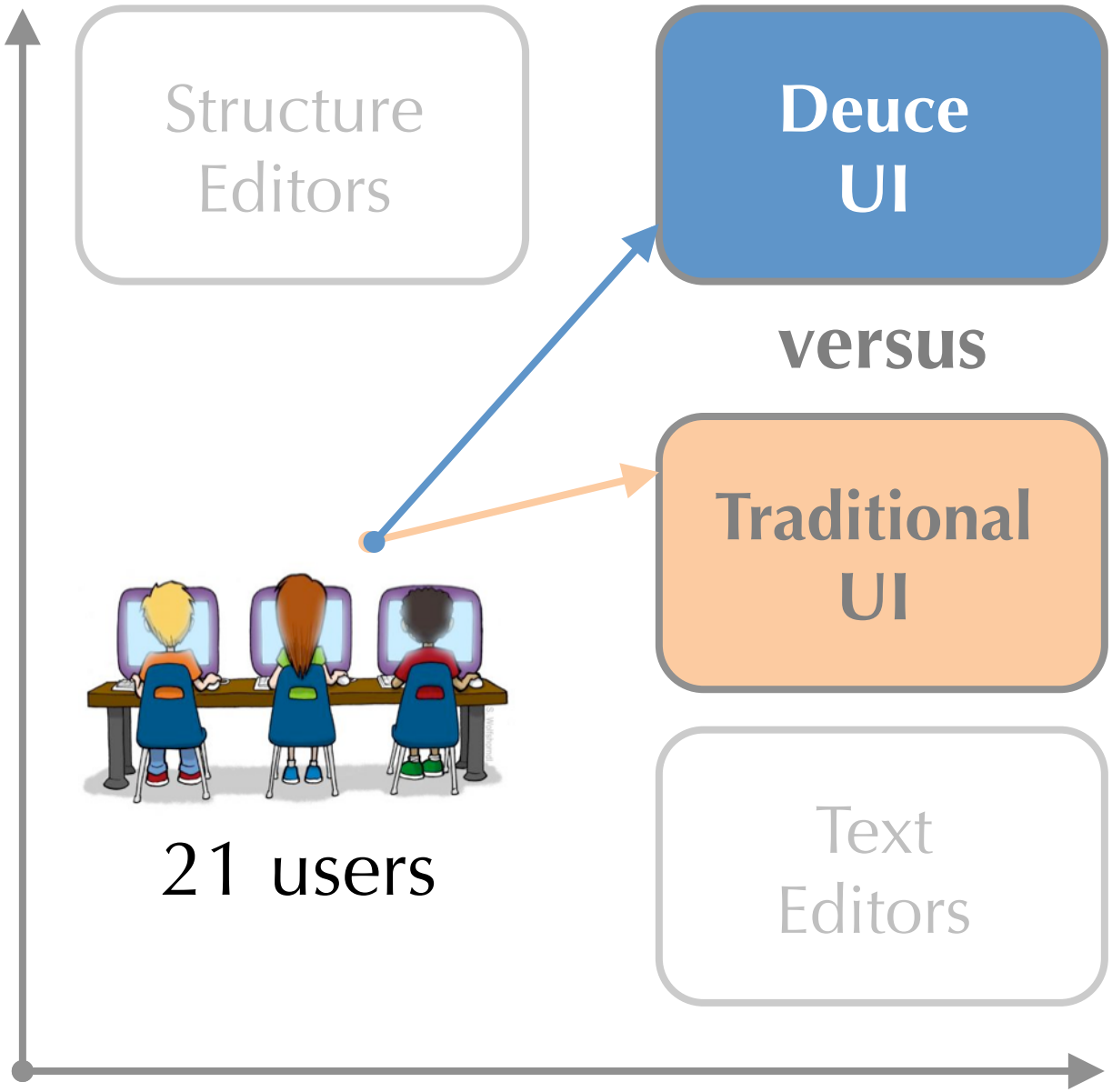


Programming with:

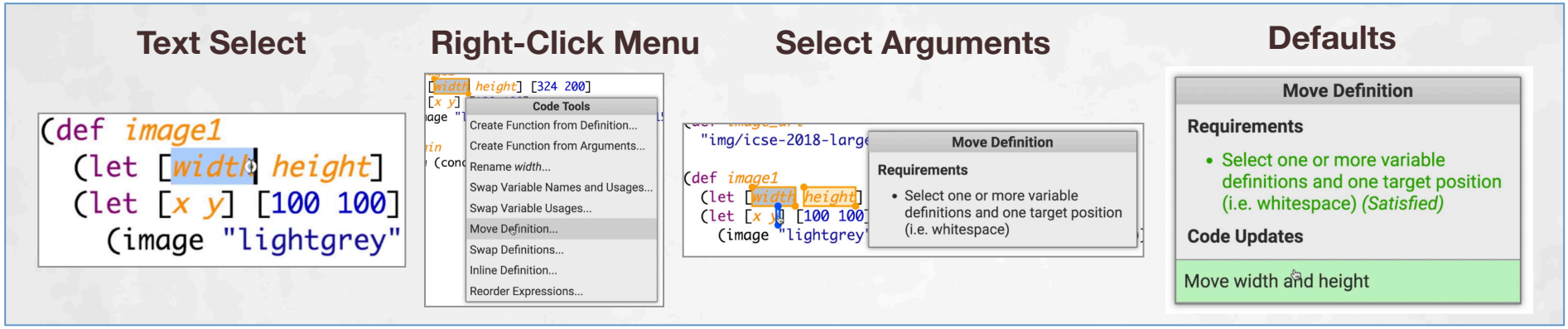
Less Keyboard.

More Mouse.

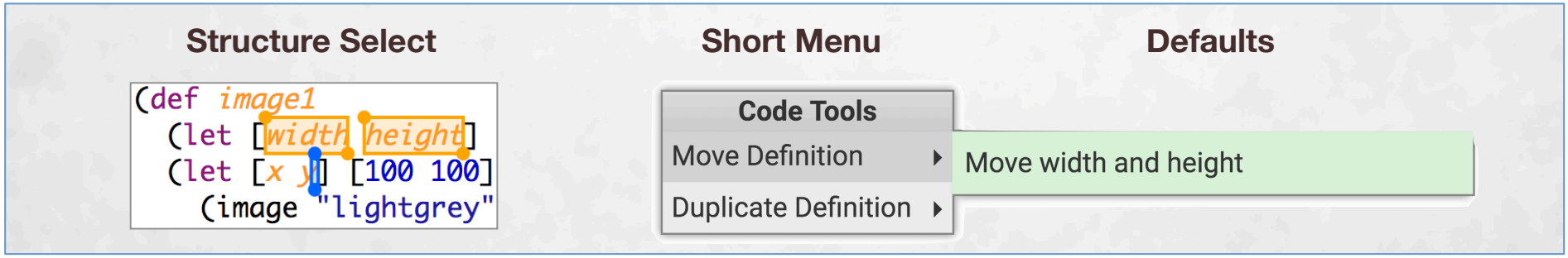
Hybrid Text-GUI Code Editors


Structure Editors
Deuce UI
versus
Traditional UI
21 users
Text Editors

Traditional ("Text-Select") Mode



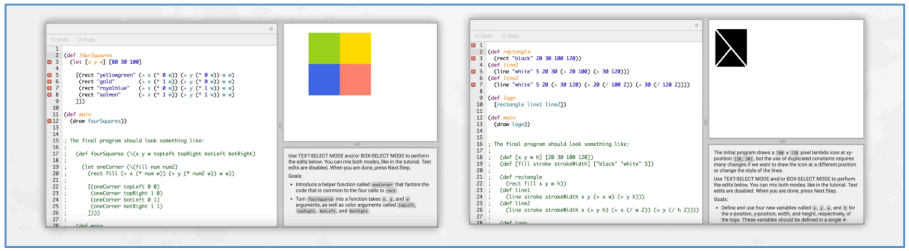
Deuce ("Box-Select") Mode



1. Tutorial

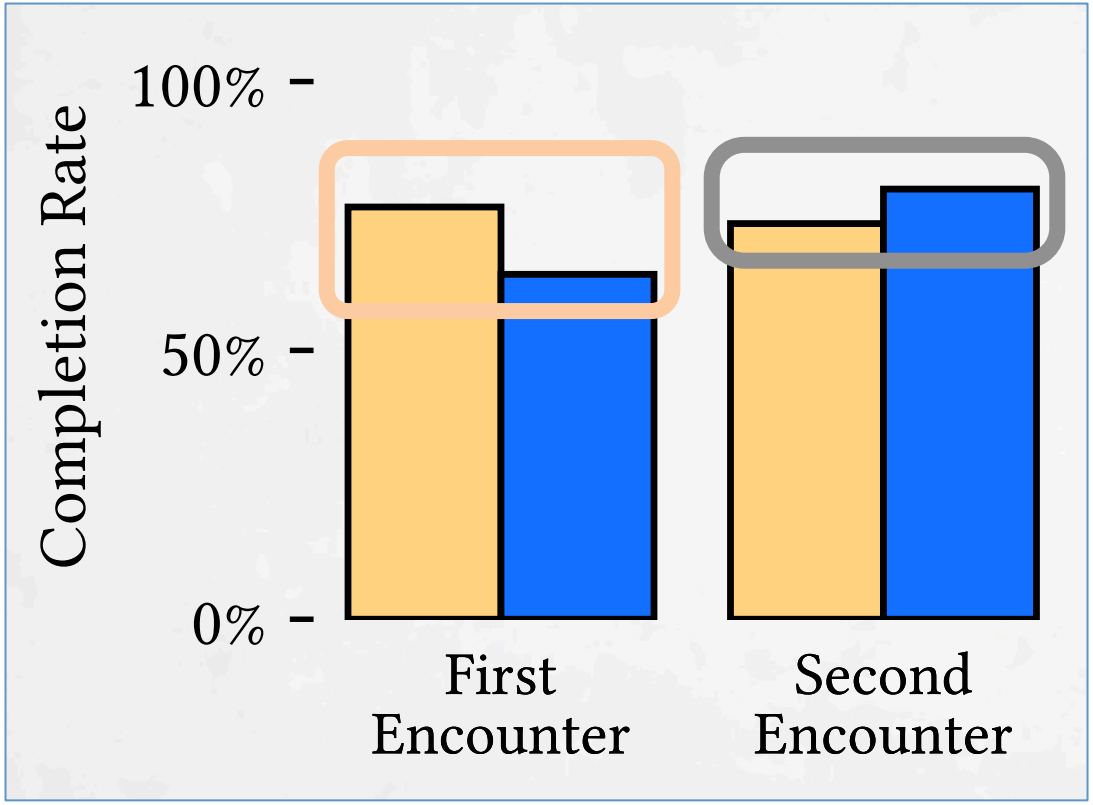
2. Head-to-Head Tasks (2x each; once per mode)

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



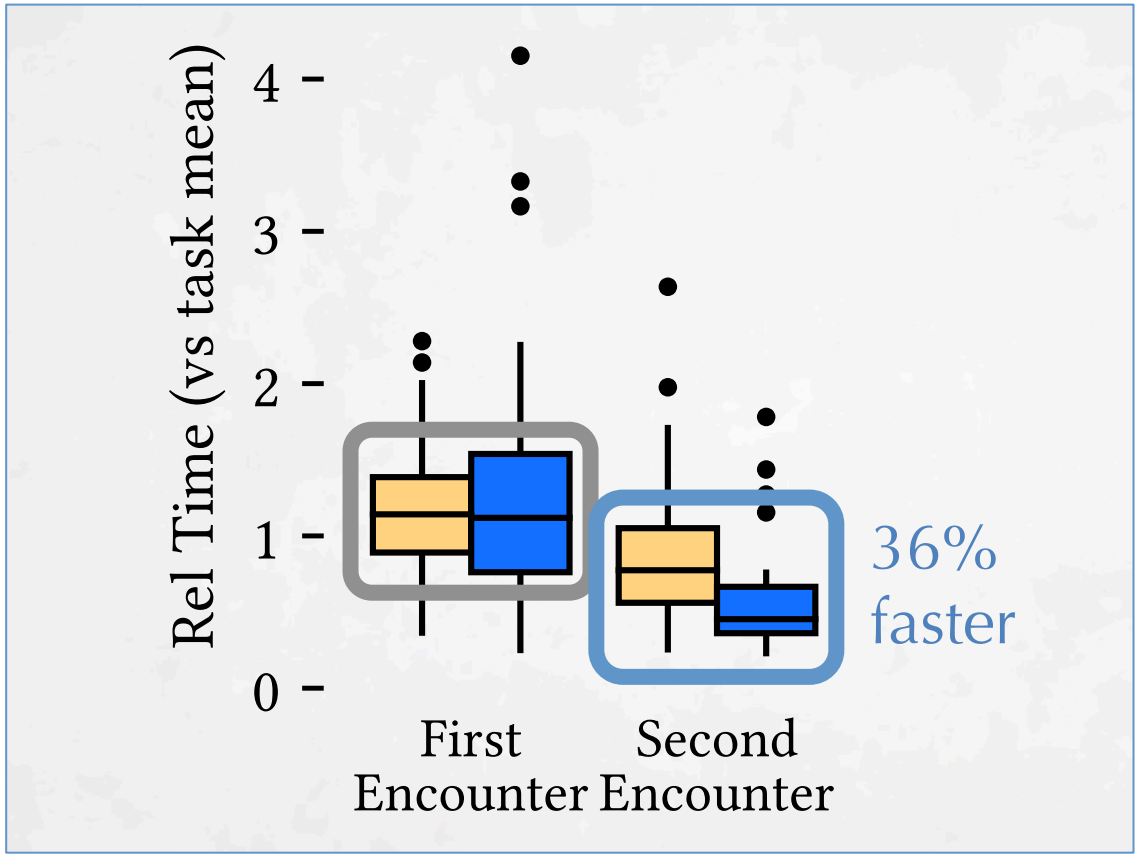
4. Exit Survey

Deuce more effective than Traditional?



Deuce doesn't help discoverability

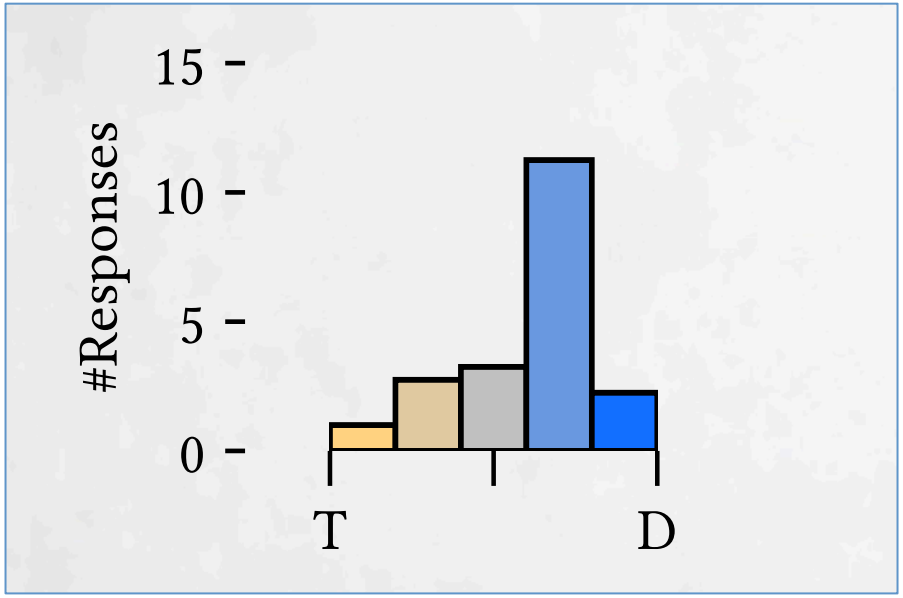
Deuce more effective than Traditional?



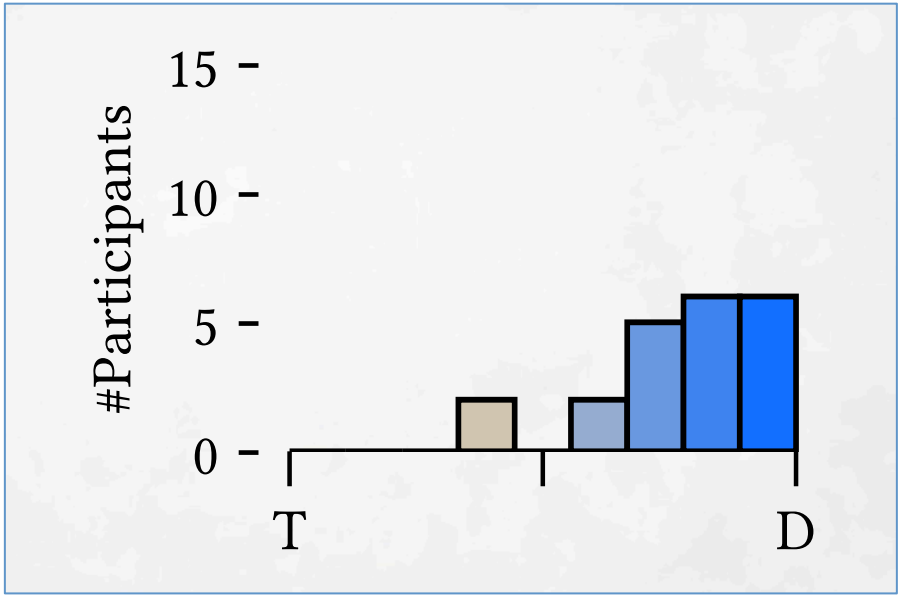
Deuce may be faster once learned

Deuce preferred to Traditional?

Survey



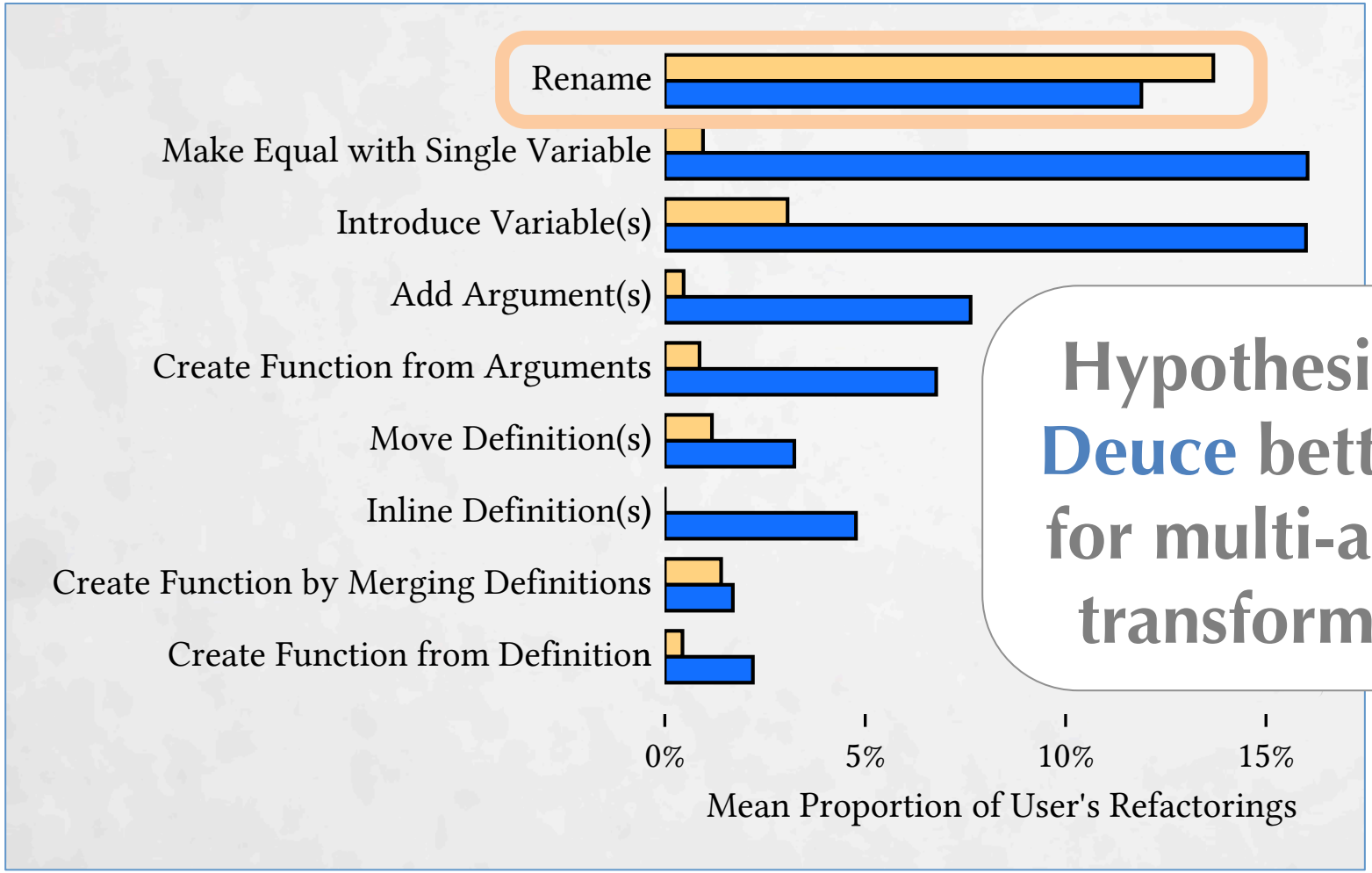
Observed



Modest subjective preference for Deuce

Almost everyone used Deuce more

Mix-and-Match Tool Usage



Hypothesis: Deuce better for multi-arg transforms

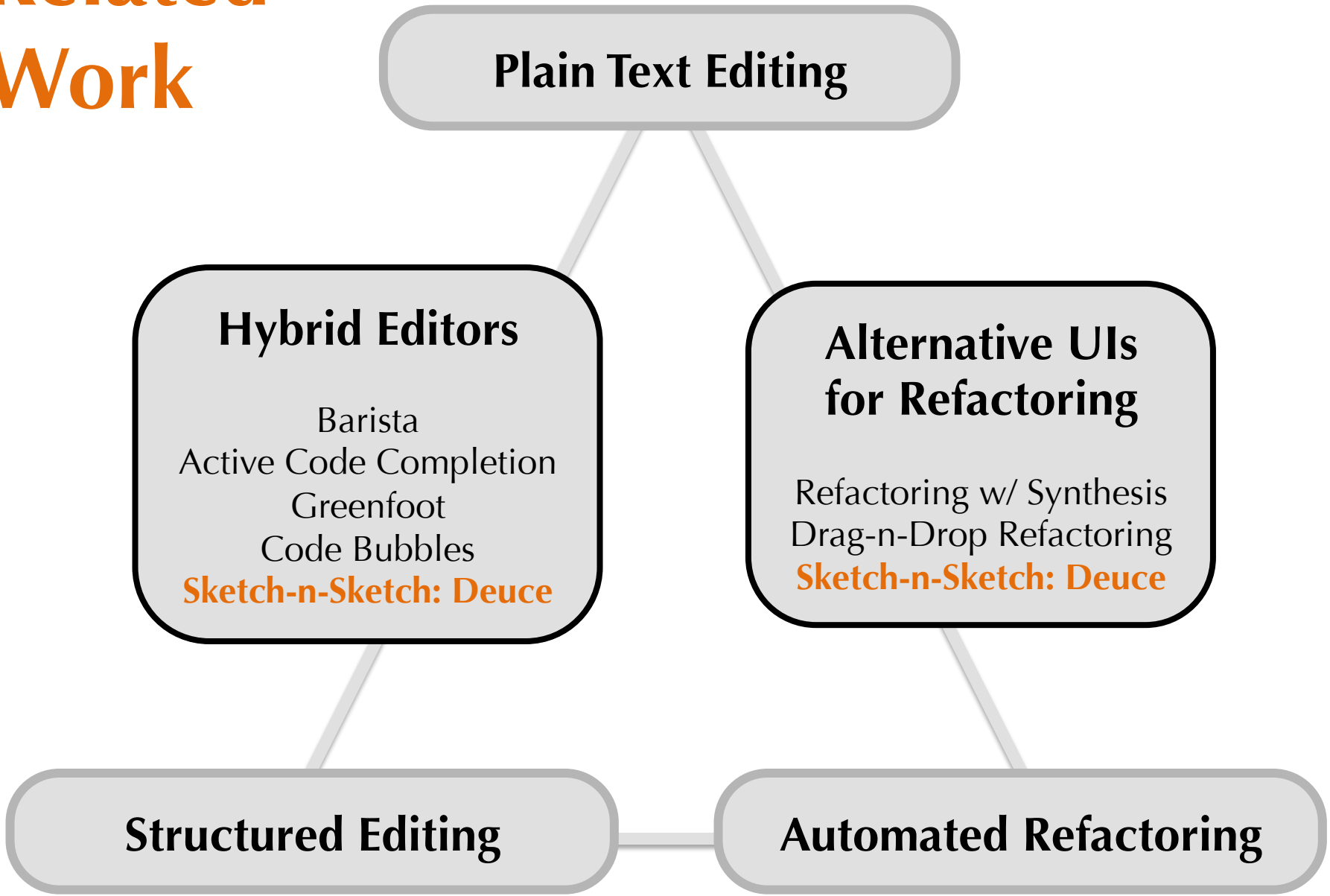
Deuce versus Traditional

Traditional may be better for discovery

Deuce may be faster once learned

Deuce strongly preferred

Related Work



END