Specification of Iterators

Bruce W. Weide
Reusable Software Research Group
The Ohio State University
http://www.cse.ohio-state.edu/rsrg
Important Assumptions

• Ultimate objective is full behavioral specification, with support for modular verification of client code

• Language has value semantics
Claims

• Full behavioral specification achieved
  – Only previously existing, and ordinary, model-based specification techniques needed
• Design and specs easy to understand
• Implementation can be as efficient as for any other iterator design
• Modular verification of client code possible
Client Code for Iteration

Start_Iterator (i, s, p)
loop
   while Length_Of_Future (i) > 0 do
      Get_Next_Item (i, p)
      /* process p, with no net change */
   end loop
Finish_Iterator (i, s, p)
StartIterator (i, s, p)
Get\_Next\_Item (i, p)
Get\_Next\_Item (i, p)
Get\_Next\_Item (i, p)
### Get\_Next\_Item(i, p)

<table>
<thead>
<tr>
<th>s</th>
<th>i</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>past:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>future:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>original:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>past:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>future:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>original:</td>
</tr>
</tbody>
</table>

**Diagram:**
- past: ~star~
- future: ~triangle~
- original: ~red star~
Finish_Iterator (i, s, p)
Specification of Iterators

Bruce W. Weide
Reusable Software Research Group
The Ohio State University
http://www.cse.ohio-state.edu/rsrg
Client Code: Differences

Start_Iterator (i, s, \(x\))
loop
  /* loop invariant; see paper */
while Length_Of_Future (i) > 0 do
  Get_Next_Item (i, x)
  /* process \(x\), with no net change; modify \(s\) if you like! */
end loop
Finish_Iterator (i, s, \(x\))
Specification Details

type family Set_Iterator is modeled by ( 
    past: string of Item,
    future: string of Item,
    original: finite set of Item
)

exemplar i
initialization ensures
    i = (<> , <>, { })
operation Start_Iterator (i: Set_Iterator, s: Set, x: Item)

ensures

there exists f: string of Item

(elements (f) = #s and
 |f| =|#s| and
 i = (<x>, f, #s)) and

s = {} and

x = #x
Specification Details

\begin{verbatim}
operation Finish_Iterator ( 
    i: Set_Iterator, s: Set, x: Item)

    requires
        \langle x \rangle \text{ is suffix of } i.past

    ensures
        \text{is_initial (} i \text{) and }
        s = \#i.original \text{ and }
        \langle x \rangle \text{ is prefix of } \#i.past
\end{verbatim}
**Specification Details**

**operation** Get_Next_Item (i: Set_Iterator, x: Item)

**requires**

\[ i.\text{future} /= \langle > \text{ and } \langle x \rangle \text{ is suffix of } i.\text{past} \]

**ensures**

\[ \text{there exists } f: \text{ string of Item} \]

\[ (#i.\text{future} = \langle x \rangle \ast f \text{ and } i = (#i.\text{past} \ast \langle x \rangle, f, #i.\text{original})) \]
operation Length_Of_Future (  
    i: Set_Iterator): Integer  
ensures  
    Length_Of_Future = |i.future|