Extending OCL Operation Contracts with Objective Functions

Matthias P. Krieger[1] Achim D. Brucker[2]

[1]Université Paris-Sud

[2]SAP Research

OCL 2011
June 29, 2011

Outline

OCL Has Some Limitations

- OCL expressions can usually be evaluated (relatively) efficiently
- OCL does not have:
  - Powerset operation
  - Unbounded quantifiers
  - ...

\+ Useful tool support is achievable

– Some things are hard to specify in OCL

- Optimization: Result is "at least as good as" *all* possible results
  Often difficult to specify in a postcondition

Easier Specification of Optimization Problems

Idea: preserve OCL expression language
Extend operation contracts with objective functions

```
context Graph::vertexCover(): Set(Vertex)

post: vertices->includesAll(result)
post: result->collect(incident) = edges

minimize: result->size()
```

For existing applications of OCL, objective functions can be ignored.

Optimization Problems are Everywhere

Optimization problems covered in Sedgewick, *Algorithms*:

- Closest pair among a set of points
- Minimum spanning tree of a graph
- Shortest path in a graph
- Maximum network flow
- Maximum matching of a graph
- Regression: Least Squares
- Knapsack problem
- Linear programming

Objective Functions are Even More Useful

Further application: Problems that do not always have a solution

```
context C::op(): T
pre: P
post: Q
```

What if the postcondition Q may be unsatisfiable
even if the precondition P is true?

```
context C::op(): T
pre: P
post: not result.oclIsUndefined() implies Q
minimize: if result.oclIsUndefined()
             then 1
             else 0
          endif
```

Example: Graph Search

Return empty sequence if there is no path

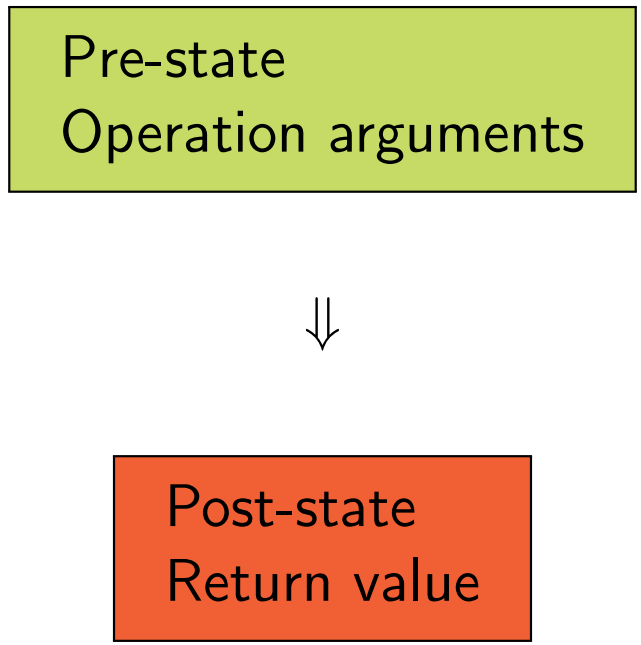
```
context Graph::findPath(start: Vertex, end: Vertex)
                    : Sequence(Vertex)

post: result->notEmpty() implies
          result->at(1) = start
          and result->at(result->size()) = end

post: Sequence{1..result->size()-1}->forAll(i |
          result->at(i).adjacent
             ->includes(result->at(i+1)))

minimize: if result->isEmpty()
             then 1
             else 0
          endif
```

Tool Support: Animation of Operation Contracts

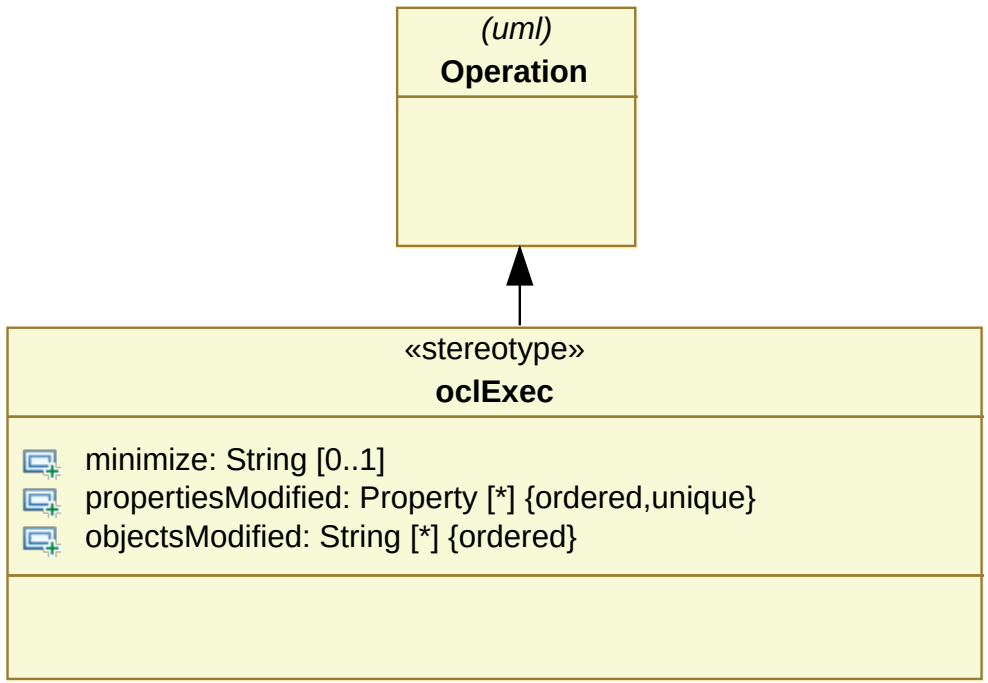


In essence: solving of constraints expressed by the postconditions

Objective Functions are Essential for Animation

Usually only (quasi-) optimal results are useful
Similar case: frame conditions

UML profile with objective functions and frame conditions:



Animation with Objective Functions

- Even without objective functions, operation contract satisfiability is not decidable.
- If valid operation results exist, they can be found by systematic enumeration.
- With objective functions, *satisfaction* of operation contracts is undecidable.
- Optimality can sometimes be proved by incomplete methods.
- General solution: specify a timeout for animation

Conclusion

- Objective functions are useful
  - for specifying optimization problems
  - and more
- Painless extension via UML profile
- Tool support for animation available