Formal Specifications for Complex Systems (236368)

Tutorial #2

OCL
OCL

• Added to UML class diagrams or state-charts
• Example system specification: LoyaltyProgram system
• Given: UML class diagram (see additional file)
OCL – operations on collections

• This is the unusual part of notation
• Collections: Bag, Set, Sequence

Example:
Bag{1,1,2,5}.AsSet() = Set{1,2,5}
Operations on Collections - examples

- Set\{acc1, acc2, acc3, acc4\} → **select**(balance > 100)
  - Meaning: subset, containing all the accounts with balance > 100
  - Generalized syntax:
    - **select**(a | a.balance > 100)
    - **select**(a : Account | a.balance > 100)
  - The symmetric operation: **reject**

- Set\{1, 2, 3, 4, 5, 6\} → **collect**(x | x mod(2)) = Bag\{1, 0, 1, 0, 1, 0\}
  - Evaluates the expression on each of the items in the collection
  - Generalized syntax:
    - **collect**(x : int | x mod(2))

constant operations on slice

always will appear logical
the result will be a subset
of the original set

always will appear expression
the result will be Bag
since **collect**
can return any value,
with repetitions
the result will be the expression
Operations on Collections - examples

- Set\{1,2,3,4,5\} $\rightarrow \text{includes}(6) = \text{false}$
- Set\{1,2,3,4,5\} $\rightarrow \text{including}(6) =$ Set\{1,2,3,4,5,6\}
- **Excludes, excluding** – symmetric operations
- Set\{Set\{1,2\},Set\{Set\{2,3\},3\},Set\{5,6\}\} $\rightarrow \text{flatten()} =$ Set\{1,2,3,5,6\}
- Set\{acc1,acc2,acc3,acc4\} $\rightarrow \text{forall}(\text{balance}>100)$
- Set\{acc1,acc2,acc3,acc4\} $\rightarrow \text{exists}(a | a.\text{balance}>100)$
Loyalty Program System

- **Loyalty program** – תוכנית גואל
- **Program partners** – חברות משתתפות
- **Customer** – לקוחות
- **Membership** – תואר חבר
- **Customer card** – כרטיס חבר
- **Loyalty account** – חשבון Kunden
- **Service** – שירות
- **Transaction** – עסקה
- **Earning** – איסור
- **Burning** – ציסור
- **Level** – ברווח

236368 Emilia Katz, Shahar Dag
OCL => Natural Language (1)

1. context Customer
   inv title = (if isMale = true then ‘Mr.’
   else ‘Ms.’) Endif)

   **Translation:** A customer’s title is ‘Mr.’ if he is male, otherwise it is ‘Ms.’

2. context LoyaltyProgram
   inv ServiceLevel->
   includesAll(Membership->collect(actualLevel))

   **Translation:** The actual service level of a membership must be a service level offered by the LoyaltyProgram
OCL => Natural Language (2)

3. context LoyaltyAccount
   inv points > 0 implies transactions exists(points > 0)
   Translation: The existence of points in a LoyaltyAccount
   implies at least one transaction that contained points.

4. context ProgramPartner
   inv self.deliveredServices.transactions -
   select(oclType() = Burning) -> collect(points) -> sum() <=
   self.deliveredServices.transactions -
   select(oclType() = Earning) -> collect(points) -> sum()
   Translation: The sum of points burned for a
   ProgramPartner’s offered services must never exceed the
   number of points earned from its services.

This is equivalent to:

deliveredServices -> collect(transactions)
Natural Language => OCL(1)

1. Customers must have a minimum age of 18 years

   context Customer
   inv age( ) >= 18

2. The CustomerCard’s “valid from” date must be earlier then the “valid to” date

   context CustomerCard
   inv validFrom.isBefore(ValidTo)
Natural Language => OCL(2)

3. The printed name on the customer card must be a title followed by the registered name of the customer

```ocl
class CustomerCard
  inv printedName = Customer.title.concat(Customer.name)
```  

4. The number of service levels is exactly 2

```ocl
class LoyaltyProgram
  inv serviceLevel->size() = 2
```
Pre- and Post-Conditions in OCL

1. For isEmpty() operation of LoyaltyAccount (returns true iff there are no points in the account)

   context LoyaltyAccount ::isEmpty( ):Boolean
   pre: -- none
   post : result = (points=0)

2. The Burn(integer i) operation of LoyaltyAccount

   context LoyaltyAccount::burn(i: Integer)
   pre: points >= i and i >= 0
   post: points = points@pre –i
Pre- and Post-Conditions in OCL

• We start off with:

```
• m() was applied and we got:
• a0.b@pre.c@pre = a0.b.c@pre = a0.b.c = 3
• a0.b@pre.c = 4
• a0.b.c@pre = 3
• a0.b.c = 4
```

```
A
m() 

B
b

c:int

a0

b0
c = 3
```

```
a0

b0
c = 4
```
Pre- and Post-Conditions in OCL

• We start off with:

• $m()$ was applied and we got:
  - $a_{0}.b@pre.c@pre = 3$
  - $a_{0}.b@pre.c = 4$
  - $a_{0}.b.c@pre = 5$
  - $a_{0}.b.c = 6$

236368 Emilia Katz, Shahar Dag
Built-in OCL Types

- **OclType**
  - Customer.name() = “Customer”
  - Customer.attributes() = Set{“isMale”, ”title”, ”birthDate”, ”name”}

- **OclAny**
  
  Example - e:Earning
  - e.oclType() = (OclType)Earning
  - e.oclIsKindOf(Transaction) = true
  - e.oclIsTypeOf(Transaction) = false