Software transformation and generation

Paulo Borba
Informatics Center
Federal University of Pernambuco

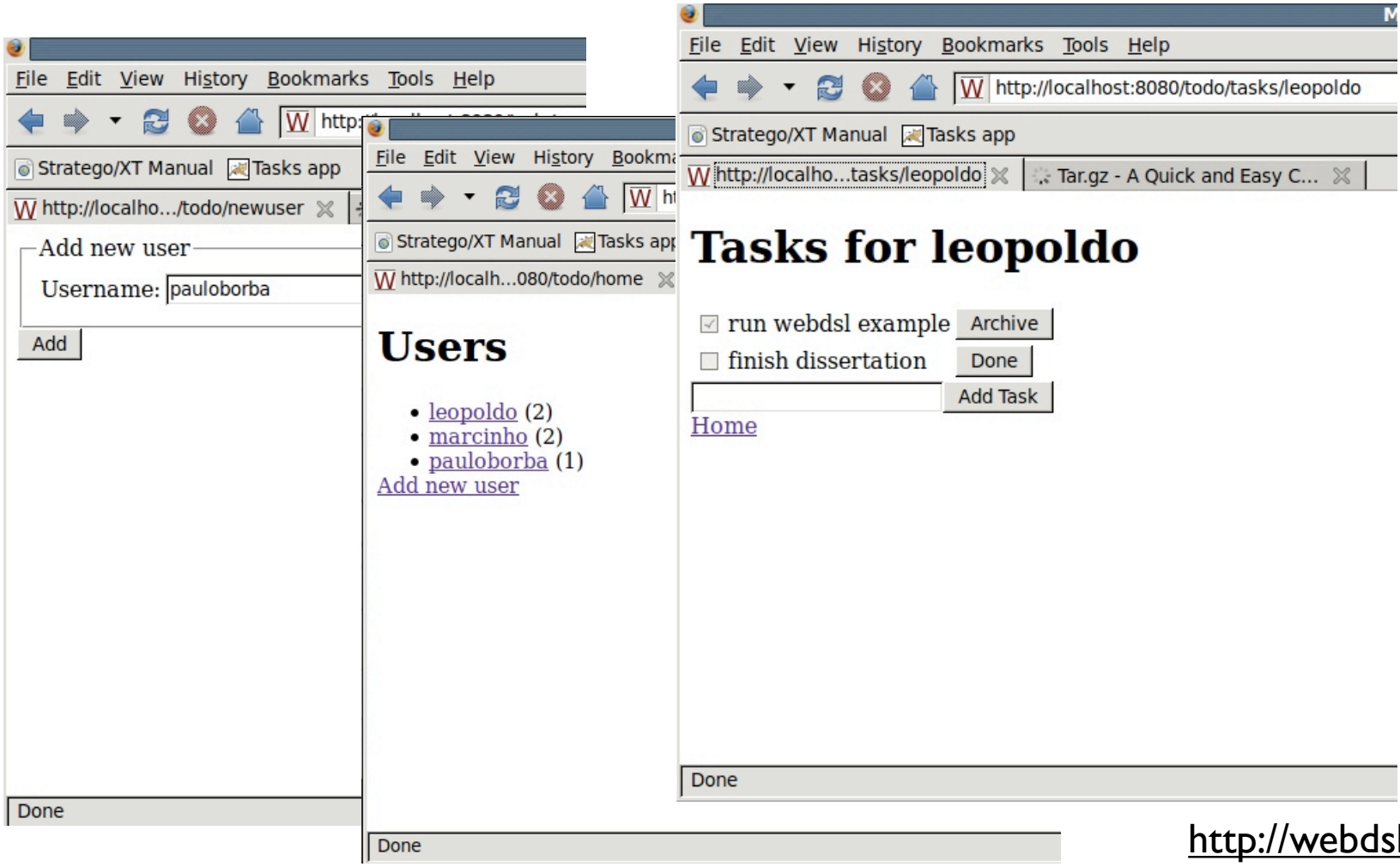
phmb@cin.ufpe.br ◈ twitter.com/pauloborba

Basic transformation concepts

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

Task lists on the web



http://webdsl.org

This is a simple, typical, web application with more than...

- 20 (100) files
- 10 (20) types
- 500 (7000) LOC
- 5 (10) APIs and technologies
- ? bugs

Actually, 80 lines of code!

```
application todo
  entity User {
    username :: String (id,name)
    tasks    -> List<Task>
    log      -> List<Task>
  }
  entity Task {
    description :: String
    done        :: Bool
  }
```

```
define page home() {
  section{
    header{"Users"}
    list{
      for(user : User) {
        listitem{
          navigate(tasks(user))
            {output(user.username)}
            " (" output(user.tasks.length) ")"
        }
      }
    }
    navigate(newuser()){"Add new user"}
  }
}
```

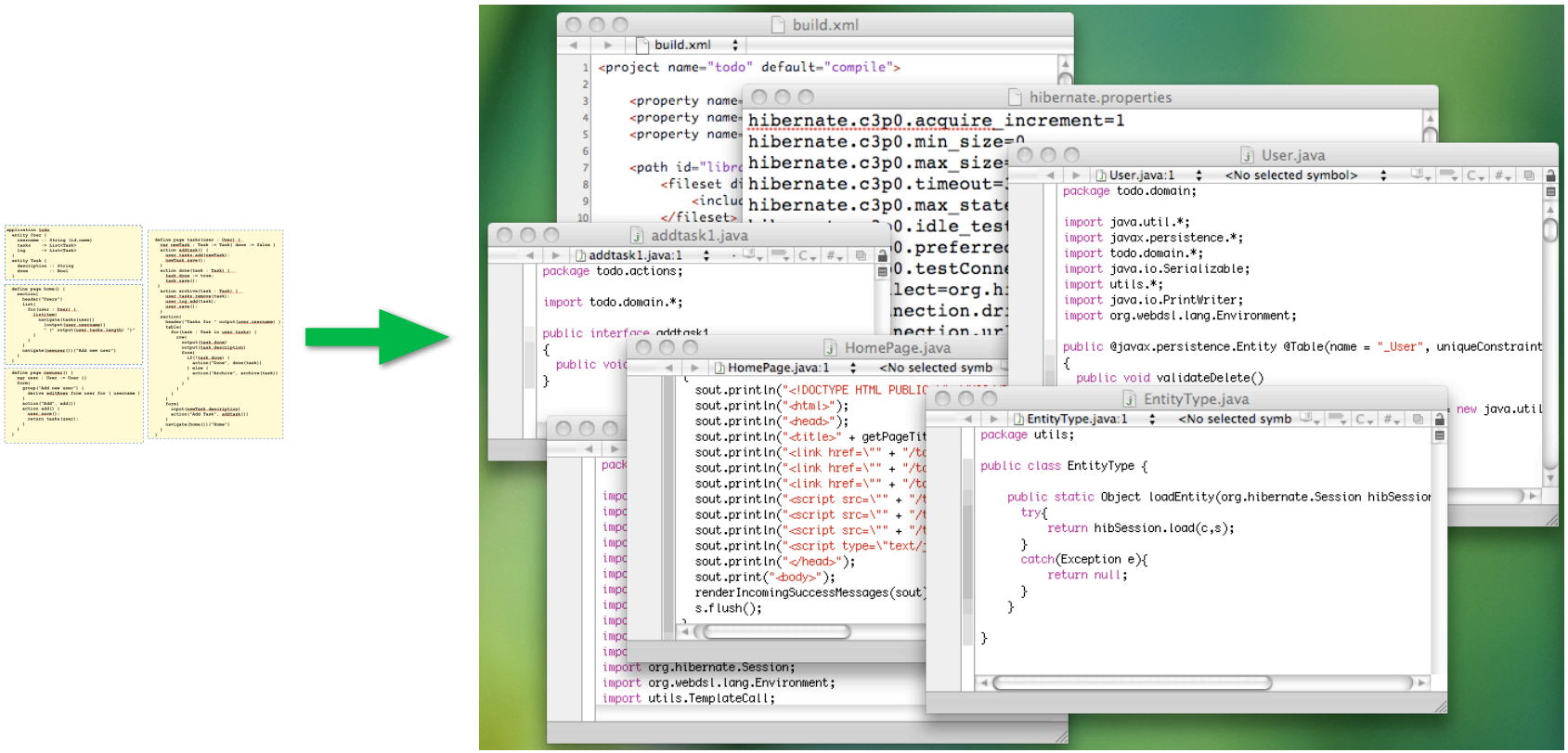
```
application todo
  entity User {
    username :: String (id,name)
    tasks    -> List<Task>
    log      -> List<Task>
  }
  entity Task {
    description :: String
    done        :: Bool
  }
```

```
  define page home() {
    section{
      header{"Users"}
      list{
        for(user : User) {
          listitem{
            navigate(tasks(user))
              {output(user.username)}
              " (" output(user.tasks.length) ")"
          }
        }
      }
      navigate(newuser()){"Add new user"}
    }
  }
```

```
  define page newuser() {
    var user : User := User {}
    form{
      group("Add new user") {
        derive editRows from user for ( username )
      }
      action("Add", add())
      action add() {
        user.save();
        return tasks(user);
      }
    }
  }
```

```
define page tasks(user : User) {
  var newTask : Task := Task{ done := false }
  action addtask() {
    user.tasks.add(newTask);
    newTask.save();
  }
  action done(task : Task) {
    task.done := true;
    task.save();
 }
  action archive(task : Task) {
    user.tasks.remove(task);
    user.log.add(task);
    user.save();
  }
  section{
    header{"Tasks for " output(user.username) }
    table{
      for(task : Task in user.tasks) {
        row{
          output(task.done)
          output(task.description)
          form{
            if(!task.done) {
              action("Done", done(task))
            } else {
              action("Archive", archive(task))
            }
          }
        }
      }
    }
    form{
      input(newTask.description)
      action("Add Task", addtask())
    }
    navigate(home()){"Home"}
  }
}
```

Transformations!



Productivity and Quality

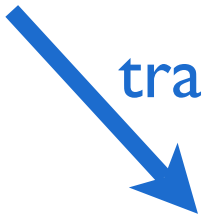
Defining a transformation

```
Not(False) -> True
```

Applying a transformation (exhaustively)

```
Not(False) -> True
```

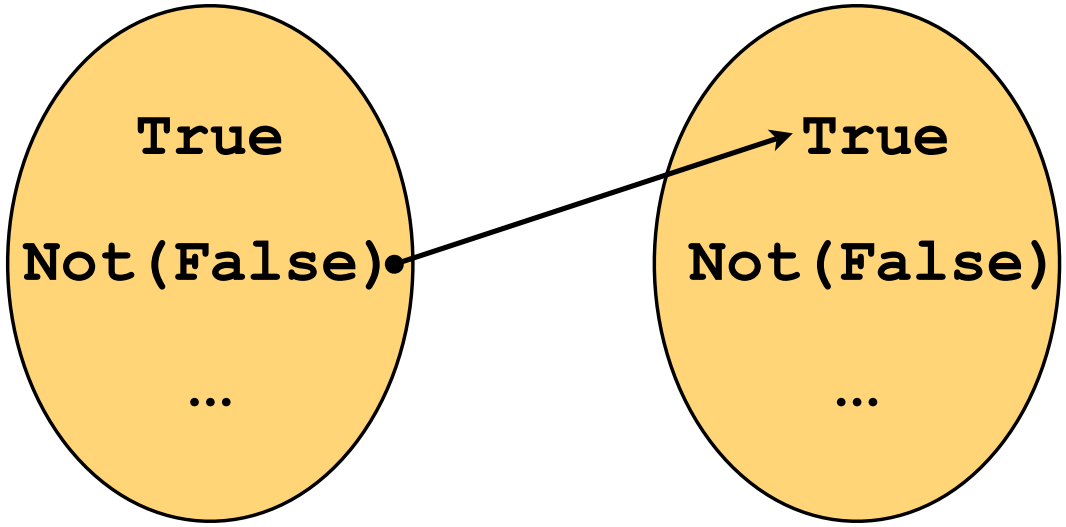
```
Not(False) And True
```



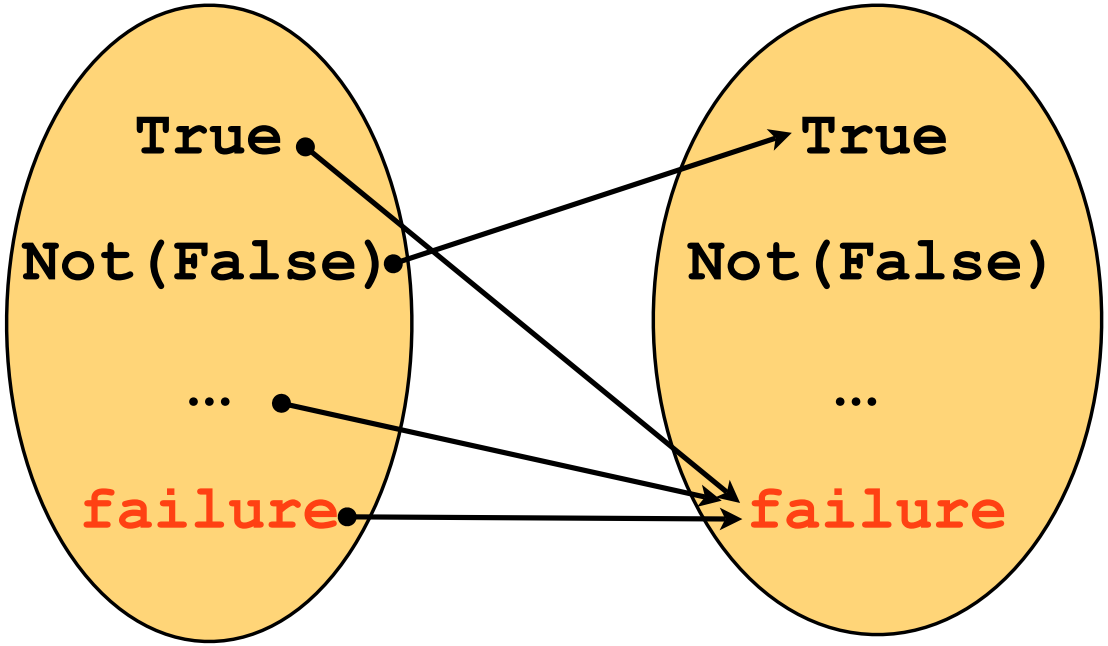

```
True And True
```

A transformation is...

a partial function from terms to terms



Alternatively... failure!



Meta-variables in transformations

```
True And x -> x
```

Applying a transformation, success

```
True And x -> x
```

```
True And True
```


```
True
```

Pattern-matching and replacement

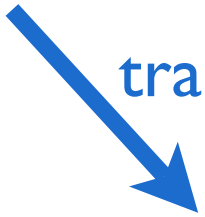
```
True And x -> x
```

Term	Match
`True And True`	`x = True`
`False And True`	no match
`True And (True Or False)`	`x = (True Or False)`
`Not(True And False)`	no match

Applying a transformation, failure

```
True And x -> x
```

```
False And True
```

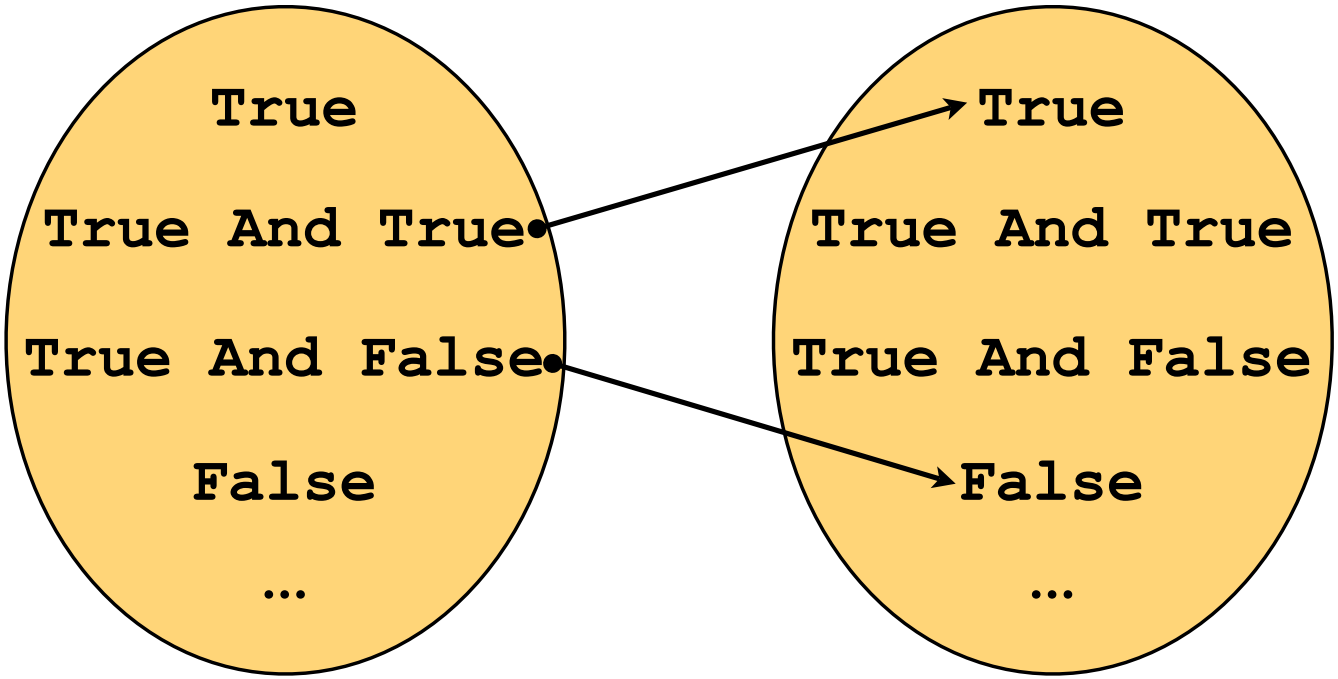


transformation tool

failure!

One transformation mapping several terms

```
True And x -> x
```



Java transformations in Stratego/XT

```
    |[ if (e) stm ]|
-> |[ if (e) {stm} ]|
```

`stm` (`stm1`, ...) and `e` (`e1`,...) are predefined meta-variables

Transformations = rewrite rules

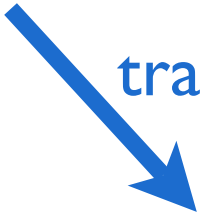
```
rules
  AddBlockToIf:
      |[ if (e) stm ]|
    -> |[ if (e) {stm} ]|

  AddBlockToIfElse:
      |[ if (e) stm else stm1 ]|
    -> |[ if (e) {stm}
          else {stm1} ]|
```

Applying a transformation, bug!

```
    |[ if (e) stm ]|
->  |[ if (e) {stm} ]|
```

```
if (a<5) {a = a+1;}
```



transformation tool

```
if (a<5) {{a = a+1;}}
```

Conditional transformations

```
rules
  AddBlockToIf:
       |[ if (e) stm ]|
    -> |[ if (e) {stm} ]|
    where
       <notBlock> stm
```

Applying a conditional transformation, failure

```
    |[ if (e) stm ]|
-> |[ if (e) {stm} ]|
where <notBlock> stm
```

```
if (a<5) {a = a+1;}
```

transformation tool

failure!

Applying a conditional transformation, success

```
    |[ if (e) stm ]|
-> |[ if (e) {stm} ]|
where <notBlock> stm
```

```
if (a<5) a = a+1;
```


```
if (a<5) {a = a+1;}
```

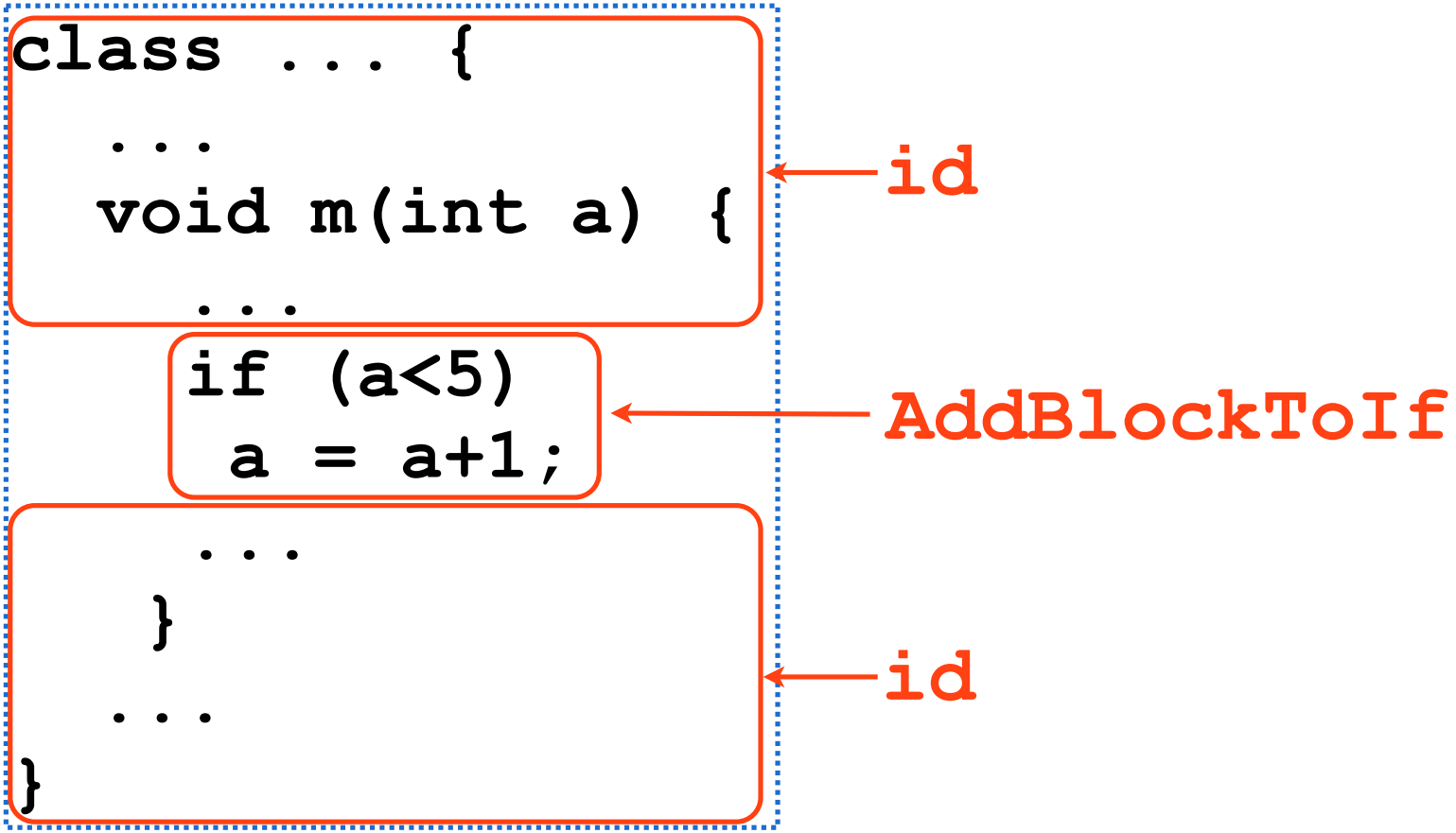
Applying to a program
(not exhaustively)

```
    |[ if (e) stm ]|
-> |[ if (e) {stm} ]|
where <notBlock> stm
```

```
...if (a<5) a = a+1;...
```

failure!

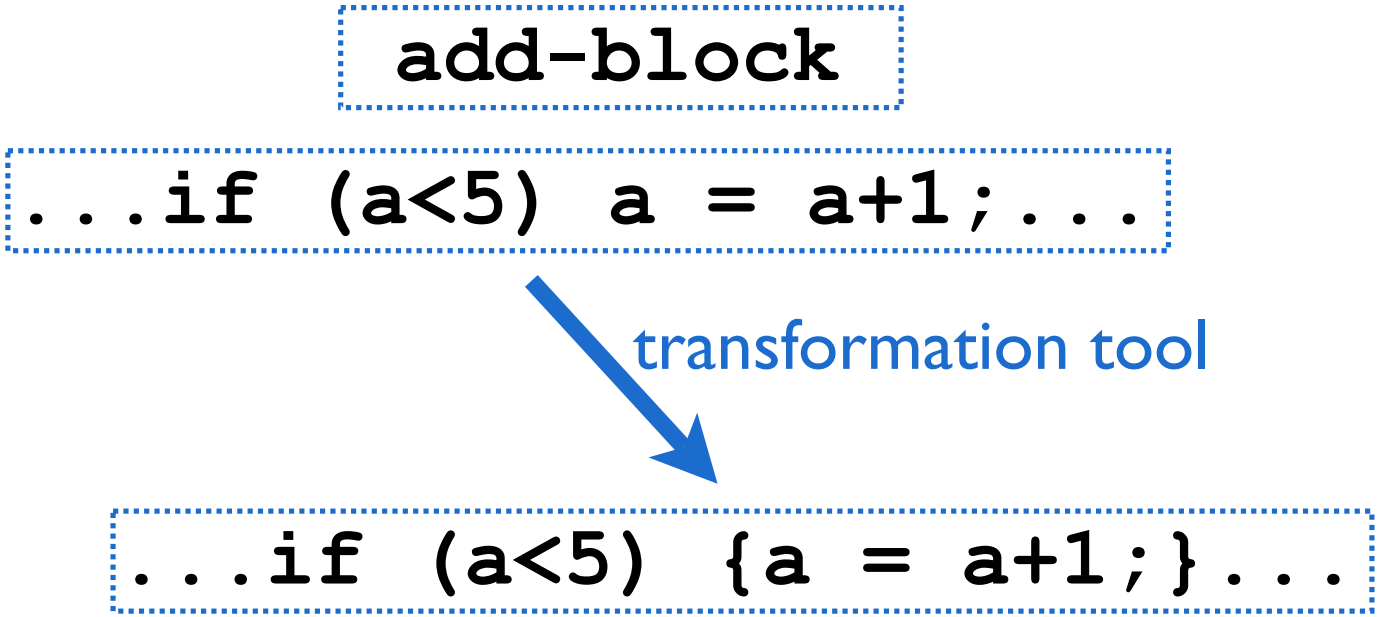
Combining transformations



Transformation strategies = complex transformations

```
strategies
  add-block =
    topdown(try(AddBlockToIf))
```

Applying strategies



Strategies and rules

```
strategies...
   add-block =
      topdown(try(AddBlockToIf))
rules
   AddBlockToIf:
         |[ if (e) stm ]|
     -> |[ if (e) {stm} ]|
     where
         <notBlock> stm
```

Conditions as strategy applications

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

rules
  AddBlockToIf:
      |[ if (e) stm ]|
    -> |[ if (e) {stm} ]|
    where
      <notBlock> stm
```

bstm is a predefined meta-variable for block statements

No ambiguity now!

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

rules
  AddBlockToIf:
      |[ if (e) stm ]|
    -> |[ if (e) {stm} ]|
    where
      <notBlock> stm
```

Modules

```
module add-block-if
imports libstratego-lib libjava-front
strategies
  main =
    io-java2java-wrap(add-block)
  add-block =
    topdown(try(AddBlockToIf))
  notBlock = not(?(stm |[ {bstm*} ]|))
rules
  AddBlockToIf:
      |[ if (e) stm ]|
    -> |[ if (e) {stm} ]|
    where
      <notBlock> stm
```

Predefined strategy operators so far

- `id`
- `?(_)`
- `not(_)`
- `topdown(_)`
- `try(_)`
- `io-java2java-wrap(_)`

Applying more than one rule...

nondeterministic choice operator

```
strategies
  ...
  add-block =
    topdown(try(AddBlockToIf +
                AddBlockToIfElse))
rules
  AddBlockToIf: ...
  AddBlockToIfElse: ...
```

with auxiliary strategy...

```
strategies
  ...
  if-rules = AddBlockToIf +
             AddBlockToIfElse
  add-block =
    topdown(try(if-rules))
rules
  AddBlockToIf: ...
  AddBlockToIfElse: ...
```

or with same rule name...

```
strategies
   ...
   add-block =
      topdown(try(AddBlockToIf))
rules
   AddBlockToIf: ...
   AddBlockToIf: ...
```

rules are tried (undefined order) until one succeeds, or all fail

same holds for strategies

Basic transformation concepts

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

Software transformation and generation

Paulo Borba
Informatics Center
Federal University of Pernambuco

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