Basic Structure of Denotational Definitions

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, A-4040 Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine
A Calculator Language

- Buttons and display screen,
- Single memory cell,
- Conditional evaluation feature.

<table>
<thead>
<tr>
<th>Input</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>(4 + 12) * 2</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>32</td>
</tr>
<tr>
<td>1 + LASTANSWER</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>33</td>
</tr>
<tr>
<td>IF LASTANSWER + 1, 0, 2 + 4</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>6</td>
</tr>
<tr>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>

(See Schmidt, Figures 4.2 and 4.3)
Evaluation Functions

- **P**: Program $\rightarrow$ $Nat^*$
  Program mapped to list of outputs.

- **S**: Expr-sequence $\rightarrow$ $Nat$ $\rightarrow$ $Nat^*$
  Expression sequence and content of memory cell mapped to list of outputs.

- **E**: Expression $\rightarrow$ $Nat$ $\rightarrow$ $Nat$
  Expression and content of memory cell mapped to evaluation result.

- **N**: Numeral $\rightarrow$ $Nat$
  Numeral mapped to natural number.
Observations

1. Global data structures are modelled as arguments to valuation functions.
   No “global variables” for functions.

2. Meaning of a syntactic construct can be a function.
   S’s functionality states that the meaning of an expression sequence is a function from a memory cell to a list of numbers.
S Rule

\[ S[[E \ \text{TOTAL} \ S]] \]

- **Calculator actions:**
  1. Evaluate \([E]\) using cell \(n\) producing value \(n'\).
  2. Print \(n'\) on the display.
  3. Place \(n'\) into the memory cell.
  4. Evaluate the rest of sequence \([S]\) using the cell.

- **Representation in semantic equation**
  1. \(E[[E]](n)\) is bound to variable \(n'\),
  2. \(n' \ \text{cons} \ldots\)
  3. and 4. \(S[[S]](n')\)

*However right-hand side of equation is a mathematical value!*
**Simplification**

\[ P[[\text{ON } 2+1 \ \text{TOTAL IF LA}, 2, 0 \ \text{TOTAL OFF}]] = S[[2+1 \ \text{TOTAL IF LA}, 2, 0 \ \text{TOTAL OFF}}](\text{zero}) \]

\[ = \text{let } n' = E[[2+1]](\text{zero}) \]
\[ \quad \text{in } n' \ \text{cons } S[[\text{IF LA}, 2, 0 \ \text{TOTAL OFF}]](n') \]
\[ = \text{let } n' = \text{three} \]
\[ \quad \text{in } n' \ \text{cons } S[[\text{IF LA}, 2, 0 \ \text{TOTAL OFF}]](n') \]
\[ = \text{three cons } S[[\text{IF LA}, 2, 0 \ \text{TOTAL OFF}]](\text{three}) \]
\[ = \text{three cons } (E[[\text{IF LA}, 2, 0]](\text{three}) \ \text{cons } \text{nil}) \]
\[ = \text{three cons } (\text{zero cons } \text{nil}) \]

\[ E[[\text{IF LA}, 2, 0]](\text{three}) \]
\[ = E[[\text{LA}]](\text{three}) \ \text{equals } \text{zero} \rightarrow \]
\[ E[[2]](\text{three}) \ \text{&} \ E[[0]](\text{three}) \]
\[ = \text{three equals } \text{zero} \rightarrow \text{two } \\text{&} \ \text{zero} \]
\[ = \text{false} \rightarrow \text{two } \\text{&} \ \text{zero} \]
\[ = \text{zero} \]
Simplification

- Each simplification step preserves meaning.
- Goal is to produce equivalent expression whose meaning is more obvious than the meaning of the original.
- Simplification process shows how program operates.
- Denotational definition $\rightarrow$ specification.
- Denotational definition plus simplification strategy $\rightarrow$ implementation.