Bimodal Tangible Functional Programming

Brian Hempel and Ravi Chugh

ECOOP
30 June 2022
Direct Manipulation UI
Why not D.M. for programming?
D.M. the AST...

Nodes-and-wires

The On-line Graphical Specification of Computer Procedures
W. Sutherland (1966)

Blocks

Scratch
Resnick et al. (2009)

...but those are **expressions** not output **values**
D.M. the values...

Programming by Demonstration (PBD)
You give a step-by-step demonstration of what you want the computer to do.

Pygmalion Smith (1975)

Have you ever used PBD?
- Domain-specific
- Rarely textual code
Bimodal Programming

ordinary code \{ \}

program's runtime data values
ALVIS Live!
Hundhausen & Brown (2007)

Array algorithms (for education)
Array algorithms (for education)
Very linear: have to manage time.
Tangible Functional Programming
Conal Elliott (2007)

Tangible Value (TV)

Canvas of tangible values

Inputs

Output

Non-linear

Pure functional programming (no state) complements non-linear editing because, without state, one need not manage time.
Goals: Non-linear + Bimodal + Synthesis

Direct Manipulation + Synthesis = Maniposynth
• **Goals**

• Demo

• Implementation

• Evaluation

• Future Work & Conclusion
• Goals

• Demo

• Implementation

• Evaluation

• Future Work & Conclusion
Tangible Values in Maniposynth

1 let-binding = 1 TV (Roughly)

```plaintext
4    int
5  let length3 = length [ 0; 0; 0 ] [@pos 405, 175]
6
```

Pattern

Expression

Value

length3 = length [ 0; 0; 0 ]
Demo: List Length

```ocaml
let int_list = [ 0; 0; 0 ]

let rec length list = match list with
| hd :: tail -> length tail in
  1 + length2
| [] -> 0

let length_int = length int_list
```
Demo: List Length

Drag to extract

Autocomplete to value

Assertions

Autocomplete to extract

Synthesis
• Goals

• Demo

• Implementation
  - Interpreter
  - Binding reordering
  - Synthesizer

• Evaluation

• Future Work & Conclusion
Architecture

Editor

Code File

Webserver

Browser

Code is “ground truth”

Server runs code and renders HTML

~8600 OCaml LOC

~2000 JS LOC

Browser polls for changes or tells server to do an action
Interpreter

• Adapted the interpreter from Camlboot [Courant et al. 2020]
  (Couldn’t just modify the standard OCaml tools because the OCaml compiler
  performs type erasure—can’t log the value when expression is at polymorphic type!)

• On each execution step, log:

  \((\text{exp/pat, call frame num, val, env})\)

• For live display, show value at that exp/pat with the current
  call frame num
Binding reordering

2D canvas is unordered, let-bindings in code are automatically reordered to bring items into scope.

Requirement: All names at the same “indentation level” must be unique.
Synthesizer

- No big ideas, just want it to work with
  (a) few examples,
  (b) no type annotations, and
  (c) produce quality results
even with the Pervasives functions in scope (e.g., addition, subtraction, etc).

- Type-directed, inspired by Myth (Osera and Zdancewic 2015)

- With a probabilistic context-free grammar (PCFG)

  \[
  \text{Expressions } e \quad ::= \quad 52\% x \\
  20\% e_1 \overline{e_i} \\
  10\% \text{fun } x \rightarrow e \\
  8.1\% \text{ctor} \\
  6.6\% c \\
  1.9\% \text{match } e_1 \text{ with } C... \rightarrow e_i \\
  1.3\% \text{if } e_1 \text{ then } e_2 \text{ else } e_3
  \]

  \[
  \text{Names } x \quad ::= \quad 73\% \text{localName} | 27\% \text{pervasivesName} \\
  \text{Local Names } \text{localName} \quad ::= \quad 31\% \text{MostRecentlyIntroduced} \\
  20\% \text{2ndMostRecentlyIntroduced} \\
  11\% \text{3rdMostRecentlyIntroduced} \\
  \ldots \text{etc...}
  \]

- More in paper and preprint appendix
• Goals
• Demos
• Implementation
• Evaluation
• Future Work & Conclusion
The Magnificent Maniposynth

• Goals
• Demos
• Implementation
• **Evaluation**
• Future Work & Conclusion
Two Evaluations

1. An expert (me) implemented 38 examples from the first lessons of a functional data structures course (IN2347 Functional Data Structures, Technische Universität München)

2. Exploratory user study with two professional OCaml programmers

Goal: qualitative insights. What is or is not working?
## Example Implementation Results

<table>
<thead>
<tr>
<th>Function</th>
<th>LOC</th>
<th>Asserts Time</th>
<th>Mouse</th>
<th>Keybd</th>
<th>Un/Re/Del</th>
<th>TypeErr</th>
<th>Crash</th>
</tr>
</thead>
<tbody>
<tr>
<td>nat_plus</td>
<td>5</td>
<td>0.8</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_minus</td>
<td>8</td>
<td>1.9</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_mult</td>
<td>9</td>
<td>1.4</td>
<td>8</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_exp</td>
<td>13</td>
<td>2.1</td>
<td>9</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_factorial</td>
<td>13</td>
<td>1.6</td>
<td>8</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_map_sumi</td>
<td>10</td>
<td>2.6</td>
<td>11</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>count</td>
<td>9</td>
<td>1.9</td>
<td>9</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>4</td>
<td>0.3</td>
<td>1</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>snoc</td>
<td>8</td>
<td>1.4</td>
<td>8</td>
<td>12</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reverse</td>
<td>8</td>
<td>1.5</td>
<td>4</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_list_max</td>
<td>17</td>
<td>4.6</td>
<td>23</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_list_sum</td>
<td>13</td>
<td>1.1</td>
<td>9</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fold</td>
<td>9</td>
<td>3.2</td>
<td>14</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shuffles</td>
<td>14</td>
<td>14.5</td>
<td>28</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contains</td>
<td>9</td>
<td>2.2</td>
<td>10</td>
<td>13</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>distinct</td>
<td>16</td>
<td>2.4</td>
<td>9</td>
<td>11</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>foldl</td>
<td>10</td>
<td>1.5</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>foldr</td>
<td>8</td>
<td>1.8</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slice</td>
<td>12</td>
<td>9.8</td>
<td>19</td>
<td>22</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>append</td>
<td>8</td>
<td>1.4</td>
<td>7</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sort_by</td>
<td>21</td>
<td>6.2</td>
<td>17</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quickselect</td>
<td>13</td>
<td>13.1</td>
<td>19</td>
<td>38</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sort</td>
<td>16</td>
<td>3.6</td>
<td>11</td>
<td>32</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ltree_inorder</td>
<td>12</td>
<td>2.9</td>
<td>7</td>
<td>20</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ltree_fold</td>
<td>13</td>
<td>3.1</td>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ltree_mirror</td>
<td>11</td>
<td>4.4</td>
<td>12</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bst_contains</td>
<td>14</td>
<td>6.6</td>
<td>11</td>
<td>32</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bstContains2</td>
<td>17</td>
<td>5.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>btree_join</td>
<td>34</td>
<td>61.7</td>
<td>82</td>
<td>64</td>
<td>51</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>bst_delete</td>
<td>36</td>
<td>14.4</td>
<td>31</td>
<td>24</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bstd_valid</td>
<td>29</td>
<td>32.2</td>
<td>63</td>
<td>100</td>
<td>4</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>bstd_insert</td>
<td>18</td>
<td>8.0</td>
<td>38</td>
<td>23</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bstd_count</td>
<td>21</td>
<td>7.6</td>
<td>15</td>
<td>32</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bst_in_range</td>
<td>31</td>
<td>9.3</td>
<td>23</td>
<td>39</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>btree_enum</td>
<td>29</td>
<td>19.2</td>
<td>31</td>
<td>51</td>
<td>6</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>btree_height</td>
<td>15</td>
<td>1.9</td>
<td>11</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>btree_pretty</td>
<td>14</td>
<td>3.7</td>
<td>4</td>
<td>21</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>btree_same_shape</td>
<td>19</td>
<td>8.1</td>
<td>14</td>
<td>34</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>566</td>
<td>44</td>
<td>277.6</td>
<td>628</td>
<td>814</td>
<td>97</td>
<td>13</td>
</tr>
</tbody>
</table>

4.5 hours, 3 tool crashes, but success!
Observations

Could hide the code

No trouble with binding order
(some trouble with nested match order)

Value-oriented vs. expression-oriented thinking

```
length3 = length[0;0;0]
```

3

Despite trying to place attention on values…

…often thought only about expressions.
User Study

• 2 participants x 3 sessions x 2 hours each

• 5 and 11 years of professional OCaml experience

• Ran Maniposynth on their own computers alongside Vim

• Participants attempted exercises with varying amounts of guidance from facilitator

• Goal: qualitative insights
Observations from User Study

- Positive about non-linearity: “fits a lot more with how I like to write code” (P1)

- Too many colors, too few labels

- Even in session three, both participants occasionally still needed guidance from the facilitator

- Writing assertions was not a problem: both wanted to do so, unprompted

- Synthesis only produced useful results 16% of the time, but participants were not bothered when it did not

- (More in paper)
P2 didn’t fully realize they were working with live values until *after* the first exercise.

P1 & facilitator stuck on a bug that was clear from looking at the live values.

P2 was so used to reading

\[ \text{Node (Node (Leaf 1, Leaf 2), Leaf 3)} \]

they were subtly repelled by beautified trees.

Expression-oriented vs. Value-oriented thinking
• Goals

• Demos

• Implementation

• Evaluation

• Future Work & Conclusion
• Goals

• Demos

• Implementation

• Evaluation

• Future Work & Conclusion
Future Work

More self-description in UI (Tooltips?)

Shrink large values

Encourage value-oriented thinking

- Display values instead of variable names?

- More actions on values?
Yes, you can have a **graphical, non-linear** interface even when the program is **ordinary code**.
Thank you!

Visit maniposynth.org for artifact and video