A Proposal For OCL λ-Expressions

Tony Clark

Middlesex University, UK

September 29, 2014

The Proposal

- OCL lacks abstraction mechanisms: *e.g.* define a sort where the predicate is an argument.
- OCL lacks modularity: local functions.
- OCL has arbitrary iteration expressions: why not *foldr* and *foldl*?
- Many collection functions can be defined functionally.
- Complex queries and aggregations can be long-winded.
- Can build libraries of functions.

Proposal: anonymous functions as sub-type of OCLExpression, recursive let, function-types, addition of functions to classes and data types, de-sugar loop expressions.

Examples

Define anonymous functions:

```
let rec add = fun(x:Integer,y:Integer):Integer x + y in add(10,20)
```

can be sugared:

```
let add(x:Integer,y:Integer):Integer = x + y in add(10,20)
```

could be recursive:

```
let rec size(s:Sequence(T)):Integer =
          if s->isEmpty then 0 else 1 + size(s->rest())
in size(Seq{1,2,3,4,5})
```

can be added to types:

```
context Sequence(T)::size():Integer =
  if self->isEmpty then 0 else 1 + s->rest()->size()
```

explains iterators:

```
S->collect(x:T | p(x)) becomes S->collect(fun(x:T):Boolean p(x))
context Sequence(T)::collect(q:(T)->Boolean):Sequence(T) =
  let s:Sequence(T) = self->rest()->collect(q)
  in if q(self->head()) then s->prepend(self->head)
     then self->first())
     else s
```