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

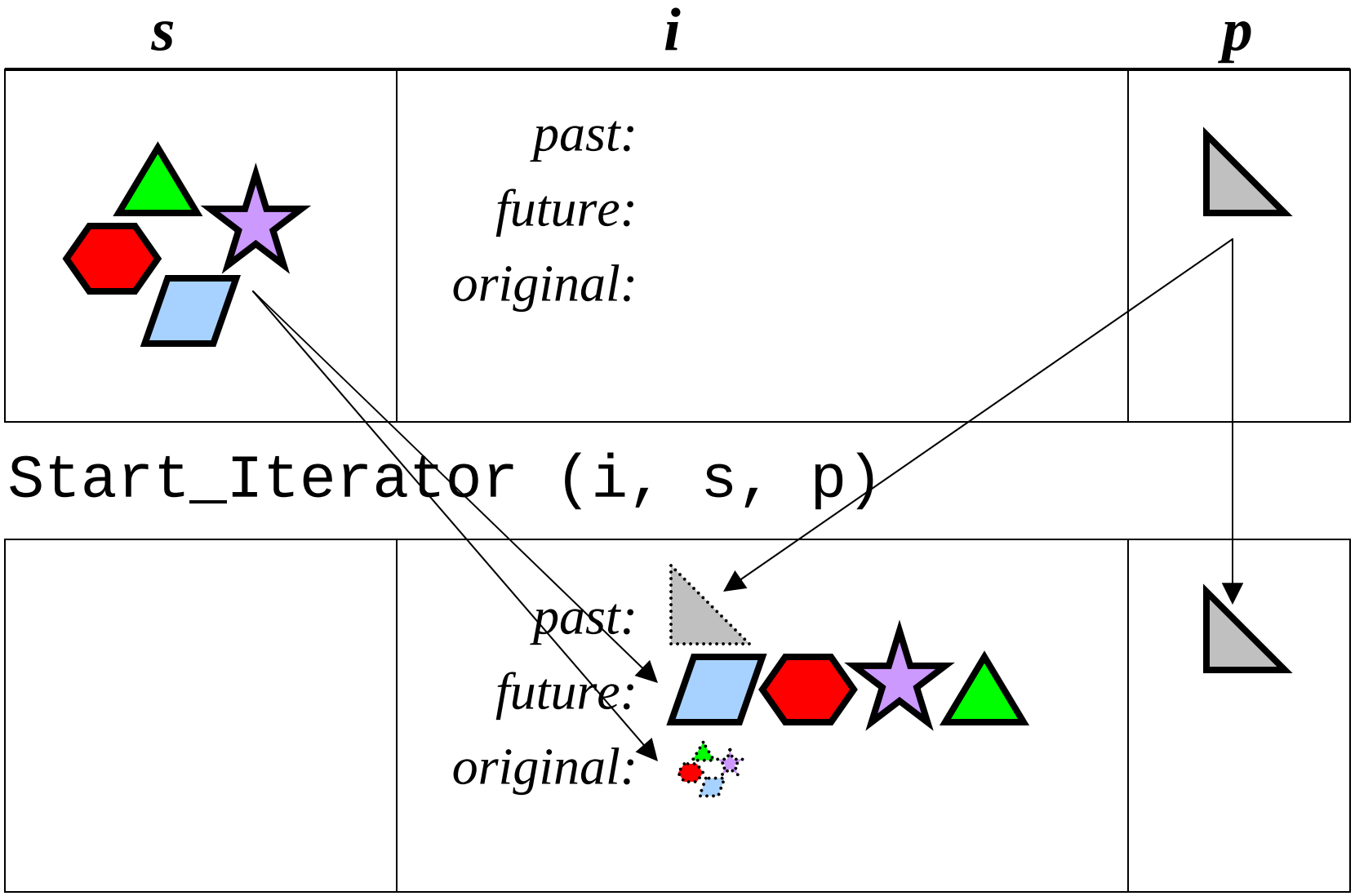
- Full behavioral specification achieved
  - Only previously existing, and ordinary, model-based specification techniques needed
- Design and specs easy to understand
- Implementation can be as efficient as for any other iterator design
- Modular verification of client code possible

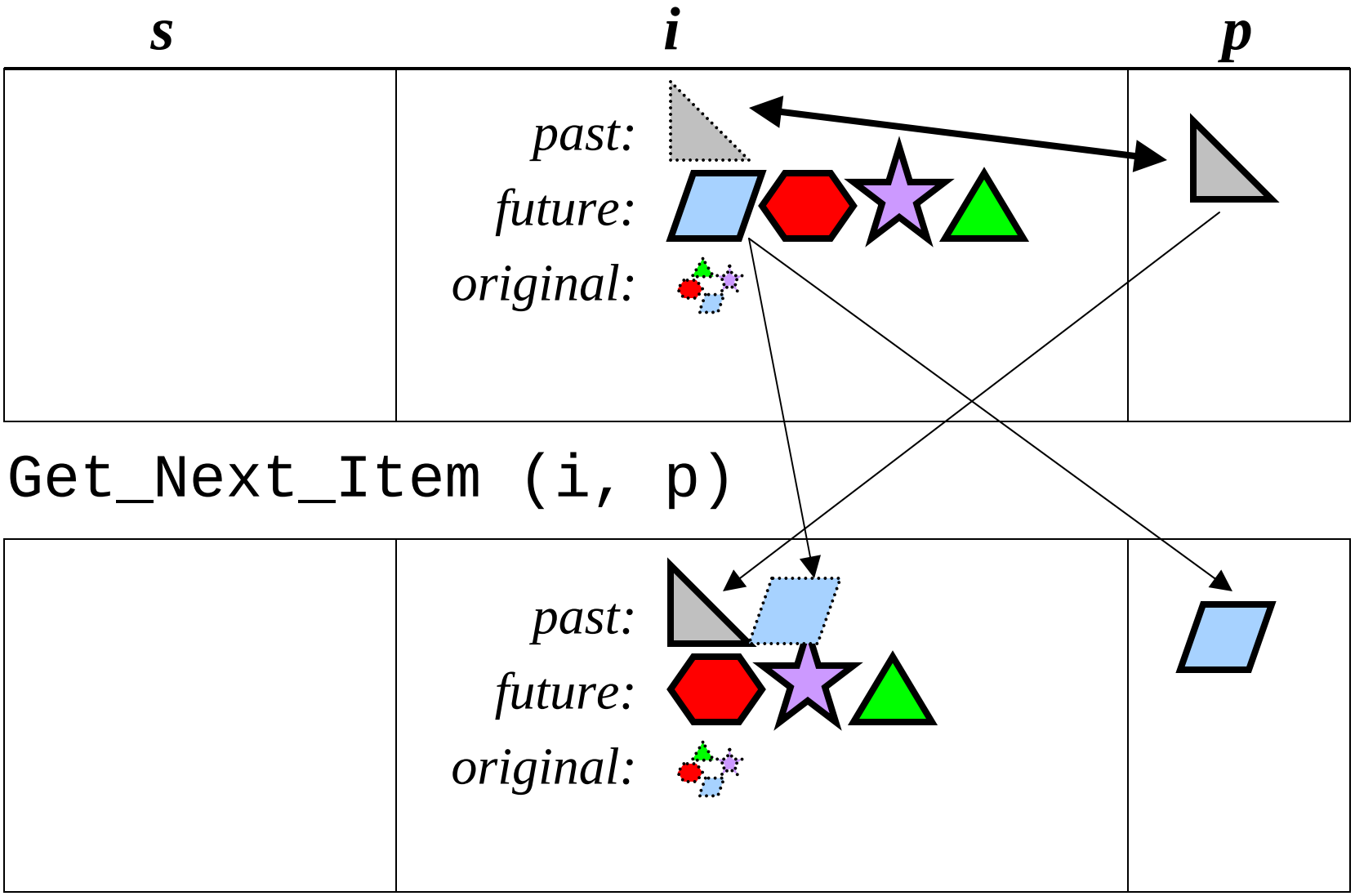
Client Code for Iteration

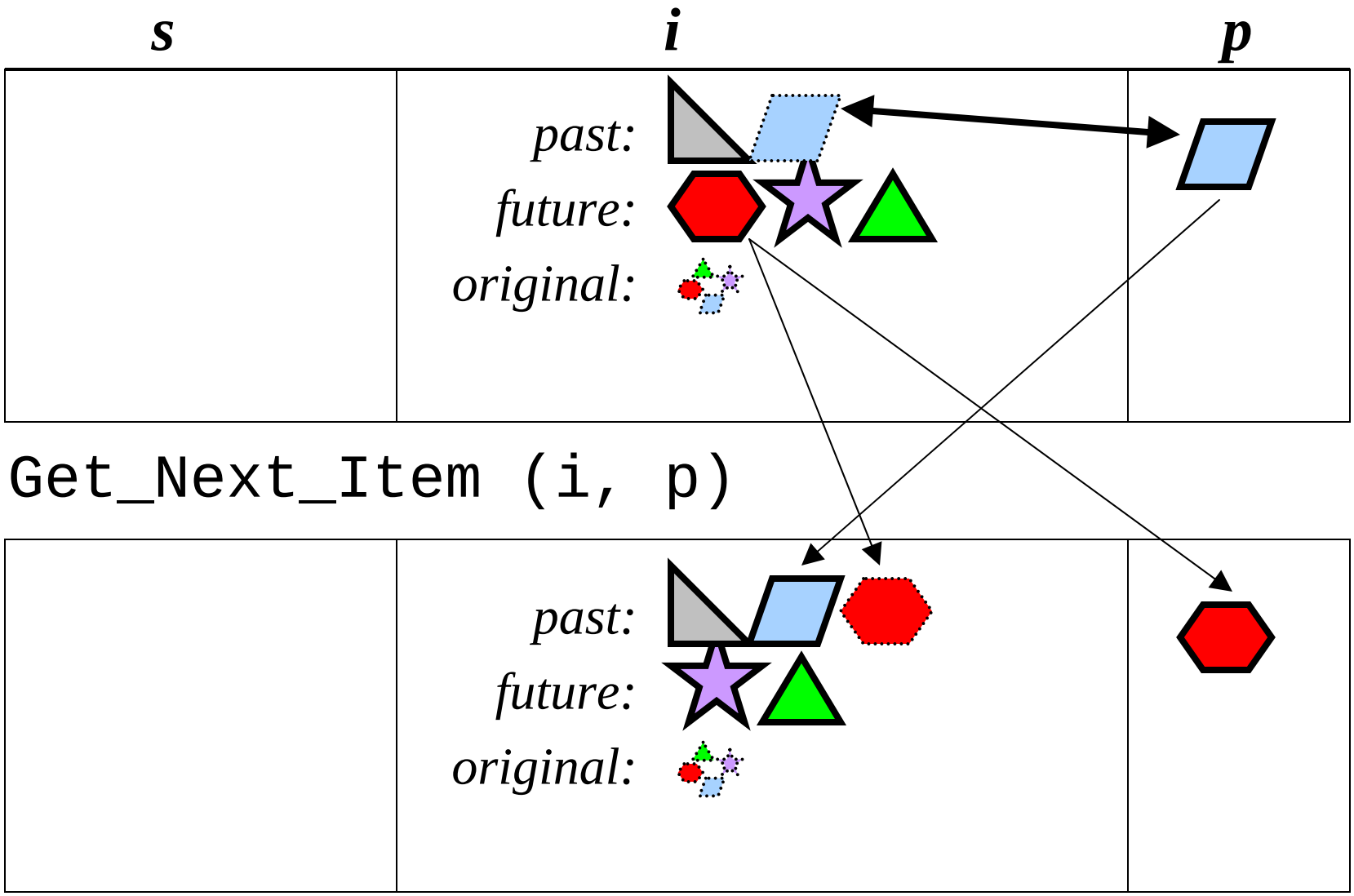
```
Start_Iterator (i, s, p)
loop
   /*                                    */
while                                    do
   Get
   /* process p, with no net change */
end loop
Finish_Iterator (i, s, p)
```

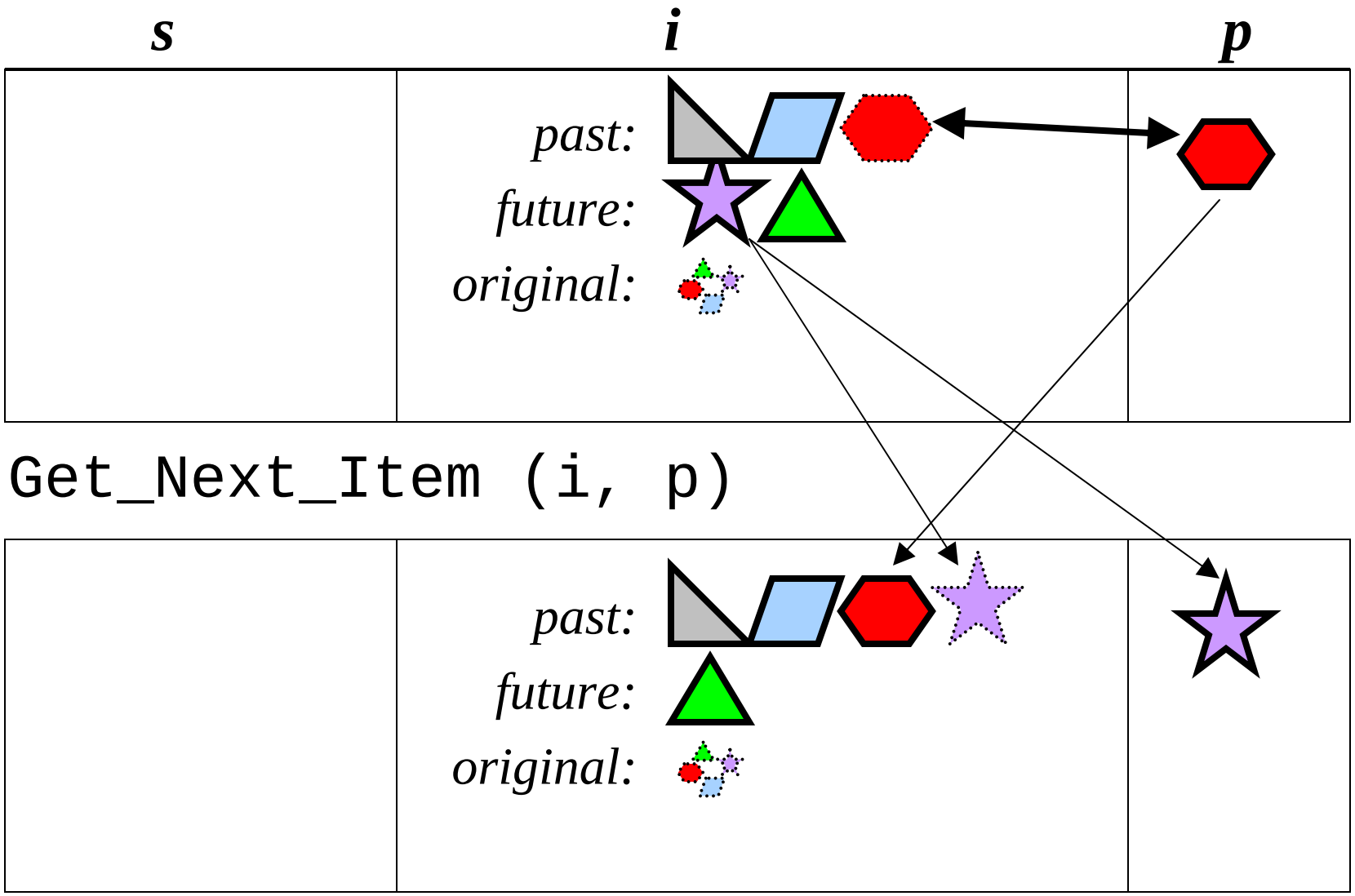
an Iterator for
a Set of Polygons

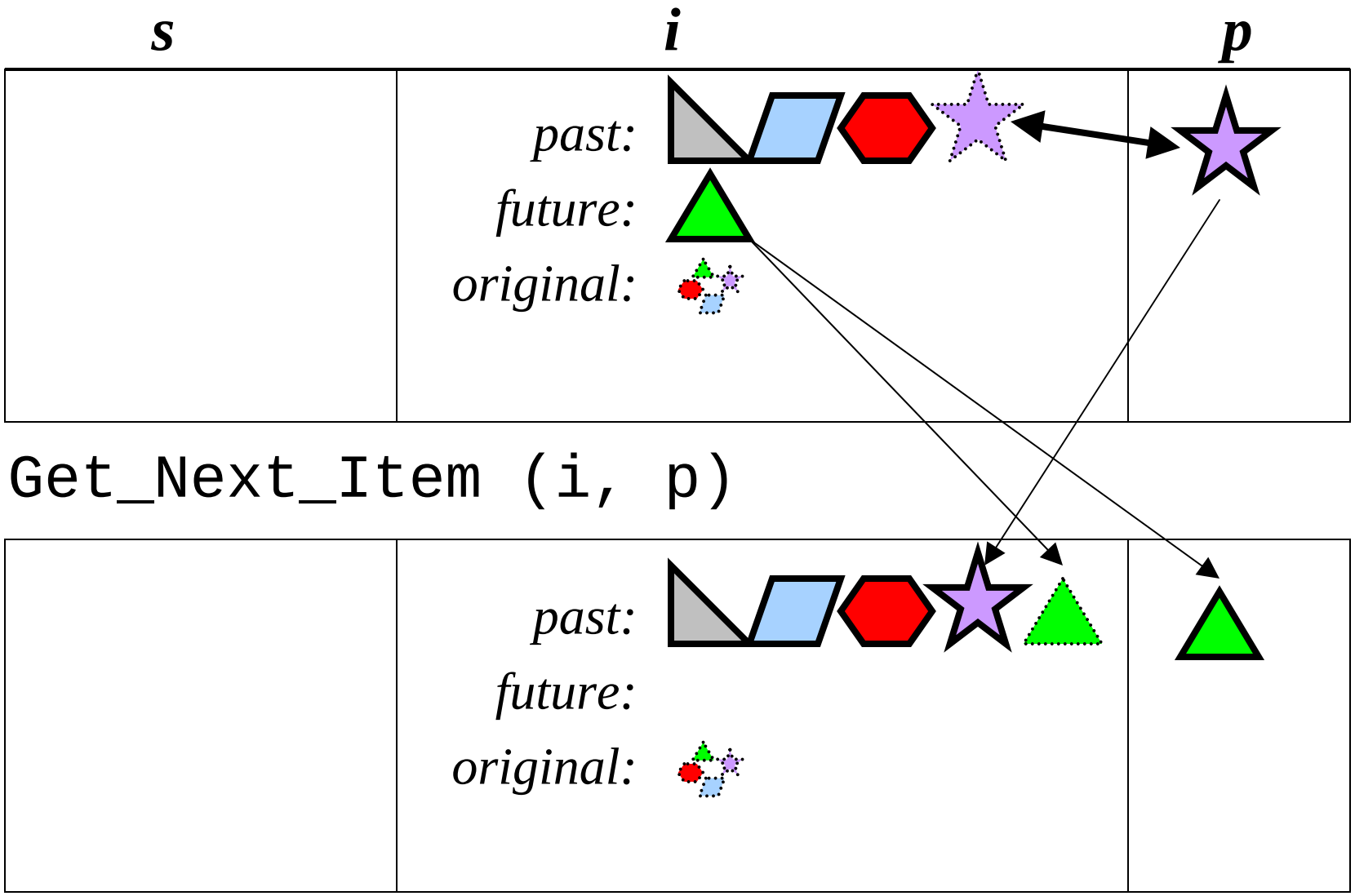
a Polygon

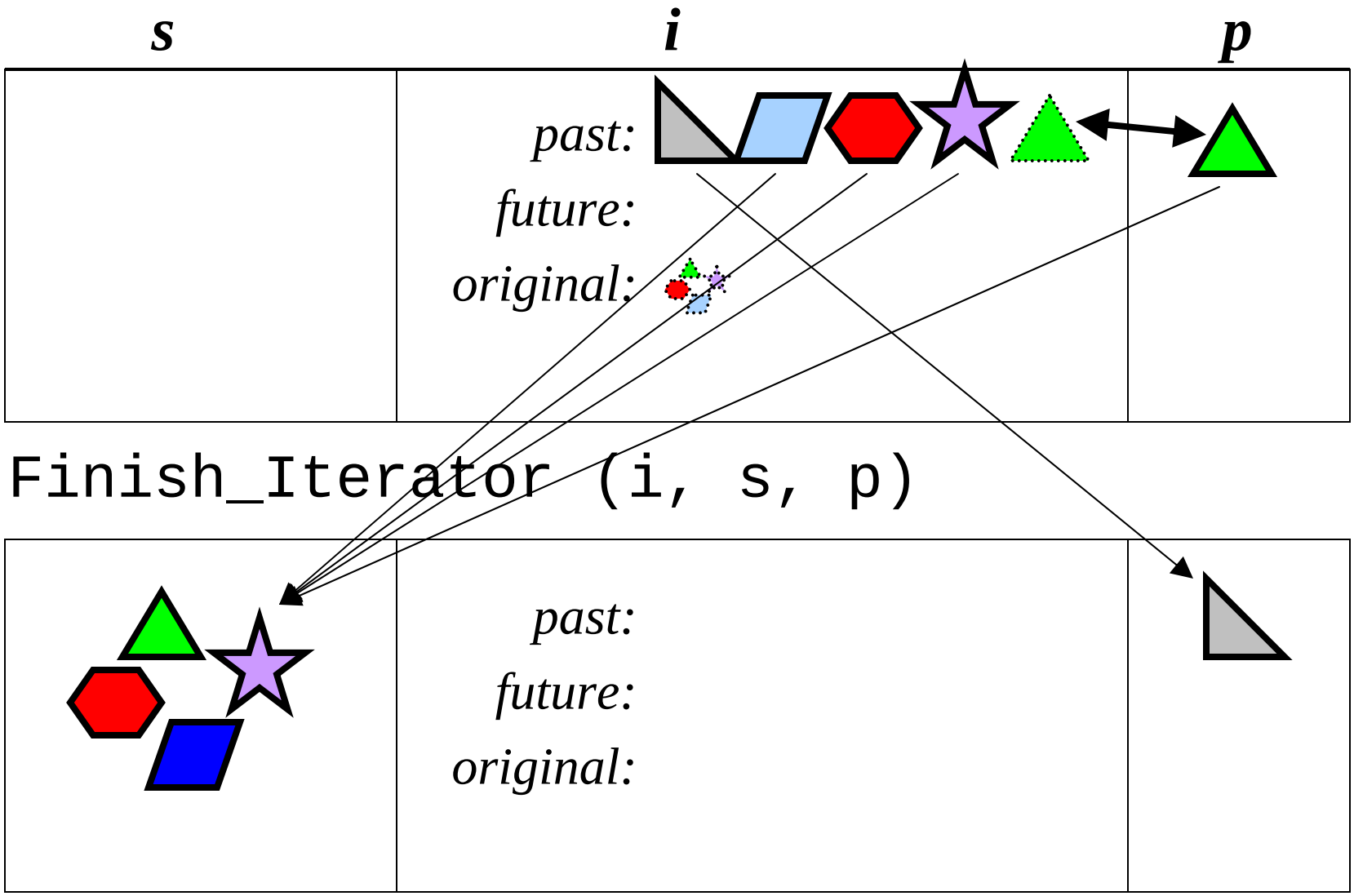

s
i
p
past:
future:
original:
Start_Iterator (i, s, p)
past:
future:
original:


s
i
p
past:
future:
original:
Get_Next_Item (i, p)
past:
future:
original:


s
i
p
past:
future:
original:
Get_Next_Item (i, p)
past:
future:
original:


s
i
p
past:
future:
original:
Get_Next_Item (i, p)
past:
future:
original:


s
i
p
past:
future:
original:
Get_Next_Item (i, p)
past:
future:
original:


s
i
p
past:
future:
original:
Finish_Iterator (i, s, p)
past:
future:
original:

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 = (< >, <>, { })
```

Specification Details

```
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

```
operation Finish_Iterator (
        i: Set_Iterator, s: Set, x: Item)
    requires
        <x> is suffix of i.past
    ensures
        is_initial (i)  and
        s = #i.original  and
        <x> is prefix of #i.past
```

Specification Details

```
operation Get_Next_Item (i: Set_Iterator,
        x: Item)
    requires
        i.future /= < >  and
        <x> is suffix of i.past
    ensures
        there exists f: string of Item
              (#i.future = <x> * f  and
               i = (#i.past * <x>, f, #i.original))
```

Specification Details

```
operation Length_Of_Future (
        i: Set_Iterator): Integer
    ensures
        Length_Of_Future = |i.future|
```