Software Productivity

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

Strategies, style, optimization

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

Adding blocks, complete solution

```
strategies
  main = io-java2java-wrap(add-block)

  add-block = topdown(try(if-else-rules))

  if-else-rules = AddBlockIf +
                  AddBlockIfElse +
                  AddBlockIfElseIf +
                  AddBlockIfElseElse

  notBlock = not(?(stm |[ {bstm*} ]|))
```

Code cloning

```
AddBlockIfElse:
  |[ if (e_) stm else stm2 ]| ->
  |[ if (e_) {stm} else {stm2} ]|
    where and(<notBlock> stm, <notBlock> stm2)

AddBlockIfElseIf:
  |[ if (e_) stm else {stm2} ]| ->
  |[ if (e_) {stm} else {stm2} ]|
     where <notBlock> stm

AddBlockIfElseElse:
  |[ if (e_) {stm} else stm2 ]| ->
  |[ if (e_) {stm} else {stm2} ]|
     where <notBlock> stm2
...
```

Code cloning, oops!

```
AddBlockIfElse:
  |[ if (e_) stm else stm2 ]| ->
  |[ if (e_) {stm} else {stm2} ]|
    where and(<notBlock> stm, <notBlock> stm2)

AddBlockIfElseIf:
  |[ if (e_) stm else {bstm*} ]| ->
  |[ if (e_) {stm} else {bstm*} ]|
     where <notBlock> stm

AddBlockIfElseElse:
  |[ if (e_) {bstm*} else stm2 ]| ->
  |[ if (e_) {bstm*} else {stm2} ]|
     where <notBlock> stm2
...
```

Refactoring: extracting rule

```
strategies...
  add-block =
    topdown(try(AddBlockIf + AddBlockIfElse))

  addBrackets = try(addBracketsRule)

rules...
  addBracketsRule:
     stm -> stm |[ {stm} ]|
        where <notBlock> stm

  AddBlockIf:
    |[ if (e_) stm ]| -> |[ if (e_) stm2 ]|
        where <addBrackets> stm => stm2
```

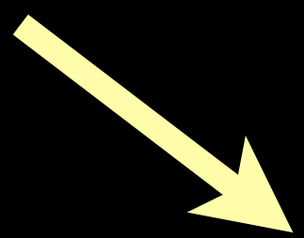
Composition versus conjuction

```
AddBlockIfElse:
    |[ if (e_) stm else stm2]| ->
    |[ if (e_) stm3 else stm4 ]|
       where <addBrackets> stm => stm3 ;
             <addBrackets> stm2 => stm4

AddBlockIfElseEquivalentToAbove:
    |[ if (e_) stm else stm2]| ->
    |[ if (e_) stm3 else stm4 ]|
       where and(<addBrackets> stm => stm3,
                 <addBrackets> stm2 => stm4)
```

Avoiding ifs

```
boolean balance?(double v) {
  if (balance >= v)
    return true;
  else
    return false
}
```



```
boolean balance?(double v) {
  return (balance >= v);
}
```

Rules?

Strategies?

Cloning, no!

```
...  |[ if (e_) {
           return true;
         } else {
           return false;
         } ...
```

```
...  |[ if (e_)
           return true;
         else {
           return false;
         } ...
```

```
...  |[ if (e_) {
           return true;
         } else
           return false;
         ...
```

```
...  |[ if (e_)
           return true;
         else
           return false;
         ...
```

Being too specific, no!

```
RemoveUnnecessaryIf-Else-OverConstrained:
    ||[
        public boolean x_Method() {
          if (e_) {
            return true;
          } else {
            return false;
          }
        }
    ]|...
```

Single rule, relying on previous normalization

```
RemoveUnnecessaryIf-Else:
      |[ if (e_) {
           return true;
         } else {
           return false;
         }
    ]|
  ->
    |[ return e_;
    ]|
```

Single rule, alternative

```
RemoveUnnecessaryIf-Else:
      |[ if (e_) {
           return e1;
         } else {
           return e2;
         }
      ]|
   ->
      |[ return e_;
      ]|
   where and(<eq>(e1,|[true]|),
             <eq>(e2,|[false]|))
```

Single rule, negation

```
RemoveUnnecessaryIf-Else-Not:
      |[ if (e_) {
           return e1;
         } else {
           return e2;
         }
     ]|
  ->
     |[ return !e_;
     ]|
  where and(<eq>(e1,|[false]|),
            <eq>(e2,|[true]|))
```

Single rule for both cases, complex

```
RemoveUnnecessaryIf-Else-Not:
    |[ if (e_) {
          return e1;
       } else {
          return e2;
       }
    ]|
->
    |[ return e3;
    ]|
where ...and(<eq>(e1,|[false]|),
             <eq>(e2,|[true]|)) ;
             !(|[ !e_ ]|) => e3 ...
```

Combining strategies

```
main =
  io-java2java-wrap(add-block;
                    remove-unnecessary-if)

remove-unnecessary-if =
  topdown(try(RemoveUnnecessaryIf-Else))
```

Optimizing, but being careful

```
main = io-java2java-wrap(
           remove-unnecessary-if-optimized)

remove-unnecessary-if-optimized =
    topdown(try(AddBlockIf + AddBlockIfElse);
                try(RemoveUnnecessaryIf-Else))

remove-unnecessary-if-wrong =
    topdown(try(
        (AddBlockIf + AddBlockIfElse);
        RemoveUnnecessaryIf-Else))

remove-unnecessary-if-wrong-too =
    topdown(try(AddBlockIf + AddBlockIfElse
                  + RemoveUnnecessaryIf-Else))
```

Choice versus disjunction

```
where (<eq>(e1,|[true]|) ; <eq>(e2,|[false]|)) +
      (<eq>(e1,|[false]|) ; <eq>(e2,|[true]|))
```

```
where or(and(<eq>(e1,|[true]|),<eq>(e2,|[false]|)),
         and(<eq>(e1,|[false]|),<eq>(e2,|[true]|)))
```

Predefined strategy operators so far

- `id`
- `?(_)`
- `not(_)`
- `topdown(_)`
- `try(_)`
- `iojava2java-wrap(_)`
- `_;_`
- `_+_`
- `and(_,_)`
- `or(_,_)`
- `eq`
- `!(_)`
- `_=>_`

Arguments to strategy operators are transformations, functions from terms to terms...

- strategies
- rules
- terms!

Terms as functions

```
|[ balance = 0; ]|
```

```
f(x) = |[ balance = 0; ]|
```

```
|[ balance = 0; ]|(x) = |[ balance = 0; ]|
```

Operator receiving strategy or term...

```
  |[ if (e) stm ]| -> |[ if (e) {stm} ]|
where
  <not(block)> stm
```

```
  |[ if (e) stm ]| -> |[ if (e) {stm} ]|
where
  not(<block> stm)
```

```
  |[ if (e) stm ]| -> |[ if (e) {stm} ]|
where
  not(<<block> stm> stm)
```

unreadable version!

```
  |[ if (e) stm ]|-> |[ if (e) {stm} ]|
where
  < not(?(stm |[ {bstm*} ]|)) > stm
```

```
  |[ if (e) stm ]|  -> |[ if (e) {stm} ]|
where
  not(<?(stm |[ {bstm*} ]|)> stm)
```

```
  |[ if (e) stm ]|  -> |[ if (e) {stm} ]|
where
  not(<<?(stm |[ {bstm*} ]|)> stm> stm)
```

WebDSL to Java transformations

```
rules
  property-to-gettersetter:
    |[ x_prop : srt ]| ->
    |[ private t x_prop;
       public t get#x_prop() {return x_prop;}
       public void set#x_prop(t x_prop) {
         this.x_prop = x_prop;
       }
    ]|
  where <builtin-java-type> srt => t
```

string concatenation

no # operator

```
rules
  property-to-gettersetter:
    |[ x_prop : srt ]| ->
    |[ private t x_prop;
       public t x1() {return x_prop;}
       public void set#x_prop(t x_prop) {
         this.x_prop = x_prop;
       }
    ]|
  where <builtin-java-type> srt => t ;
        <concatString(|"get")> x_prop => x1
```

Strategies, style, optimization

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