ATL – ATLAS transformāciju valoda. Specifiskās konstrukcijas un izpilde.

6. lekcija

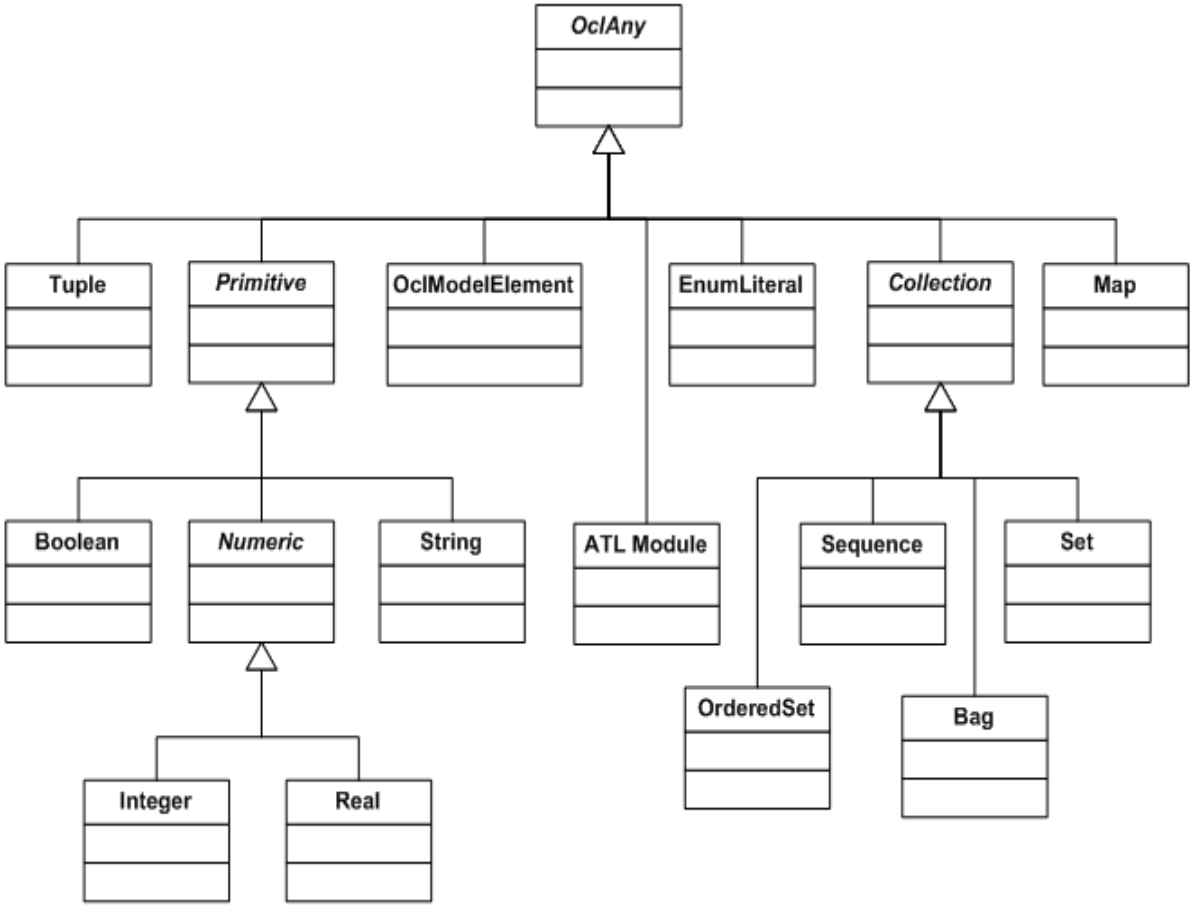
Ērika Asņina, DITF LDI LDK

OclType klases iespējami eksemplāri: datu tipi

OclType ir kā tipu defīnīcija ATL valodā.

Map ir konstrukcijas, kura ir realizēta ATL valodā, bet nepastāv OCL valodā.

ATL module ir ATL vienību (moduļa un vaicājuma) tips.



ATL specifiskās OclAny tipa operācijas

- *output(s : String)* writes the string s to the Eclipse console. Since the operation has no return value, it shall only be used in ATL imperative blocks;
- *debug(s : String)* returns the *self* value and writes the *"s : self_value"* string to the eclipse console;

Datu tips *ATL Module*

- Attēlo ATL vienību
  - Modulis - module
  - Vaicājums – query
- Izpildlaikā eksistē šā tipa VIENS eksemplārs, pie kura drīkst griezties ar
  - *thisModule*
  - Iespējama piekļuve līdzētājiem un atribūtiem

ATL Module tipa *resolveTemp()*

- *resolveTemp(var, target_pattern_name)*
  - *var* – ATL mainīgais, kas satur avota modeļa elementu no kura tiek ģenerēts mērķa modeļa elements
  - target_pattern_name – simbolu virknes (string) vērtība, kura satur mērķa parauga (*to* daļas) elementa nosaukumu, kurš kartē avota modeļa elementu (*var*) uz meklējamo mērķa modeļa elementu.
- Atļauts izsaukt no
  - Jebkura sakrišanas likuma *mērķa parauga* un *do* iedaļām
  - Izsaucama likuma *mērķa parauga* un *do* iedaļām, nodrošinot tā izpilde pēc sakrišanas posma

resolveTemp() izmantošanas piemērs

- rule AtoAnnotedB {
  from a : MMA!A
  to ann : MMB!Annotation (), b : MMB!B (
  annotation <- ann)
  }
- rule ARefToBRef {
  from aRef : MMA!ARef
  to bRef : MMB!BRef (
  ref <- thisModule.resolveTemp(aRef.ref, 'b'))
  }

Primitīvie datu tipi

- **Boolean**
  - **true, false**
- **Integer**
  - **1, -5, 2, 34 ...**
- **Real**
  - **1.5, 3.14 ...**
- **String**
  - **Simbols tiek iekodēts kā viena simbolu *string* - ‘b’**
  - **Virknes pirmā pozīcija (indekss) ir 1 - ‘modelis’**

Būla izteiksmes novērtēšana

- if (*izt1* and *izt2*)
  then ...
  else ...
  endif

```
if(self.attributes->size() > 0 and
    self.attributes->first().attr)
    then ...
```

Šajā gadījuma *izt2* būs novērtēta vienmēr, neatkarīgi no izteiksmes *izt1* vērtības!

ATL specifiskās String tipa operācijas

- *writeTo(fileName : String)*
  - ļauj rakstīt *self* virkni uz failu, identificējamo ar virkni *fileName*. *fileName* var satur vai nu pilnu ceļu līdz failam vai relatīvo \eclipse direktorijam, no kuras ATL rīks ir palaists. Ja fails jau eksistē, tad tā saturs tiek pārrakstīts.
- *println()*
  - raksta *self* virkni uz izeju pēc noklusēšanas, proti, Eclipse konsoli

Kolekcijas

- **Set, OrderedSet, Sequence, Bag**
- Kolekcijas var būt uzskatītas par šablona datu tipu.
- Deklarācija un specificēšana
  - kolekcijas_tips(elementa_datu_tips)
    - elementa_datu_tips var būt arī citas kolekcijas tips
  - kolekcijas_tips{elementi}
  - Set(Integer)
    Set{1,2,3}

Kolekcijas iteratīva apstrāde

- Sequence{8, -1, 2, 2, -3}->
```
iterate(e; res : Integer = 0 |
        if e > 0
        then res + e
        else res
        endif
)
```
- Sequence{8, -1, 2, 2, -3}->
```
select(e | e > 0)->sum()
```

Enumeration datu tips

- OclType tipa elements
- Jābūt definētam transformācijas *avota un mērķa metamodeļos*
- *OCL valodā piekļuve ir*
  - *Dzimums::sieviete*
- *ATL realizācijā ir*
  - *#sieviete*

Tuple datu tips

- Deklarējot kortežu mainīgo pēc shēmas
  TupleType(var_name1 : var_type1, ..., var_nameN : var_typeN)
  datu tipi var būt izlaisti, piemēram divi ekvivalenti ir
  - Tuple{editor : String = 'ATL Eds.', title : String = 'ATL Manual',
    a : MMAuthor!Author = anAuthor}
  - Tuple{title = 'ATL Manual', a = anAuthor, editor = 'ATL Eds.'}
  - Kortežiem nav nosaukuma!
- ATL ir definēta operācija *asMap()*, kura atgriež *map* mainīgo, kurā korteža daļu nosaukumi ir asociēti ar to vērtībām.

Map datu tips (tikai ATL)

- Ļauj pārvaldīt struktūru, kurā katra vērtība ir asociēta ar unikālo atslēgu tas piekļušanai
- Deklarācija

  Map(key_type, value_type)

  Map(Integer, MMAuthor!Author)
- Specificēšana

  Map{(key1, value1), ..., (keyn, valuen)}

  Map{(0, anAuthor1), (1, anAuthor2)}
- Tāpat kā kortežiem, *map* nav nosaukuma!
- Sīkāk http://download.oracle.com/javase/1.4.2/docs/api/java/util/Map.html

Modeļa elementu datu tips

- Atsaukšanai uz elementu izmanto konstrukciju
  metamodelis!klase
- Piekļuve elementa īpašībām
  self.īpašības_nosaukums
- Deklarācija
  <Package1Name>::<Package2Name>::<ClassifierName>

taču ATL nespēj apstrādāt "::", līdz ar to pilns ceļš jāņem pēdiņās, piemēram *MM!"Pakotne1::Pakotne2::Klasifikators"*

- OclAnyl datu tipa operācija *oclIsUndefined()*
  - Atribūtam ar daudzkāršību 0..1 (void vai nē) der, taču
  - Atribūtam ar daudzkāršību 0..n (kolekcijai) neder, jo kolekcija var būt tukša un ne void (inicializēta) vienlaicīgi.

ATL komentāri

- ATL nodrošina vienas rinda komentārus, kuri sākas ar “--“
  - -- Tas ir komentāra piemērs
- Sākot ar ATL 3.1. versiju tiek nodrošinātas līdzētāju un likumu individuāla komentēšana, izmantojot “---”
  - --- Tas ir likuma un līdzētāja komentārs

OCL deklaratīvas izteiksmes *if* un *let*

- if condition
  then exp1
  -- else daļu drīkst izlaist
  else exp2
  endif

- if izteiksme ļauj ietvert citas saliktas OCL izteiksmes, tai skaita operāciju izsaukumus, "let" izteiksmes vai ieliktas "if" izteiksmes.

- "let" ļauj definēt mainīgos, pie tam vairākas let definīcijas var būt lietotas secīgi

  let var_name : var_type = var_init_exp in exp

a) let a : Integer = 1 in a + 1
b) let x : Real = if aNumber > 0 then aNumber.sqrt() else aNumber.square() endif in let y : Real = 2 in x/y
c) let firstElt : Real = aSequence->first() in firstElt.square()
   (izmantots *aSequence->first().square()* atkļūdošanai)

Padomi

- Jāatceras, ka OCL izteiksmēs loģiskajos operatoros tiek aprēķinātas **visas izteiksmes, neatkarīgi no tā vai pirmā izteiksme bija patiesa vai aplama.**

- *if not person.oclIsUndefined() and person.name = 'Vitalijs'*
- *if person.oclIsUndefined()*
  then
  false
  else
  person.name = 'Vitalijs'
  endif

ATL likumu konstrukcijas

- mērķis <- izteiksme;
- if konstrukcija

```
if (nosacījums) {
        izteiksmes1
}
[else {
        izteiksmes2
}]
```

- *else* daļa var būt izlaista

Sakritoša likuma konstrukcija

```
rule rule_name {
        from in_var : in_type [in model_name]? [(
                    condition )]?
        [using {
              var1 : var_type1 = init_exp1;
              ... varn : var_typen = init_expn;
        }]?
        to
            out_var1 : out_type1 [in model_name]? (
                        bindings1
            ),
            out_var2 : distinct out_type2 foreach(e in collection)(
                        bindings2
           ),
            ... out_varn : out_typen [in model_name]? (
                        bindingsn
           )
       [do {
              statements
             }]?
}
```

Lazy likuma konstrukcija un izsaukšana

- Definēšana

```
lazy rule getCross {
  from
        i: ecore!EObject
  to
    rel: metamodel!Relationship ( )
 }
```

- Izsaukšana

```
rule Example {
      from
          s : ecore!EObject
      to
           t : metamodel!Node ( name <- s.toString(),
                                edges <- thisModule.getCross(s) )
}
```

Bibliogrāfija

- ATL/User Guide - The ATL Language http://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Language
- Eclipse rīka palīgfailos
- ATL/User Guide - The ATL Tools http://wiki.eclipse.org/ATL/User_Guide_-_The_ATL_Tools