An OCL Map Type

Edward Willink

Willink Transformations Ltd
Eclipse Foundation
MMT Component co-Lead
OCL Project Lead
QVTd Project Lead
QVTo Committer
OMG (Model Driven Solutions)
OCL 2.3, 2.4, 2.5 RTF Chair
QVT 1.2, 1.3, 1.4 RTF Chair

OCL 2019 @ MODELS 2019

16th September 2019

OCL Aggregate Types

- Collections (Bag, OrderedSet, Sequence, Set)
- Tuples
- Nested Collections
- Type Constructors (shadow types)
- ?? Map ??
  - use Set(Tuple(key, value))
  - No
    - Set(Tuple(key, value)) has unique key+value pairs
    - Map(key, value) has unique keys with associated 'co-value'

Map philosophy

- Java Map very familiar - mutable
  - put(k, v) modifies the map
  - get(k) / contains(k) interrogates the map
    - get() returning null is ambiguous: null-value-hit/key-miss
- OCL Map - immutable
  - 'mutation' => new Map with changed contents
    - including(k,v) rather than put(k,v)
    - at(k) rather than get(k)
      - returns invalid for a miss
      - returns null for a null value
      - semantically identical to OrderedCollection.at(Integer)
    - includes(k) rather than contains(k)

Map in Eclipse OCL (2015-06)

- **Map**(K,V) type declaration
- **Map{**k1**<-**v1, k2**<-**v2**}** literal expression
- Operations
  - isEmpty(), notEmpty(), size(), =, <>
  - at(k), keys(), values()
  - excludes(k), excludes(k,v), excludesAll(ks), excludesMap(m), excludesValue(v)
  - includes(k), includes(k,v), includesAll(ks), includesMap(m), includesValue(v)
  - excluding(k), excluding(k,v), excludingAll(k), excludingMap(m)
  - including(k, v), includingMap(m)

Map implementation Phase 1

- Supports simple Map construction and use
- Any non-trivial Map construction is >= quadratic
  - new Map is constructed for each addition

Map implementation Phase 2

- Support iterated Map construction
  - potentially linear cost
- Support Map iterators over key and co-value

Map in Eclipse OCL (2019-03)

- Collection(T)
  - **collectBy**(k | f(k)) returns Map(k<-f(k))
  - **inverseCollectBy**(k | f(k)) returns Map(f(k)<-k)
- Map(K,V) like-a Set(K) with co-values of type V
  - any(), collect(), collectNested(), exists(), forAll(), isUnique(), iterate(), one(), reject(), select()
  - **collectBy**(k | f(k)) returns Map(k<-f(k))
- co-iterators: iterator-decl **<- co-iterator-decl**
  - aMap->collect(k1**<-v1** | f(k1,v1))
  - aMap->forAll(k1**<-v1**, k2**<-v2** | f(k1,v1,k2,v2))

Future Work

- Collection(T)
  - collectBy(k | kf(k) <- vf(k)) returns Map(kf(k)<-vf(k))
    - makes inverseCollectBy redundant
- Entry(K,V)
  - collectBy(k | let f = f(k) in kf(f) <- vf(f))
    - needs an Entry(K,V) type for "f"
- Multi-map
  - OCL is lossless
    - collectBy collisions => multimap rather than loss