Software transformation and generation

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

Strategy operators and lists of terms

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

Strategies for dealing with (property) lists

- **`map(s)`**, succeeds if **`s`** succeeds for each element of the list, yields the modified list of terms
- **`filter(s)`**, applies **`s`** to each element of a list, yielding a list with the elements resulting from the successful application of **`s`**
- **`concat`**, turns a list of lists into a list of elements

Composing strategies that manipulate lists of terms

```
strategies
  generate-gets-and-sets =
    map(property-to-gettersetter) ; concat

  mapconcat(s) = map(s) ; concat

  generate-gets-and-sets-equivalent =
    mapconcat(property-to-gettersetter)
```

Translating entities

```
entity-to-class :
  |[ entity x_Class { prop* } ]| ->
  |[ @Entity public class x_Class {
        public x_Class () { }
        @Id @GeneratedValue private Long id;
        public Long getId() { return id; }
        private void setId(Long id) {
          this.id = id;
        }
        ~*cbds
     }
   ]|
where <generate-gets-and-sets> prop* => cbds
```

Quotation and anti-quotation for Java

```
"|[" ClassBodyDec "]|" -> E {cons("ToMetaExpr")}
"|[" ClassBodyDec* "]|" -> E {cons("ToMetaListExpr")}

"~"  E  -> ClassBodyDec  {cons("FromMetaExpr")}
"~*" E  -> ClassBodyDec* {cons("FromMetaExpr")}

variables
  "stm"  [0-9\']*      -> Stm         {prefer}
  "bstm" [0-9\']* "*"  -> BlockStm*   {prefer}
```

ASTs and signatures

```
module Entity
signature
  constructors
    SimpleSort : Id -> Sort
    Property   : Id * Sort -> Property
    Entity     : Id * List(Property) -> Entity
               : Entity -> Definition
    Model      : List(Definition) -> Model
               : String -> Id
```

Concrete and abstract syntax in rules

```
property-to-gettersetterConcrete:
  |[ x_prop : srt ]| -> ...
property-to-gettersetterAbstract:
  Property(x_prop,srt) -> ...
```

```
addBlockAbstract:
  If(e, stm) -> If(e, Block([stm]))
    where <not(?Block(_))> stm
addBlockConcreteAbstract:
  If(e, stm) -> |[ if (e) {stm} ]|
    where <not(?Block(_))> stm
```

Basic strategies

- `id`, returns the term to which it is applied
- `fail`, always fails
- `?(t)`, fails if the term to which it is applied does not match `t`, otherwise returns the term and binds variables
- `!(t)`, yields the term `t`

Basic strategy operators

- `s ; t`, strategy (function) composition, applies `s` to the term to which it is applied and the result is applied to `t`, fails when `s` or `t` fails
- `s < r + t`, guarded left choice operator, if `s` succeeds `r` is applied to its result, else `t` is applied to the original term, fails when `r` or `t` is selected and fails
- `s + t`, choice

Many other operators are a combination...

```
not(s)          = s < fail + id
s1 <+ s2        = s1 < id + s2
try(s)          = s <+ id
repeat(s)       = try(s; repeat(s))
<s> p           = !p ; s
s => p          =  s ; ?p
<s> p1 => p2 = !p1 ; s; ?p2
```

Including rules!

```
L : p1 -> p2 where s
```

is syntactic sugar for

```
L = ?p1; where(s); !p2
```

term variable scope

```
where(s)
```

is syntactic sugar for

```
{x: ?x; s; !x}
```

Strategy operators and lists of terms

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