

Software Productivity

Paulo Borba
Informatics Center
Federal University of Pernambuco

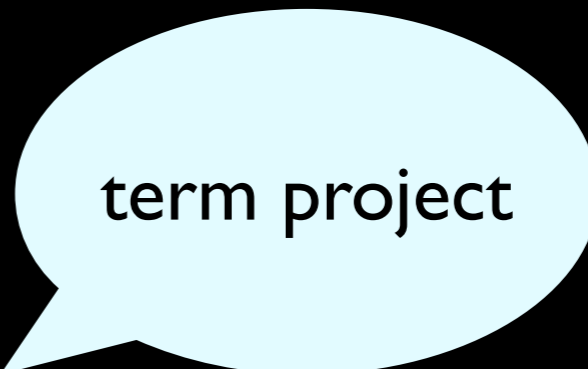
phmb@cin.ufpe.br ♦ twitter.com/pauloborba

Congruences and iterations

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba

Pre-defined low-level strategies

```
io-entity-eval =  
  io-stream-wrap (  
    debug  
    ; ?(<id>, fout)  term project  
    ; debug  
    ; parse-entity-stream  
    ; debug  
    ; entity-to-class  
    ; debug  
    ; <WriteToFile>  
      ("transformed.txt", <id>)  
  )
```

term wrap

Term project and wrap

`?p [<s>]`

is syntactic sugar for

`{x: ?p[x]; <s> x}`

`!p [<s>]`

is syntactic sugar for

`{x: where (s => x); !p[x]}`

Strategy library, data operators as strategy operators

- add,
- div,
- mul,
- sub_t
- nub
- union
- diff
- isect
- set-eq
- ...

Congruence operators

```
map(s) = [] <+ [s | map(s)]
```

```
fetch(s) = [s | id] <+ [id | fetch(s)]
```

```
filter(s) = [] +
```

```
    ([s | filter(s)] <+
```

```
    ?[ |<id>]; filter(s))
```

Interface generation: method declarations into signatures

```
method-dec-to-abstract-method-dec:
```

```
MethodDecHead(mods, x, t, x_method, args, y)
```

```
->
```

```
AbstractMethodDec(mods, x, t, x_method, args, y)
```

```
where <fetch(?Public())> mods
```

Transforming lists

```
<foldr(s1,s2)> [t1,t2,t3]
yields
  <s2>(t1,
    <s2>(t2,
      <s2>(t3,
        <s1> [ ])))
```

```
foldr(s1, s2) =
  []; s1 <+
  \ [y|ys] ->
    <s2> (y, <foldr(s1, s2)> ys) \
```

anonymous rule

Sum

```
<foldr (!0, add) > [t1, t2, t3]  
yields  
  <add> (t1,  
        <add> (t2,  
              <add> (t3,  
                    <!0> [ ])))
```

```
sum = foldr (!0, add)
```

Length and occurrences, combining map and foldr

```
foldr (s1, s2, f) =  
  []; s1 <+  
  \ [y|ys] ->  
    <s2> (<f> y, <foldr (s1, s2, f)> ys) \
```

```
length = foldr (!0, add, !1)  
list-occurrences (s) =  
  foldr (!0, add, s < !1 + !0)
```

Interface generation:

term as
parameter

```
create-local-interface(|x_Interface):  
  class  
->  |[ @Local public interface x_Interface {  
    ~*methodsdecs  
  }  
  ]|  
where  
  <extract-method-signatures> class  
=> methodsdecs
```

Reflection, match strategy

`?p1# (p2)`

decomposes a constructor application into its
constructor name and the list of direct subterms

Occurrences, beyond lists

```
crush(nul, sum, s) :
```

```
  c#(xs) -> <foldr(nul, sum, s)> xs
```

```
occurrences(s) =
```

```
  <add>(<s < !1 + !0> ,
```

```
    <crush(!0, add, occurrences(s))>)
```

Collecting terms

```
collect(s) =
```

```
! [<s>] <+
```

```
crush(![], union, collect(s))
```

```
collect-all(s) =
```

```
! [<s> | <crush(![], union, collect(s))>]
```

```
<+ crush(![], union, collect(s))
```

Interface generation: traversing a class

```
extract-method-signatures =  
  collect(method-dec-to-abstract-method-dec)
```

```
method-dec-to-abstract-method-dec:  
  MethodDecHead(mods, x, t, x_method, args, y)  
->  
  AbstractMethodDec(mods, x, t, x_method, args, y)  
  
where <fetch(?Public())> mods
```

Interface generation:

term as
parameter

```
create-local-interface(|x_Interface):  
  class  
->  |[ @Local public interface x_Interface {  
    ~*methodsdecs  
  }  
  ]|  
where  
  <extract-method-signatures> class  
=> methodsdecs
```


Congruences and iterations

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba

Software Productivity

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba