Reimagining the User Interfaces for Programming

Ravi Chugh
User Interfaces for Programming
I ♥ GUIs
I love GUIs
I ❤️ GUIs
I love PLs

I love GUIs
I♥PLs

I♥GUIs
<table>
<thead>
<tr>
<th>PLs</th>
<th>GUls</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Graphics</td>
<td><img src="image" alt="P" /></td>
</tr>
<tr>
<td>Documents</td>
<td><img src="image" alt="LaTeX" /></td>
</tr>
<tr>
<td>Presentations</td>
<td><img src="image" alt="LaTeX" /></td>
</tr>
<tr>
<td>Spreadsheets</td>
<td><img src="image" alt="DB" /> <img src="image" alt="SQL" /></td>
</tr>
<tr>
<td>Web Apps</td>
<td><img src="image" alt="JS" /></td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>PLs</td>
<td>GUIs</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>2D Graphics</td>
<td><img src="image1" alt="Illustrator" /></td>
</tr>
<tr>
<td>Documents</td>
<td><img src="image2" alt="LaTeX" /></td>
</tr>
<tr>
<td>Presentations</td>
<td><img src="image3" alt="MarkDown" /></td>
</tr>
<tr>
<td>Spreadsheets</td>
<td><img src="image4" alt="DB" /></td>
</tr>
<tr>
<td>Web Apps</td>
<td><img src="image5" alt="SQL" /></td>
</tr>
<tr>
<td>Programs</td>
<td><img src="image6" alt="JavaScript" /></td>
</tr>
<tr>
<td>Text</td>
<td><img src="image7" alt="Scratch" /></td>
</tr>
<tr>
<td>Trees/Blocks</td>
<td><img src="image8" alt="Snap!" /></td>
</tr>
</tbody>
</table>
PLs + GUIs

2D Graphics

Presentations

Documents

Spreadsheets

Web Apps

Programs

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]

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=6TGBTuuaV3E
```plaintext
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
    List.map
      (Html.td [color])
      row
in
  ListindexedMap drawRow rows
in
Html.table [padding] [] (headerRow :: stateRows)
```

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

https://www.youtube.com/watch?v=GE6KSG5Q__0
Sketch the **Code** + Sketch the **Output**

Sketch the Code

Directly Manipulate Code

Text-Edit Code

Directly Manipulate Output
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]
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]
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 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"
⇒
"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"

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

⇐

"Bx 2019
@ PLW 2018"
let \( f \ x = x ++ " \ 2018" \) in
\( f \ "B" ++ " @ " ++ f \ "PLW" \)

⇒
\( "B 2018 @ PLW 2018" \)

let \( f \ x = x ++ " \ 2019" \) in
\( f \ "Bx" ++ " @ " ++ f \ "PLW" \)

⇐
\( "Bx 2019 @ PLW 2018" \)
let \( f \) \( x = x + \text{" 2018"} \) in
\[
\text{\"B\" ++ \" @ \" ++ f \"PLW\"}
\]

\[
\text{\"B 2018 @ PLW 2018"}
\]

\[
\Rightarrow
\]

let \( f \) \( x = x + \text{" 2019"} \) in
\[
\text{\"Bx\" ++ \" @ \" ++ f \"PLW\"}
\]

\[
\text{\"Bx 2019 @ PLW 2018"}
\]

\[
\Leftarrow
\]

let \( f \) \( x = x + \text{" 2019"} \) in
\[
\text{\"Bx\" ++ \" @ \" ++ f \"PLW\"}
\]

\[
\text{\"Bx 2019 @ PLW 2019"}
\]

\[
\Rightarrow
\]
let \( f \ x = \ x \ ++ \ " \ 2018" \) in
\( f \ "B" \ ++ \ " @ " \ ++ \ f \ "PLW" \)

⇒

"B 2018 @ PLW 2018"

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"
```

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

```
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

$$f \ x \ \Rightarrow \ y$$

Repair "code" + "data"

$$f' \ x' \ \Leftarrow \ y'$$

Extra repairs

$$f'' \ x'' \ \Rightarrow \ y''$$
Unrestricted function

Repair "code" + "data"

Extra repairs
Unrestricted function

L x

Repair "code" + "data"

L.get

⇒

y

⇒

Extra repairs

⇒
Unrestricted function

\[ L \ x \]

Repair "code" + "data"

\[ L \ x^I \]

Extra repairs
Unrestricted function

\[ \text{L} \ x \Rightarrow y \]

Repair "code" + "data"

\[ \text{L} \ x^I \leftarrow y^I \]

Extra repairs

\[ \text{L} \ x^I \Rightarrow y^I \]
Bidirectional Evaluation

Sketch-n-Sketch

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

let headerRow =
  let styles = [padding] in
  Html.tr (List.map (Html.th styles []) headers)
in
let stateRows =
  let colors = ["yellow", "white"] in
  let drawRow i row =
    let color = List.nth colors (mod i 2) in
    let columns = List.map
      (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

Filter

element.style {
  padding: 3px;
  background-color: yellow;
}

td[Attributes Style]yellowgreen
-webkit-user-modify: read-write-plaintext-only

yellowgreen

-lightgoldenrodyellow

-yellow
Bidirectional Evaluation

Sketch-n-Sketch

Elm-like PL + HTML GUI

\( e \Rightarrow v \quad \Rightarrow v' \Rightarrow e' \Rightarrow v'' \)
Bidirectional Evaluation

\[ e \xrightarrow{\text{eval}} v \]

\[ e^l \xleftarrow{\text{uneval}} v^l \]
Bidirectional Evaluation

\[(E \vdash e) \Rightarrow \mathbf{v}\]
\[(E' \vdash e') \Leftarrow \mathbf{v}'\]

Environment-style simpler than substitution-style
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

\[ E ::= – \mid (E, x \mapsto v) \]

\[
(E \vdash e) \Rightarrow v
\]

\[
(E' \vdash e') \Leftarrow v'
\]
\[
e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots
\]

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

\[
E ::= \_ \mid (E, x \mapsto v)
\]

\[
(E \vdash e) \Rightarrow v \Downarrow
\]

\[
(E' \vdash e') \Leftarrow v'
\]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 \, e_2 \mid \ldots \]

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

\[ E ::= – \mid (E, x \mapsto v) \]

\[(E \vdash e) \Rightarrow v \]

\[(E' \vdash e') \Leftarrow v' \]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 \, e_2 \mid \ldots \]

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[
\begin{align*}
(\_ & \vdash 2018) \Rightarrow 2018 \\
& \downarrow \Downarrow \\
& 2019
\end{align*}
\]
\[ e ::= \textcolor{red}{c} \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 \cdot e_2 \mid \ldots \]

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

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

\[ (\textcolor{red}{E} \vdash e) \Rightarrow v \]

\[ (E' \vdash e') \Leftarrow v' \]

\[ (\textcolor{red}{-} \vdash 2018) \Rightarrow 2018 \]

\[ (\textcolor{red}{-} \vdash 2019) \Leftarrow 2019 \]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]
\[ v ::= c \mid [E] \lambda x.e \mid (v_1, v_2) \]
\[ E ::= \_ \mid (E, x \mapsto v) \]

\[ (E \vdash e) \Rightarrow v \]
\[ \Downarrow \]

\[ (E' \vdash e') \Leftarrow v' \]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]
\[ v ::= c \mid [E] \lambda x.e \mid (v_1, v_2) \]
\[ E ::= \bot \mid (E, x \mapsto v) \]

\[
(- \vdash \text{let } x = 2018 \text{ in } x) \Rightarrow 2018
\]

\[
\Rightarrow \Downarrow 2019
\]
\( e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \)
\( v ::= c \mid [E] \lambda x. e \mid (v_1, v_2) \)
\( E ::= \vdash \mid (E, x \mapsto v) \)

\[
(x \mapsto 2018) \vdash x \Rightarrow 2018
\]

\[
(x \mapsto 2018) \vdash x \Rightarrow 2018
\]

\[
2019
\]
\[
e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid ...
\]

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

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

\[
((x \mapsto 2018) \vdash x) \Rightarrow 2018
\]

\[
((x \mapsto 2019) \vdash x) \Leftarrow 2019
\]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

\[ E ::= – \mid (E, x \mapsto v) \]

\[
\begin{align*}
(– \vdash \text{let } x = 2018 \text{ in } x) & \Rightarrow 2018 \\
(– \vdash \text{let } x = 2019 \text{ in } x) & \Leftarrow 2019
\end{align*}
\]

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

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[ (E \vdash e) \Rightarrow v \]

\[ (E' \vdash e') \Leftarrow v' \]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[ (\_ \vdash e) \Rightarrow v \]

\[ (E' \vdash e') \Leftrightarrow v' \]

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

\[ \downarrow \]

\[ (19, 18) \]
\[
\begin{align*}
e &::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \\
v &::= c \mid [E] \lambda x. e \mid (v_1, v_2) \\
E &::= \vdash \mid (E, x \mapsto v)
\end{align*}
\]

\[
(((x \mapsto 18) \vdash (x, x)) \Rightarrow (18, 18) \\
\]

\[
(19, 18)
\]
\[
\begin{align*}
e & ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \\
v & ::= c \mid [E] \lambda x.e \mid (v_1, v_2) \\
E & ::= \_ \mid (E, x \mapsto v)
\end{align*}
\]

\[
\begin{align*}
((x \mapsto 18) \vdash (x, x)) & \Rightarrow (18, 18) \\
((x \mapsto 18\text{?} 19\text{?}) \vdash (x, x)) & \Leftarrow (19, 18)
\end{align*}
\]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

\[ E ::= – \mid (E, x\mapsto v) \]

\[
\begin{align*}
((\textcolor{green}{x \mapsto 18}) \vdash (x, x)) & \Rightarrow (18, 18) \\
((\textcolor{green}{x \mapsto 18? 19? 20?}) \vdash (x, x)) & \Leftarrow (19, 20)
\end{align*}
\]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

\[ E ::= \ldots \mid (E, x \mapsto v) \]

**Merge:** \( E_1 \oplus_{\text{conservative}} E_2 \)

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

Enables a PutGet theorem
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

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

\[
\begin{align*}
(E \vdash e) & \Rightarrow v \\
\Leftrightarrow (E' \vdash e') & \Leftarrow v'
\end{align*}
\]

**Merge:** \( E_1 \oplus_{\text{conservative}} E_2 \)

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

Enables a PutGet theorem

**Merge:** \( E_1 \oplus_{\text{optimistic}} E_2 \)

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

Enables many desirable interactions
\[
\begin{align*}
e &::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \\
v &::= c \mid [E] \lambda x.e \mid (v_1, v_2) \\
E &::= \_ \mid (E, x \mapsto v)
\end{align*}
\]

\[
\begin{align*}
((x \mapsto 18) \vdash (x, x)) &\Rightarrow (18, 18) \\
((x \mapsto 18?) \vdash (x, x)) &\Leftrightarrow (19, 20)
\end{align*}
\]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]

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

\[ E ::= – \mid (E, x\mapsto v) \]

\[
(\mathcal{E} \vdash e) \Rightarrow \mathbf{v} \\
\mathcal{E}' \vdash e' \Leftarrow \mathbf{v}'
\]

\[
(– \vdash \text{let } x = 18 \text{ in } (x, x)) \Rightarrow (18, 18)
\]

\[
(– \vdash \text{let } x = \begin{cases} 18? \\ 19? \\ 20? \end{cases} \text{ in } (x, x)) \Leftarrow (19, 20)
\]

Ask user to choose
\[
e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots
\]
\[
v ::= c \mid [E] \lambda x. e \mid (v_1, v_2)
\]
\[
E ::= \varepsilon \mid (E, x \mapsto v)
\]

\[
(E \vdash e) \Rightarrow v
\]

\[
(E' \vdash e') \Leftarrow v'
\]
\[
\begin{align*}
e &::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \\
v &::= c \mid [E] \lambda x. e \mid (v_1, v_2) \\
E &::= \_ \mid (E, x \mapsto v)
\end{align*}
\]

\[
(- \vdash \text{let } f = \lambda x. x + + "18"
\text{ in } f "B"
) \Rightarrow "B 18"
\]

\[
\Rightarrow "Bx 19"
\]
\[
e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots
\]

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

\[
E ::= \_ \mid (E, x \mapsto v)
\]

\[
((f \mapsto \lambda x. x ++ "18") \vdash f "B ") \Rightarrow "B 18"
\]

\[
"B x 19"
\]
e ::= c | \( \lambda x . e \) | x | (e_1, e_2) | e_1 e_2 | ...

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

E ::= \epsilon | (E, x \mapsto v)

((x \mapsto "B ") \vdash x ++ "18") \Rightarrow "B 18"

"Bx 19"
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 \cdot e_2 \mid \ldots \]

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[ (E \vdash e) \Rightarrow v \]

\[ (E' \vdash e') \Leftarrow v' \]

\[ ((x \mapsto \text{"B "}) \vdash x ++ \text{"18"}) \Rightarrow \text{"B 18"} \]

\[ ((x \mapsto \text{"Bx"}) \vdash x ++ \text{"19"}) \Leftarrow \text{"Bx 19"} \]

"data" "code"
\[
\begin{align*}
e &::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \\
v &::= c \mid [E] \lambda x.e \mid (v_1, v_2) \\
E &::= \vdash \mid (E, x\!\mapsto\!v) \\
\end{align*}
\]

\[
\begin{align*}
((f \mapsto \lambda x. x \text{ ++ "18"}) \vdash f \text{ "B "}) &\Rightarrow \text{ "B 18"}

((f \mapsto \lambda x. x \text{ ++ "19"}) \vdash f \text{ "Bx "}) &\Leftarrow \text{ "Bx 19"}
\end{align*}
\]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \]
\[ v ::= c \mid [E] \lambda x.e \mid (v_1, v_2) \]
\[ E ::= \_ \mid (E, x \mapsto v) \]

\[
(\_ \vdash e) \Rightarrow v
\]
\[
(E \vdash e') \Leftarrow v'
\]

\[
\text{let } f = \lambda x. x \mathbin{++} "18" \text{ in } f "B"
\]
\[
\Rightarrow "B\ 18"
\]

\[
\text{let } f = \lambda x. x \mathbin{++} "19" \text{ in } f "B\ x"
\]
\[
\Leftarrow "B\ x\ 19"
\]
\[ e ::= c \ | \ \lambda x.e \ | \ x \ | \ (e_1, e_2) \ | \ e_1 e_2 \ | \ \ldots \]
\( e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \mid \ldots \)
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 \ e_2 \mid \ldots \]

\[ (E \vdash e) \Rightarrow \nu \]

\[ (E' \vdash e') \Leftarrow \nu' \]

\[ (- \vdash "B \ ++ \ "18") \Rightarrow "B \ 18" \]

\[ (- \vdash "Bx \ ++ \ "19") \Leftarrow "Bx \ 19" \]

Primitive lens: \((++)_\text{get}, (++)_\text{put}\)
Libraries and users can define custom lenses

```
Primitive lens: 
```

```
Primitive lens: ((++)_get , (++)_put)
```

```
e ::= c | \lambda x.e | x | (e_1, e_2) | e_1 e_2 | ... | appLens (e_{get}, e_{put}) e_x
```

```
(E ⊢ e) ⇒ v
⇒
(E' ⊢ e') ⇐ v'
```

```
(¬ ⊬ "B " ++ "18") ⇒ "B 18"

(¬ ⊬ "Bx " ++ "19") ⇐ "Bx 19"
```

```
Libraries and users can define custom lenses
```
Bidirectional Programs are Lenses
Bidirectional Programs

with

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

Variety of "bidirectional docs"
### Example
<table>
<thead>
<tr>
<th>Example</th>
<th>LOC</th>
<th>Eval</th>
<th>#Upd</th>
<th>#Sol</th>
<th>Average Upd</th>
</tr>
</thead>
<tbody>
<tr>
<td>States Table A*</td>
<td>37</td>
<td>304±20</td>
<td>11</td>
<td>1.18</td>
<td>85±20</td>
</tr>
<tr>
<td>States Table B*</td>
<td>128</td>
<td>224±70</td>
<td>11</td>
<td>1.18</td>
<td>32±88</td>
</tr>
<tr>
<td>Recipe*</td>
<td>193</td>
<td>1455±80</td>
<td>17</td>
<td>1.05</td>
<td>1328±500</td>
</tr>
<tr>
<td>Budgetting</td>
<td>37</td>
<td>282±110</td>
<td>10</td>
<td>1</td>
<td>9±2</td>
</tr>
<tr>
<td>MVC</td>
<td>71</td>
<td>720±50</td>
<td>10</td>
<td>1</td>
<td>289±80</td>
</tr>
<tr>
<td>Linked-Text</td>
<td>91</td>
<td>855±40</td>
<td>5</td>
<td>1.2</td>
<td>2025±200</td>
</tr>
<tr>
<td>Markdown</td>
<td>128</td>
<td>1179±110</td>
<td>6</td>
<td>1</td>
<td>1607±200</td>
</tr>
<tr>
<td>Dixit</td>
<td>130</td>
<td>705±40</td>
<td>15</td>
<td>1</td>
<td>417±1500</td>
</tr>
<tr>
<td>Translation</td>
<td>122</td>
<td>357±20</td>
<td>8</td>
<td>2</td>
<td>415±200</td>
</tr>
<tr>
<td>LATEX in HTML</td>
<td>534</td>
<td>1648±200</td>
<td>6</td>
<td>1</td>
<td>943±1000</td>
</tr>
<tr>
<td><strong>Total / Average</strong></td>
<td><strong>1469</strong></td>
<td><strong>833±400</strong></td>
<td><strong>92</strong></td>
<td><strong>1.18</strong></td>
<td><strong>(723±900) (70×)</strong></td>
</tr>
</tbody>
</table>

Variety of "bidirectional docs"

Variety of updates with relatively little ambiguity
<table>
<thead>
<tr>
<th>Example</th>
<th>LOC</th>
<th>Eval</th>
<th>#Upd</th>
<th>#Sol</th>
<th>Average Upd</th>
</tr>
</thead>
<tbody>
<tr>
<td>States Table A*</td>
<td>37</td>
<td>304±20</td>
<td>11</td>
<td>1.18</td>
<td>85±20</td>
</tr>
<tr>
<td>States Table B*</td>
<td>120</td>
<td>720±50</td>
<td>10</td>
<td>1.2</td>
<td>289±80</td>
</tr>
<tr>
<td>Recipe*</td>
<td>193</td>
<td>1455±80</td>
<td>17</td>
<td>1.05</td>
<td>1328±500</td>
</tr>
<tr>
<td>Budgetting</td>
<td>37</td>
<td>288±140</td>
<td>11</td>
<td>1.18</td>
<td>85±20</td>
</tr>
<tr>
<td>MVC</td>
<td>71</td>
<td>720±50</td>
<td>10</td>
<td>1.2</td>
<td>2025±200</td>
</tr>
<tr>
<td>Linked-Text</td>
<td>91</td>
<td>855±40</td>
<td>5</td>
<td>1.2</td>
<td>2025±200</td>
</tr>
<tr>
<td>Markdown</td>
<td>128</td>
<td>1179±110</td>
<td>6</td>
<td>1</td>
<td>1607±200</td>
</tr>
<tr>
<td>Dixit</td>
<td>130</td>
<td>705±40</td>
<td>15</td>
<td>1</td>
<td>417±1500</td>
</tr>
<tr>
<td>Translation</td>
<td>122</td>
<td>755±40</td>
<td>10</td>
<td>1</td>
<td>415±200</td>
</tr>
<tr>
<td>\LaTeX \ in HTML</td>
<td>534</td>
<td>1648±200</td>
<td>6</td>
<td>1</td>
<td>943±1000</td>
</tr>
<tr>
<td><strong>Total / Average</strong></td>
<td>1469</td>
<td>833±400</td>
<td>92</td>
<td>1.18</td>
<td>(723±900) (70×)</td>
</tr>
</tbody>
</table>

Variety of "bidirectional docs"

Variety of updates with relatively little ambiguity

Uneval time $\approx$ Eval time (with optimizations)
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 $\lambda$-calculus eliminates function calls $(e_1, e_2)$
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]

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]

Hybrid Text-GUI Code Editors
[ICSE 2018, VL/HCC 2020]

Output-Directed Programming
[UIST 2016, UIST 2019]

PL + UI
length : List a -> Int
length =
  ??

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

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 [] == [])

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)

max n m =
  case n of
    Zero     -> m
    Succ n'  -> case m of
                  Zero     -> n
                  Succ m'  -> Succ (max n' m')
Synthesizer
Myth [Osera et al. PLDI 2015, POPL 2016]
length : List a -> Int
length =
  ??

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

- **Type**
- **Trace-Complete Examples**
- **Sketch**
- **Program**

**Guess-and-Check**

$E \vdash e \models ex$

**Example Refinement**

**Branching**
Example Satisfaction in Myth

\[ E \vdash e \models \text{ex} \]

**Eval**

\[ E \vdash e \Rightarrow v \]

**Check**

\[ v \models \text{ex} \]

\[ v ::= \ldots \mid \lambda x.e \]

\[ \text{ex} ::= \ldots \mid \{ v_{\text{in}} \rightarrow \text{ex}_{\text{out}} \} \]

Input-output examples for functions
How to **Eval** and **Check** Sketches ??

\[
E \vdash e \Rightarrow v \\
+ \\
\text{Check} \\
v \models \text{ex}
\]
E ⊢ e ⊨ ex

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

Partially evaluate sketch to result $r$ (value or "paused")
How to **Eval** and **Check** Sketches ??

- **Live Eval**
  \[ E \leftarrow e \Rightarrow r \]
  Partially evaluate sketch to result \( r \) (value or "paused")

- **Live Uneval**
  \[ r \leftarrow ex \rightarrow \neg X \]
  Constraints \( X \) s.t. \( r \) will resume and satisfy \( ex \)
Live Bidirectional Evaluation

Live Eval:

\[ E \leftarrow e \Rightarrow r \]

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

Live Uneval:

\[ r \leftrightarrow ex \rightarrow X \]

Constraints \( X \) s.t. \( r \) will resume and satisfy \( ex \)

>>> Skip Examples >>>
Live Eval

\[ E \leftarrow e \Rightarrow r \]

Partially evaluate sketch to result \( r \) (value or "paused")
Holes typically represented as raise "Not yet implemented"
Expressions

<table>
<thead>
<tr>
<th>Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paused</td>
<td></td>
</tr>
</tbody>
</table>

Live Eval

\[ e \Rightarrow r \]

Partially evaluate "around" holes

\[
(??, 2019 + 1) \Rightarrow (??, 2020) \\
snd (??, 2019 + 1) \Rightarrow 2020 \\
(snd ??, 2019 + 1) \Rightarrow (snd ??, 2020)
\]
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
Example constraints required by Myth
\[ r \triangleq \text{ex} \rightarrow X \]

\[
\begin{align*}
  r &::= \ldots \mid \lambda x. e \mid [E] \ ??_h \mid \text{snd } r \mid r_1 + r_2 \mid \ldots \\
  \text{ex} &::= \ldots \mid \{ v_{\text{in}} \rightarrow \text{ex}_{\text{out}} \} \\
  X &::= \ldots \mid X, (E \vdash ??_h \models \text{ex})
\end{align*}
\]

\[
\begin{align*}
  2020 &\triangleq 2020 \rightarrow \emptyset \\
  [E] \ ??_1 &\triangleq 2020 \rightarrow (E \vdash ??_1 \models 2020) \\
  4 + [E] \ ??_2 &\triangleq 2020 \rightarrow (E \vdash ??_2 \models 2016) \\
  \text{ite } ([E] \ ??_3) \ 2016 \ 2020 &\triangleq 2020 \rightarrow (E \vdash ??_3 \models \text{False})
\end{align*}
\]
Live Eval

\[ e \Rightarrow r \]

Live Uneval

\[ r \leftrightarrow ex \rightarrow X \]

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

\[ ex ::= \ldots | \{ v_{in} \rightarrow ex_{out} \} \]

\[ X ::= \ldots | X, (E \vdash ??h \models ex) \]
Live Bidirectional Evaluation

Live Eval

\[ E \leftarrow e \Rightarrow r \]

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

Live Uneval

\[ r \leftarrow ex \leftarrow X \]

Constraints \( X \) s.t. \( r \) will resume and satisfy \( ex \)
Example Refinement

\[ E \vdash e \Rightarrow r \leftarrow ex \rightarrow X \]
<table>
<thead>
<tr>
<th><strong>SMYT</strong></th>
<th><strong>Experiment</strong></th>
<th><strong>1</strong></th>
<th><strong>2a</strong></th>
<th><strong>2b</strong></th>
<th><strong>3a</strong></th>
<th><strong>3b</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sketch / Objective</strong></td>
<td><strong>None / Top-1</strong></td>
<td><strong>None / Top-1</strong></td>
<td><strong>Base Case / Top-1-R</strong></td>
<td><strong>Base Case / Top-1-R</strong></td>
<td><strong>Base Case / Top-1-R</strong></td>
<td><strong>Base Case / Top-1-R</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Expert Time</strong></td>
<td><strong>Expert Random (50%, 90%)</strong></td>
<td><strong>Random (50%, 90%)</strong></td>
<td><strong>Expert Random (50%, 90%)</strong></td>
<td><strong>Random (50%, 90%)</strong></td>
<td><strong>Random (50%, 90%)</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>bool_band</td>
<td>4 0.004</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>4</td>
</tr>
<tr>
<td>bool_bor</td>
<td>4 0.003</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>3</td>
</tr>
<tr>
<td>bool_impl</td>
<td>4 0.004</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool_neg</td>
<td>2 0.001</td>
<td>3 (100%)</td>
<td>(2,2)</td>
<td>2 (100%)</td>
<td>(2,2)</td>
<td>2</td>
</tr>
<tr>
<td>bool_xor</td>
<td>4 0.009</td>
<td>4 (100%)</td>
<td>(4,4)</td>
<td>4 (100%)</td>
<td>(4,4)</td>
<td>4</td>
</tr>
<tr>
<td>list_append</td>
<td>6 0.008</td>
<td>6 (63%)</td>
<td>(3,4)</td>
<td>6 (63%)</td>
<td>(3,4)</td>
<td>1+1 (33%)</td>
</tr>
<tr>
<td>list_compress</td>
<td>13 timeout</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>list_concat</td>
<td>6 0.010</td>
<td>3 (50%)</td>
<td>(2,4)</td>
<td>3 (50%)</td>
<td>(2,4)</td>
<td>incorrect (1,3,1+5)</td>
</tr>
<tr>
<td>list_drop</td>
<td>11 0.092</td>
<td>5 (45%)</td>
<td>(6,9)</td>
<td>5 (45%)</td>
<td>(6,9)</td>
<td>1+2 (27%)</td>
</tr>
<tr>
<td>list_even_parity</td>
<td>7 overspec</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>list_filter</td>
<td>9* 0.144</td>
<td>5 (56%)</td>
<td>(6,9)</td>
<td>5 (56%)</td>
<td>(6,9)</td>
<td>1+2 (27%)</td>
</tr>
<tr>
<td>list_fold</td>
<td>9 0.838</td>
<td>3 (33%)</td>
<td>a</td>
<td>3 (33%)</td>
<td>a</td>
<td>1+3 (44%)</td>
</tr>
<tr>
<td>list_hd</td>
<td>3 0.003</td>
<td>2 (67%)</td>
<td>(2,3)</td>
<td>2 (67%)</td>
<td>(2,3)</td>
<td>2</td>
</tr>
<tr>
<td>list_incl</td>
<td>4 0.018</td>
<td>2 (50%)</td>
<td>(2,2)</td>
<td>2 (50%)</td>
<td>(2,2)</td>
<td>2</td>
</tr>
<tr>
<td>list_last</td>
<td>6 0.007</td>
<td>4 (67%)</td>
<td>(3,9)</td>
<td>4 (67%)</td>
<td>(3,9)</td>
<td>1+2 (50%)</td>
</tr>
<tr>
<td>list_length</td>
<td>3 0.002</td>
<td>3 (100%)</td>
<td>(2,4)</td>
<td>3 (100%)</td>
<td>(2,4)</td>
<td>1+1 (67%)</td>
</tr>
<tr>
<td>list_map</td>
<td>8 0.049</td>
<td>4 (50%)</td>
<td>a</td>
<td>4 (50%)</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>list_nth</td>
<td>13 0.124</td>
<td>5 (38%)</td>
<td>(7,14)</td>
<td>5 (38%)</td>
<td>(7,14)</td>
<td>1+2 (23%)</td>
</tr>
<tr>
<td>list_pairwise_swap</td>
<td>7 0.634</td>
<td>5 (71%)</td>
<td>timeout</td>
<td>overspec</td>
<td>timeout</td>
<td></td>
</tr>
<tr>
<td>list_rev_append</td>
<td>5 0.107</td>
<td>3 (60%)</td>
<td>(5,8)</td>
<td>3 (60%)</td>
<td>(5,8)</td>
<td>1+2 (60%)</td>
</tr>
<tr>
<td>list_rev_fold</td>
<td>5 0.035</td>
<td>2 (40%)</td>
<td>(2,4)</td>
<td>2 (40%)</td>
<td>(2,4)</td>
<td>a</td>
</tr>
<tr>
<td>list_rev_snoc</td>
<td>5 0.010</td>
<td>3 (60%)</td>
<td>(3,6)</td>
<td>3 (60%)</td>
<td>(3,6)</td>
<td>1+1 (40%)</td>
</tr>
<tr>
<td>list_rev_tailcall</td>
<td>8 0.008</td>
<td>3 (38%)</td>
<td>(3,4)</td>
<td>3 (38%)</td>
<td>(3,4)</td>
<td>1+1 (25%)</td>
</tr>
<tr>
<td>list_snoc</td>
<td>8 0.012</td>
<td>3 (38%)</td>
<td>(3,4)</td>
<td>3 (38%)</td>
<td>(3,4)</td>
<td>1+1 (25%)</td>
</tr>
<tr>
<td>list_sorted_insert</td>
<td>7 0.015</td>
<td>3 (43%)</td>
<td>(3,6)</td>
<td>3 (43%)</td>
<td>(3,6)</td>
<td>1+1 (29%)</td>
</tr>
<tr>
<td>list_sorted_insert</td>
<td>12 2.902</td>
<td>7 (38%)</td>
<td>timeout</td>
<td>overspec</td>
<td>timeout</td>
<td></td>
</tr>
<tr>
<td>list_stutter</td>
<td>3 0.003</td>
<td>2 (67%)</td>
<td>(3,3)</td>
<td>2 (67%)</td>
<td>(3,3)</td>
<td>1+1 (67%)</td>
</tr>
<tr>
<td>list_sum</td>
<td>3 0.029</td>
<td>2 (67%)</td>
<td>(2,2)</td>
<td>2 (67%)</td>
<td>(2,2)</td>
<td>2</td>
</tr>
<tr>
<td>list_take</td>
<td>12 0.065</td>
<td>5 (42%)</td>
<td>(6,9)</td>
<td>5 (42%)</td>
<td>(6,9)</td>
<td>1+3 (33%)</td>
</tr>
<tr>
<td>list_ttl</td>
<td>3 0.002</td>
<td>2 (67%)</td>
<td>(2,3)</td>
<td>2 (67%)</td>
<td>(2,3)</td>
<td>2</td>
</tr>
<tr>
<td>nat_add</td>
<td>9 0.006</td>
<td>4 (44%)</td>
<td>(5,6)</td>
<td>4 (44%)</td>
<td>(5,6)</td>
<td>1+1 (22%)</td>
</tr>
<tr>
<td>nat_iseven</td>
<td>4 0.003</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>3 (75%)</td>
<td>(4,4)</td>
<td>1+2 (75%)</td>
</tr>
<tr>
<td>nat_max</td>
<td>9 0.041</td>
<td>9 (100%)</td>
<td>(8,12)</td>
<td>9 (100%)</td>
<td>(8,12)</td>
<td>1+4 (56%)</td>
</tr>
<tr>
<td>nat_pred</td>
<td>3 0.001</td>
<td>2 (67%)</td>
<td>(2,3)</td>
<td>2 (67%)</td>
<td>(2,3)</td>
<td>2</td>
</tr>
<tr>
<td>tree_bininsert</td>
<td>20 timeout</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>tree_collect_leaves</td>
<td>6 0.074</td>
<td>3 (50%)</td>
<td>(3,4)</td>
<td>3 (50%)</td>
<td>(3,4)</td>
<td>1+2 (50%)</td>
</tr>
<tr>
<td>tree_count_leaves</td>
<td>7 2.660</td>
<td>3 (43%)</td>
<td>(4,3)</td>
<td>timeout</td>
<td>timeout</td>
<td></td>
</tr>
<tr>
<td>tree_count_nodes</td>
<td>6 0.351</td>
<td>3 (50%)</td>
<td>timeout</td>
<td>3 (50%)</td>
<td>timeout</td>
<td></td>
</tr>
<tr>
<td>tree_inorder</td>
<td>5 0.123</td>
<td>4 (80%)</td>
<td>(3,4)</td>
<td>4 (80%)</td>
<td>(3,4)</td>
<td>1+2 (60%)</td>
</tr>
<tr>
<td>tree_map</td>
<td>7 0.061</td>
<td>4 (57%)</td>
<td>a</td>
<td>4 (57%)</td>
<td>a</td>
<td>1+3 (57%)</td>
</tr>
<tr>
<td>tree_nodes_at_level</td>
<td>11 timeout</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>tree_postorder</td>
<td>20 timeout</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>tree_preorder</td>
<td>5 0.153</td>
<td>3 (60%)</td>
<td>(3,4)</td>
<td>3 (60%)</td>
<td>(3,4)</td>
<td>1+2 (60%)</td>
</tr>
<tr>
<td><strong>Averages</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>61%*</td>
<td>46%</td>
</tr>
</tbody>
</table>
Compared to prior state-of-the-art:

Trace-complete examples (without a sketch)
**Compared to prior state-of-the-art:**

*Non*-trace-complete examples reduce specification size
Compared to prior state-of-the-art:

**Non**-trace-complete examples reduce specification size

**Sketches** reduce specification size
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
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]

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]

Hybrid Text-GUI Code Editors
[ICSE 2018, VL/HCC 2020]

Output-Directed Programming
[UIST 2016, UIST 2019]
Output-Directed Programming

Sketch-n-Sketch
Output-Directed Programming

Sketch-n-Sketch

Elm-like PL + SVG GUI

\[ e_0 \Rightarrow v_0 \Rightarrow e_1 \Rightarrow v_1 \Rightarrow e_2 \Rightarrow v_2 \]

\[ e_0' \Leftarrow v_0' \Leftarrow e_1' \Leftarrow v_1' \Leftarrow e_2' \]
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]

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]

Hybrid Text-GUI Code Editors
[ICSE 2018, VL/HCC 2020]

Output-Directed Programming
[UIST 2016, UIST 2019]

PL + UI

>>> Skip >>>
Hybrid Text-GUI Code Editors
(func arg)
Show AST on Text
(func arg)
# Traditional Refactoring UI

<table>
<thead>
<tr>
<th>Text Select</th>
<th>Right-Click Menu</th>
<th>Select Arguments</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>(def image1</td>
<td><img src="image" alt="Code Tools" /></td>
<td><img src="image" alt="Move Definition" /></td>
<td><img src="image" alt="Move Definition" /></td>
</tr>
<tr>
<td>(let [width</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(let [x y]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[100 100]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(image &quot;lightgrey&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Text-Select**
Text Select + Long Menu

Traditional Refactoring UI

Text Select

(def image1
(let [width height]
(let [x y [100 100]]
(image "lightgrey")

Right-Click Menu

Select Arguments

Defaults

Move Definition

Requirements
- Select one or more variable definitions and one target position (i.e. whitespace) (Satisfied)

Code Updates
Move width and height
## Traditional Refactoring UI

<table>
<thead>
<tr>
<th>Text Select</th>
<th>Right-Click Menu</th>
<th>Select Arguments</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(def image1 (let [width height] (let [x y] [100 100] (image &quot;lightgrey&quot;)))</code></td>
<td><img src="image" alt="Code Tools" /></td>
<td><img src="image" alt="Move Definition" /></td>
<td><img src="image" alt="Code Updates" /></td>
</tr>
</tbody>
</table>

- **Text-Select** + **Long Menu** + **Configuration Choices**
Traditional Refactoring UI

Deuce Refactoring UI

Text-Select + Long Menu + Configuration Choices

Structure Select

Short Menu

Defaults

Code Tools
Move Definition
Move width and height
Duplicate Definition

Move Definition
Requirements
- Select one or more variable definitions and one target position (i.e. whitespace) (Satisfied)

Code Updates
Move width and height
Traditional Refactoring UI

Text Select

(def image1
  (let [width height]
    (let [x y [100 100]]
      (image "lightgrey"))

Right-Click Menu

Select Arguments

Defaults

Move Definition

Requirements
- Select one or more variable definitions and one target position (i.e. whitespace) (Satisfied)

Code Updates

Move width and height

Text-Select + Long Menu + Configuration Choices

Deuce Refactoring UI

Structure Select

(def image1
  (let [width height]
    (let [x y [100 100]]
      (image "lightgrey"))

Short Menu

Code Tools
- Move Definition
- Duplicate Definition

Defaults

Move width and height

Structural Multi-Select + Context-Sensitive Transforms

135
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]

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]

Hybrid Text-GUI Code Editors
[ICSE 2018, VL/HCC 2020]

Output-Directed Programming
[UIST 2016, UIST 2019]
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")
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

PLATEAU 2020
11th annual workshop on the intersection of HCI and PL
Online Edition! November 2020
Co-located with CHICAGO SPLASH 2020

Human Aspects of Types and Reasoning Assistants
HATRA 2020

NO.147
Self-supporting, Extensible Programming Languages and Environments for Exploratory, Live Software Development
User Interfaces for Programming

Session 5-A: Viz
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

Session 7B: Live Programming and Synthesis

Focused Live Programming with Loop Seeds
Sorin Lerner

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

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

2D Graphics
Documents
Presentations
Spreadsheets
Web Apps

Code

LaTeX
Scholarly
MARKDOWN

LaTeX

SQL

JS

...
Creative Coding

Intro to Variables

```javascript
noStroke();

// face
fill(255, 255, 0);
ellipse(200, 200, 100, 100);

// eyes
fill(255, 255, 0);
ellipse(150, 150, 40, 40);
ellipse(300, 140, 40, 40);

// mouth
fill(120, 0, 120);
ellipse(250, 200, 130, 130);
```

Function Parameters

```javascript
for (var i = 0; i < 10; i++) {
    var face = random(0, 360);
    fill(255, 255, 0);
    ellipse(face, face, 200, 200); // face
    fill(255, 255, 0);
    ellipse(face, face, 50, 50); // eyes
    ellipse(face, face, 50, 50); // eyes
    ellipse(face, face, 50, 50); // eyes
    fill(255, 0, 120);
    ellipse(face, face, 50, 50); // mouth
}
```
Creative Coding

p5.js

```javascript
// color palette
var colors = [
    "#ff0000", "#ff30f", "#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 = [];
```
Creative Coding (CMSC 11111)

p5.js

```
// color palette
var colors = [
   "#ff0000", "#feb30f", "#0a4f7", "#0a0000", "#ffffff"];

// set weights for each color
var weights = [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 = [];
```
Creative Coding

Crative Coding

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)

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.

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-sketch...

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

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"
"Everyday Coding"

Spreadsheet Cells i...

Existing GUI + Inline Formulas

Sketch-n-Script

Formula Pane

Install
2D Graphics
Documents
Presentations
Spreadsheets
Web Apps

...
<table>
<thead>
<tr>
<th>2D Graphics</th>
<th>Documents</th>
<th>Presentations</th>
<th>Spreadsheets</th>
<th>Web Apps</th>
</tr>
</thead>
</table>

...
Data and Visualization

- 2D Graphics
- Documents
- Presentations
- Spreadsheets
- Web Apps
- ...
- ...

---

159
Data and Visualization
Computational Notebooks
Code

- P
- LaTeX
- LaTeX
- SQL
- JS

2D Graphics
Documents
Presentations
Spreadsheets
Web Apps

GUls

- Ai
- W
- P
- X
- Google Web Designer

Text

...
Thanks!

Nick Collins  
Brian Hempel  
Justin Lubin  
Mikaël Mayer  
Cyrus Omar

Grace Lu  
Mitch Spradlin  
Jacob Albers
EXTRA SLIDES
Bidirectional Evaluation
\[
e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 \cdot e_2
\]
\[
v ::= c \mid [E] \lambda x.e \mid (v_1, v_2)
\]
\[
E ::= \_ \mid (E, x \mapsto v)
\]
\[
(E \vdash e) \Rightarrow v
\]
\[
(E' \vdash e') \Leftarrow v'
\]
\[
(E \vdash e_1 \cdot e_2) \Rightarrow v
\]
\[
\Leftarrow v'
\]
\[\begin{align*}
e & ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \\
v & ::= c \mid [E] \lambda x.e \mid (v_1, v_2) \\
E & ::= \_ \mid (E, x \mapsto v)
\end{align*}\]

\[(E \vdash e) \Rightarrow v\]

\[(E' \vdash e') \iff v'\]

\[(E \ldots \vdash f \ x) \Rightarrow v\]
\[
\begin{align*}
e & ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \\
v & ::= c \mid [E] \lambda x.e \mid (v_1, v_2) \\
E & ::= \_ \mid (E, x \mapsto v)
\end{align*}
\]

\[
((E, f \mapsto [E_f] \lambda y.e_f, x \mapsto v_x) \vdash f x) \Rightarrow v \quad \Downarrow \quad v'
\]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[
\begin{align*}
(E \vdash e) & \Rightarrow v \\
(E' \vdash e') & \Leftrightarrow v'
\end{align*}
\]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[ (E \vdash e) \Rightarrow v \]

\[ (E' \vdash e') \Leftarrow v' \]
\[
\begin{align*}
e &::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 \cdot e_2 \\
v &::= c \mid [E] \lambda x.e \mid (v_1, v_2) \\
E &::= \_ \mid (E, x \mapsto v)
\end{align*}
\]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]
\[ v ::= c \mid [E] \lambda x. e \mid (v_1, v_2) \]
\[ E ::= \_ \mid (E, x \mapsto v) \]

\[
\begin{align*}
((E_f, y \mapsto v_x) \vdash e_f) & \Rightarrow v \\
\end{align*}
\]

\[
\begin{align*}
((E, f \mapsto [E_f] \lambda y. e_f, x \mapsto v_x) \vdash f x) & \Rightarrow v \\
\end{align*}
\]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]
\[ v ::= c \mid [E] \lambda x. e \mid (v_1, v_2) \]
\[ E ::= \_ \mid (E, x \mapsto v) \]

\[
\begin{align*}
((E_f, y \mapsto v_x) \vdash e_f) & \Rightarrow v \\
\downarrow \\
((E'_f, y \mapsto v'_x) \vdash e'_f) & \leftarrow v'
\end{align*}
\]

\[
\begin{align*}
((E, f \mapsto [E_f] \lambda y. e_f, x \mapsto v_x) \vdash f x) & \Rightarrow v \\
\downarrow \\
((E, f \mapsto , x \mapsto ) \vdash f x) & \leftarrow v'
\end{align*}
\]
\[ e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]

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

\[ E ::= \_ \mid (E, x \mapsto v) \]

\[ (E \vdash e) \Rightarrow v \]

\[ (E', \vdash e') \Leftarrow v' \]

**New Code**
e ::= c | λx.e | x | (e₁, e₂) | e₁ e₂
v ::= c | [E] λx.e | (v₁, v₂)
E ::= – | (E, x↦v)

New Code
New Data
Key Idea

\[
(E \vdash e) \Rightarrow v
\]

\[
(E' \vdash e') \iff v'
\]

\[
((E_f, y \mapsto v_x) \vdash e_f) \Rightarrow v
\]

\[
((E'_f, y \mapsto v'_x) \vdash e'_f) \iff v'
\]

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

\[
((E, f \mapsto [E'_f] \, \lambda y . e'_f, x \mapsto v'_x) \vdash f \, x) \iff v'
\]

New Code

New Data
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]

\[
\begin{align*}
(\mathcal{E} \vdash e) & \Rightarrow v \\
(\mathcal{E}' \vdash e') & \Leftarrow v'
\end{align*}
\]
e ::= c | \lambda x.e | x | (e_1, e_2) | e_1 e_2 \\
| e_1 :: e_2 | [e_1, \ldots, e_n] \\
| \{ f_1 = e_1, \ldots, f_n = e_n \} \\
| let x = e_1 in e_2 \\
| if e_1 then e_2 else e_3 \\
| e_1 + e_2 | \neg e | \ldots \\
| dict_get e_1 e_2 | \ldots \\
| re_match e_1 e_2 | \ldots \\
| eval e \\

\[(E \vdash e) \Rightarrow v \]
\[(E' \vdash e') \iff v'\]
\[
\begin{align*}
e &::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 \ e_2 \\
e_1 &::= e_2 \mid [e_1, \ldots, e_n] \\
\{f_1 = e_1, \ldots, f_n = e_n\} &
\]
| let x = e_1 in e_2 
| if e_1 then e_2 else e_3 
| e_1 + e_2 \mid \neg e \mid \ldots 
| dict\_get e_1 e_2 \mid \ldots 
| re\_match e_1 e_2 \mid \ldots 
| eval \ e
\end{align*}
\]
\[
e ::= c \mid \lambda x. e \mid x \mid (e_1, e_2) \mid e_1 e_2
\]

- \(e_1 :: e_2\)  
- \([e_1, \ldots, e_n]\)
- \(\{f_1 = e_1, \ldots, f_n = e_n\}\)
- let \(x = e_1\) in \(e_2\)
- if \(e_1\) then \(e_2\) else \(e_3\)
- \(e_1 + e_2\)  
- \(\neg e\)
- \(\text{dict\_get} e_1 e_2\)
- \(\text{re\_match} e_1 e_2\)
- eval \(e\)

Structural updates through, e.g.,

\[\text{List.map} f' \text{x}s' \iff \text{vs}'\]
\[ e ::= c \mid \lambda x.e \mid x \mid (e_1, e_2) \mid e_1 e_2 \]

- \( e_1 ::= e_2 \mid [e_1, \ldots, e_n] \)
- \( \{f_1 = e_1, \ldots, f_n = e_n\} \)
- let \( x = e_1 \) in \( e_2 \)
- if \( e_1 \) then \( e_2 \) else \( e_3 \)
- \( e_1 + e_2 \mid \neg e \)
- dict_get \( e_1 e_2 \)
- re_match \( e_1 e_2 \)
- eval \( e \)

\[
(E \vdash e) \Rightarrow v \\
(E' \vdash e') \Leftrightarrow v'
\]

Structural updates through, e.g.,

\[ \text{List.map } f' \text{ xs'} \Leftrightarrow vs' \]

Domain-specific primitives?
e ::= c | λx.e | x | (e₁, e₂) | e₁ e₂

- e₁ :: e₂
- [e₁, ..., eₙ]
- {f₁ = e₁, ..., fₙ = eₙ}
- let x = e₁ in e₂
- if e₁ then e₂ else e₃
- e₁ + e₂ | ¬e | ...
- dict_get e₁ e₂ | ...
- re_match e₁ e₂ | ...
- eval e

Structural updates through, e.g.,
List.map f' xs' ⇐ vs'

Domain-specific primitives?

User-defined lenses for customization
e ::= ... | \text{appLens} \ (e_{\text{get}}, e_{\text{put}}) \ e_x

(E \vdash e) \Rightarrow v

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

(E \vdash \text{appLens} \ (f_{\text{get}}, f_{\text{put}}) \ e_x) \Rightarrow
e ::= \ldots \mid \text{appLens} (e_{\text{get}}, e_{\text{put}}) \ e_x

\begin{align*}
(E \vdash e) & \Rightarrow v \\
(E' \vdash e') & \leftrightarrow v'
\end{align*}

((E, x \mapsto v_x) \vdash \text{appLens} (f_{\text{get}}, f_{\text{put}}) \ x) \Rightarrow
\[
e ::= \ldots \mid \text{appLens} \left( e_{\text{get}}, e_{\text{put}} \right) e_x
\]

\[
(E \vdash e) \Rightarrow v \\
(E' \vdash e') \Leftrightarrow v'
\]

\[
(E \vdash f_{\text{get}} v_x) \Rightarrow
\]

\[
((E, x \mapsto v_x) \vdash \text{appLens} \left( f_{\text{get}}, f_{\text{put}} \right) x) \Rightarrow
\]
\( e ::= \ldots \mid \text{appLens (e}\_\text{get, e}\_\text{put}) e_x \)

\[
\begin{align*}
(E \vdash e) & \Rightarrow v \\
(E' \vdash e') & \Leftarrow v'
\end{align*}
\]

\[
\begin{align*}
(E \vdash f\_\text{get } v_x) & \Rightarrow v_y \\
((E, x \mapsto v_x) \vdash \text{appLens (f}\_\text{get, f}\_\text{put}) x) & \Rightarrow v_y
\end{align*}
\]
\( e ::= \ldots \mid \text{appLens} \ (e_{\text{get}}, \ e_{\text{put}}) \ e_x \)

\[
(E \vdash e) \Rightarrow v \\
(E' \vdash e') \Leftarrow v'
\]

\[
(E \vdash f_{\text{get}} \ v_x) \Rightarrow v_y
\]

\[
((E, x \mapsto v_x) \vdash \text{appLens} \ (f_{\text{get}}, \ f_{\text{put}}) \ x) \Rightarrow v_y \\
\Rightarrow v'_y
\]
\[ e ::= \ldots \mid \text{appLens} (e_{\text{get}}, e_{\text{put}}) e_x \]

\[(E \vdash e) \Rightarrow v \]

\[(E' \vdash e') \Leftarrow v' \]

\[
\begin{align*}
(E \vdash f_{\text{get}} v_x) & \Rightarrow v_y \\
(E \vdash f_{\text{put}} v_x v_y v_y') & \Rightarrow \\
((E, x \mapsto v_x) \vdash \text{appLens} (f_{\text{get}}, f_{\text{put}}) x) & \Rightarrow v_y \\
& \Leftarrow v_y'
\end{align*}
\]
\[ e ::= \ldots \mid \text{appLens}\ (e_{\text{get}}, e_{\text{put}})\ e_x \]

\[
\begin{align*}
(E \vdash e) & \Rightarrow v \\
(E' \vdash e') & \iff v'
\end{align*}
\]

\[
\begin{align*}
(E \vdash f_{\text{get}}\ v_x) & \Rightarrow v_y \\
(E \vdash f_{\text{put}}\ v_x\ v_y\ v_{y'}) & \Rightarrow v'_{x'}
\end{align*}
\]

\[
\begin{align*}
((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{\text{get}}, f_{\text{put}})\ x) & \Rightarrow v_y \\
 & \iff v_{y'}
\end{align*}
\]
\[ e ::= \ldots \mid \text{appLens}\ (e_{\text{get}}, e_{\text{put}})\ e_x \]

\[
\begin{align*}
(E \vdash f_{\text{get}}\ v_x) &\Rightarrow v_y \\
(E \vdash f_{\text{put}}\ v_x\ v_y\ v_y') &\Rightarrow v_x'
\end{align*}
\]

\[
\begin{align*}
((E, x \mapsto v_x) \vdash \text{appLens}\ (f_{\text{get}}, f_{\text{put}})\ x) &\Rightarrow v_y \\
((E, x \mapsto v_x') \vdash \text{appLens}\ (f_{\text{get}}, f_{\text{put}})\ x) &\Leftarrow v_y'
\end{align*}
\]
\[ e ::= \ldots \mid \text{appLens} \ (e_{\text{get}}, e_{\text{put}}) \ e_x \]

\[
\begin{align*}
(E \vdash f_{\text{get}} \ v_x) & \Rightarrow v_y \\
(E \vdash f_{\text{put}} \ v_x \ v_y \ v_y') & \Rightarrow v_x'
\end{align*}
\]

- PutGet ENFORCED BY RADAR

\[
\begin{align*}
((E, x \mapsto v_x) & \vdash \text{appLens} \ (f_{\text{get}}, f_{\text{put}}) \ x) \Rightarrow v_y \\
((E, x \mapsto v_x') & \vdash \text{appLens} \ (f_{\text{get}}, f_{\text{put}}) \ x) \Leftarrow v_y'
\end{align*}
\]
### Related Work

<table>
<thead>
<tr>
<th>Repair Algorithm</th>
<th>[Chugh et al. PLDI 2016]</th>
<th>[OOPSLA 2018]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation with tracing + constraint solving</td>
<td>Re-run evaluation &quot;in reverse&quot; only <em>as needed</em></td>
<td>Lens API for <em>[customization]</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Interface</th>
<th>[Chugh et al. PLDI 2016]</th>
<th>[OOPSLA 2018]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little $\lambda$-calculus $\Rightarrow$ SVG</td>
<td><strong>Elm</strong>-like language $\Rightarrow$ HTML</td>
<td><strong>User disambiguation</strong> via dropdown menu</td>
</tr>
<tr>
<td>Auto disambiguation via simple heuristics</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Live Programming by Example
Live Programming by Example

Our Goal:

Sketching with Example-Based Synthesis
- Escher
- Lambda²
- Myth
- Myth²

Sketching with Logic-Based Synthesis
- Sketch
- Leon
- Rosette
- Synquid
Program Sketching
e.g. [Sketch]

Logic-Based Synthesis
e.g. [Synquid]

Example-Based Synthesis
e.g. [Myth]
Examples from evaluating sketch

Solve multiple interdependent holes

Often requiring fewer user-provided examples
**Myth**

[Check]

\[
E e \Rightarrow v \quad v \models ex \\
\hline
\hline
e \models (E \vdash \bullet \models ex)
\]

[Check-Input-Output]

\[
e_f = \lambda x. e[\{v_i \rightarrow ex_i\}^{i \in [n]} / f] \\
\{ e_f v_i \Rightarrow v'_i \quad v'_i \models ex_i \}^{i \in [n]} \\
\hline
\hline
\text{fix } f (\lambda x. e) \models \{v_i \rightarrow ex_i\}^{i \in [n]}
\]

**Sketch-n-Myth**

[Live-Check]

\[
E e \Rightarrow r \quad r \Leftarrow ex + K \\
\hline
\hline
e \models (E \vdash \bullet \models ex) + K
\]

[Live-Check-Input-Output]

\[
(fix f (\lambda x. e)) v \Rightarrow r \quad r \Leftarrow ex + K \\
\hline
\hline
\text{fix } f (\lambda x. e) \Leftarrow \{v \rightarrow ex\} + K
\]
<table>
<thead>
<tr>
<th>Name</th>
<th>Experiment</th>
<th>Smyth</th>
<th>Leon</th>
<th>Synquid</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool_band</td>
<td>4</td>
<td>0.004</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>bool_bor</td>
<td>4</td>
<td>0.003</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>bool_impl</td>
<td>4</td>
<td>0.004</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>bool_neg</td>
<td>2</td>
<td>0.001</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>bool_xor</td>
<td>4</td>
<td>0.009</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_append</td>
<td>6</td>
<td>0.008</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_compress</td>
<td>13</td>
<td>timeout</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_concat</td>
<td>6</td>
<td>0.010</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_drop</td>
<td>11</td>
<td>0.092</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_evenparity</td>
<td>7</td>
<td>overspec</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_filter</td>
<td>9</td>
<td>0.144</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_fold</td>
<td>9</td>
<td>0.838</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_hd</td>
<td>3</td>
<td>0.003</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_inc</td>
<td>4</td>
<td>0.018</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_last</td>
<td>6</td>
<td>0.007</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_length</td>
<td>3</td>
<td>0.002</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_map</td>
<td>8</td>
<td>0.049</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_mth</td>
<td>13</td>
<td>0.124</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_pairwise_swap</td>
<td>7</td>
<td>0.634</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_rev_append</td>
<td>5</td>
<td>0.107</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_rev_fold</td>
<td>5</td>
<td>0.035</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_rev_snoc</td>
<td>5</td>
<td>0.010</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_rev_tailcall</td>
<td>8</td>
<td>0.008</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_snoc</td>
<td>8</td>
<td>0.012</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_sort_sorted_insert</td>
<td>7</td>
<td>0.015</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_sorted_insert</td>
<td>12</td>
<td>2.902</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_stutter</td>
<td>3</td>
<td>0.003</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_sum</td>
<td>3</td>
<td>0.029</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_take</td>
<td>12</td>
<td>0.065</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>list_tl</td>
<td>3</td>
<td>0.002</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>nat_add</td>
<td>9</td>
<td>0.006</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>nat_iseven</td>
<td>4</td>
<td>0.003</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>nat_isodd</td>
<td>9</td>
<td>0.041</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>nat_max</td>
<td>3</td>
<td>0.001</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>nat_pred</td>
<td>20</td>
<td>timeout</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_hinsert</td>
<td>6</td>
<td>0.074</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_collect_leaves</td>
<td>7</td>
<td>2.660</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_count_leaves</td>
<td>6</td>
<td>0.351</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_inorder</td>
<td>5</td>
<td>0.123</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_map</td>
<td>7</td>
<td>0.061</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_nodes_at_level</td>
<td>11</td>
<td>timeout</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_postorder</td>
<td>20</td>
<td>timeout</td>
<td>1</td>
<td>2a</td>
</tr>
<tr>
<td>tree_preorder</td>
<td>3</td>
<td>0.153</td>
<td>1</td>
<td>2a</td>
</tr>
</tbody>
</table>

Averages: 61%* 46%
Output-Directed Programming
Programming with:

Less Keyboard.

More Mouse.
Hybrid Text-GUI Code Editors
21 users

Structure Editors

Deuce UI

versus

Traditional UI

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

*Modest subjective preference for Deuce*

Observed

*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

Plain Text Editing

Hybrid Editors
- Barista
- Active Code Completion
- Greenfoot
- Code Bubbles
- Sketch-n-Sketch: Deuce

Structured Editing

Alternative UIs for Refactoring
- Refactoring w/ Synthesis
- Drag-n-Drop Refactoring
- Sketch-n-Sketch: Deuce

Automated Refactoring