Bit and Pieces Again

Presentation for CAMPaM workshop

by

Anneke Kleppe

Contents

- OCL
- Using OCL in UML
- Semantics: Denotational meta-modeling
- Grasland: Defining languages
- Concrete textual syntax
- Future work
- Sideline: MDA/MDE

Part 1

Object Constraint Language

Version 2.0

OCL

- What is OCL?
  - Query language on UML models based on set theory and first order logic

```
context
  Program::findType(name: String) :
  Type
body:
  types->any( t : Type | t.name =
  name )
```

2001-2003: OCL version 2

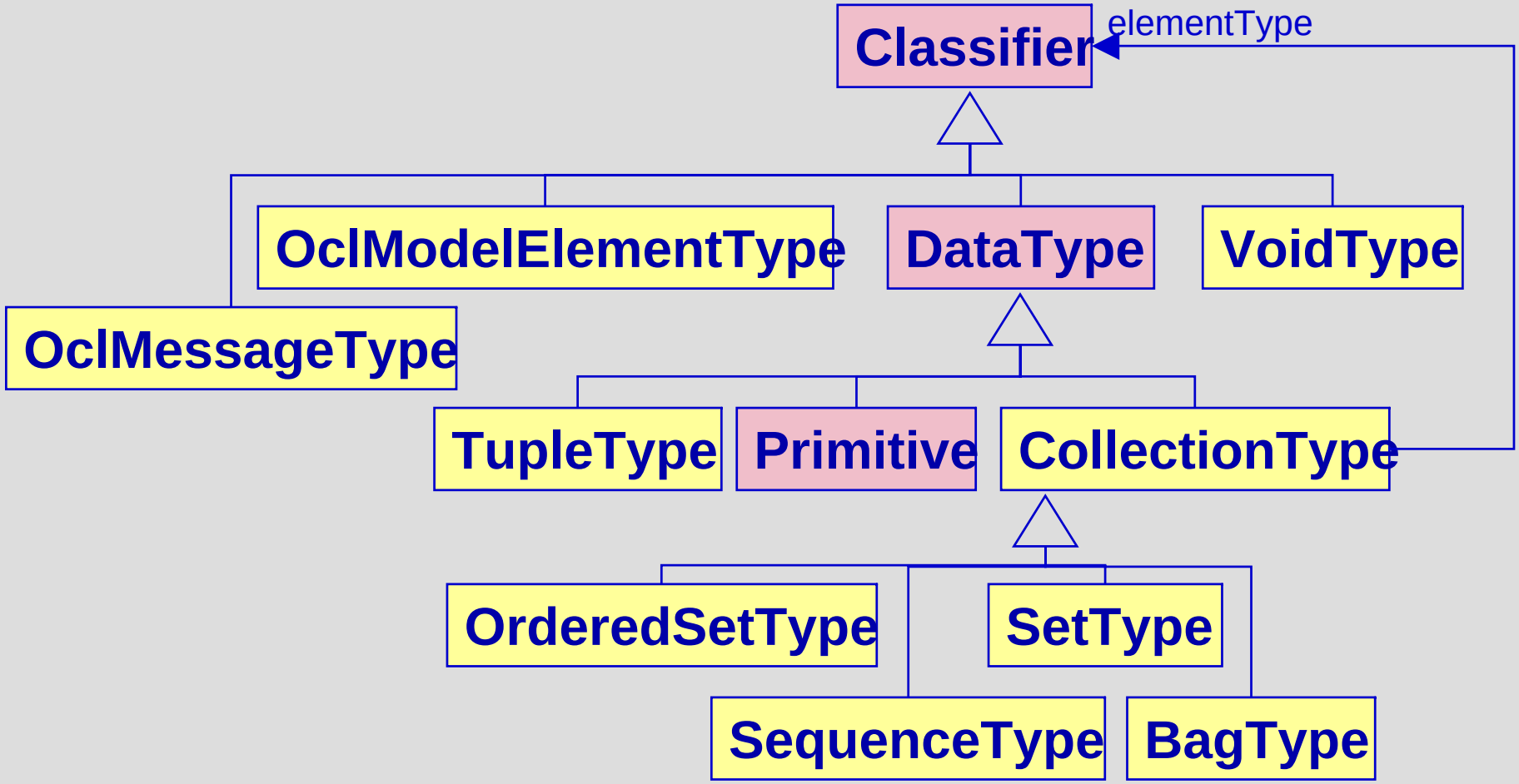
- OMG only wanted definition of abstract syntax
- We gave them:
  - Abstract syntax
  - Concrete syntax, plus mapping to abstract
  - Semantics
    - Formal by Mark Richters, Hamburg Univ.
    - UML-based by AK.

OCL 2.0 Abstract syntax

- Defined using meta-model
- Explicitly defines the associations with meta-classes in the UML meta-model

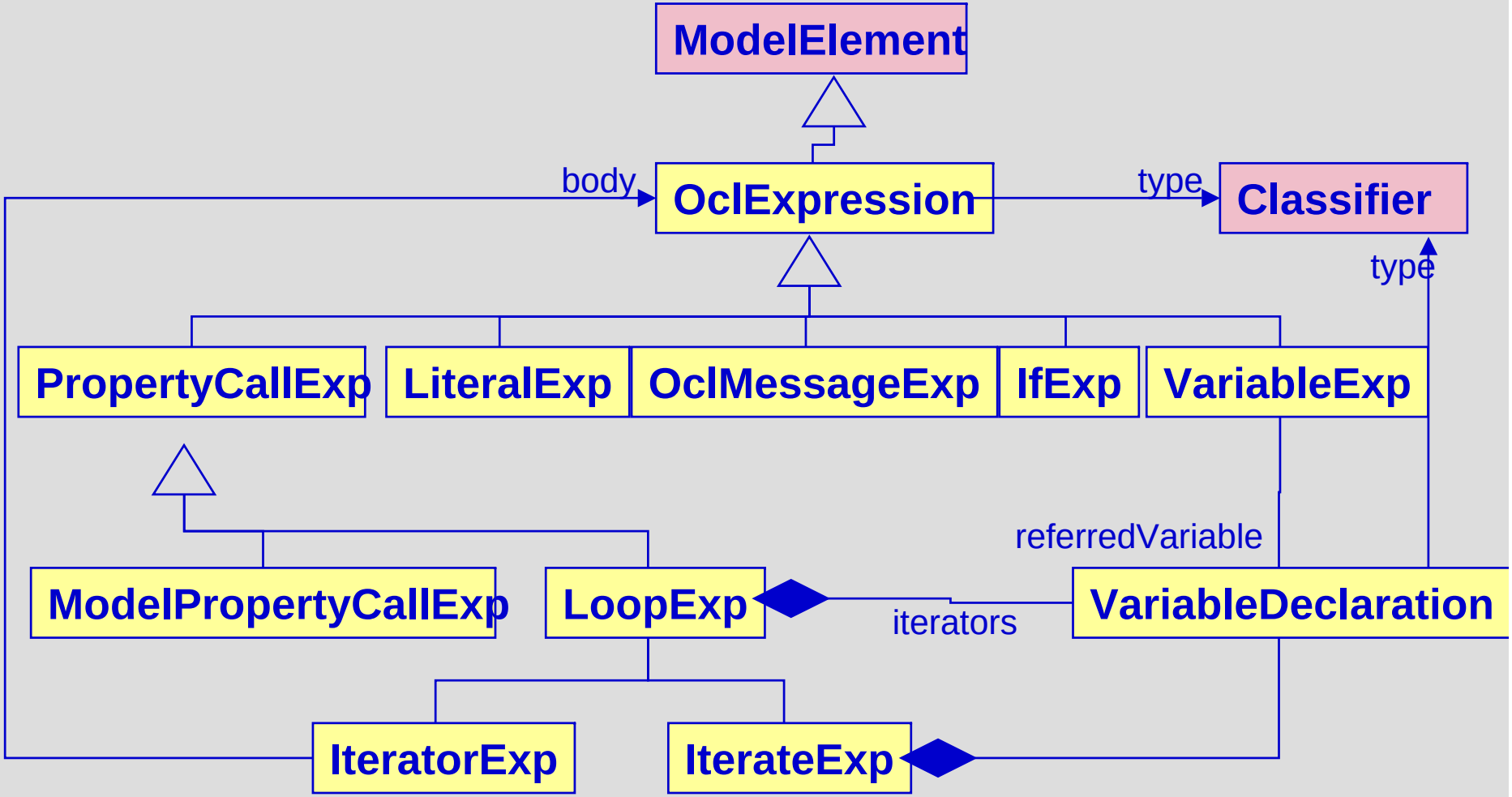
Types Package

University of Twente
Enschede - The Netherlands



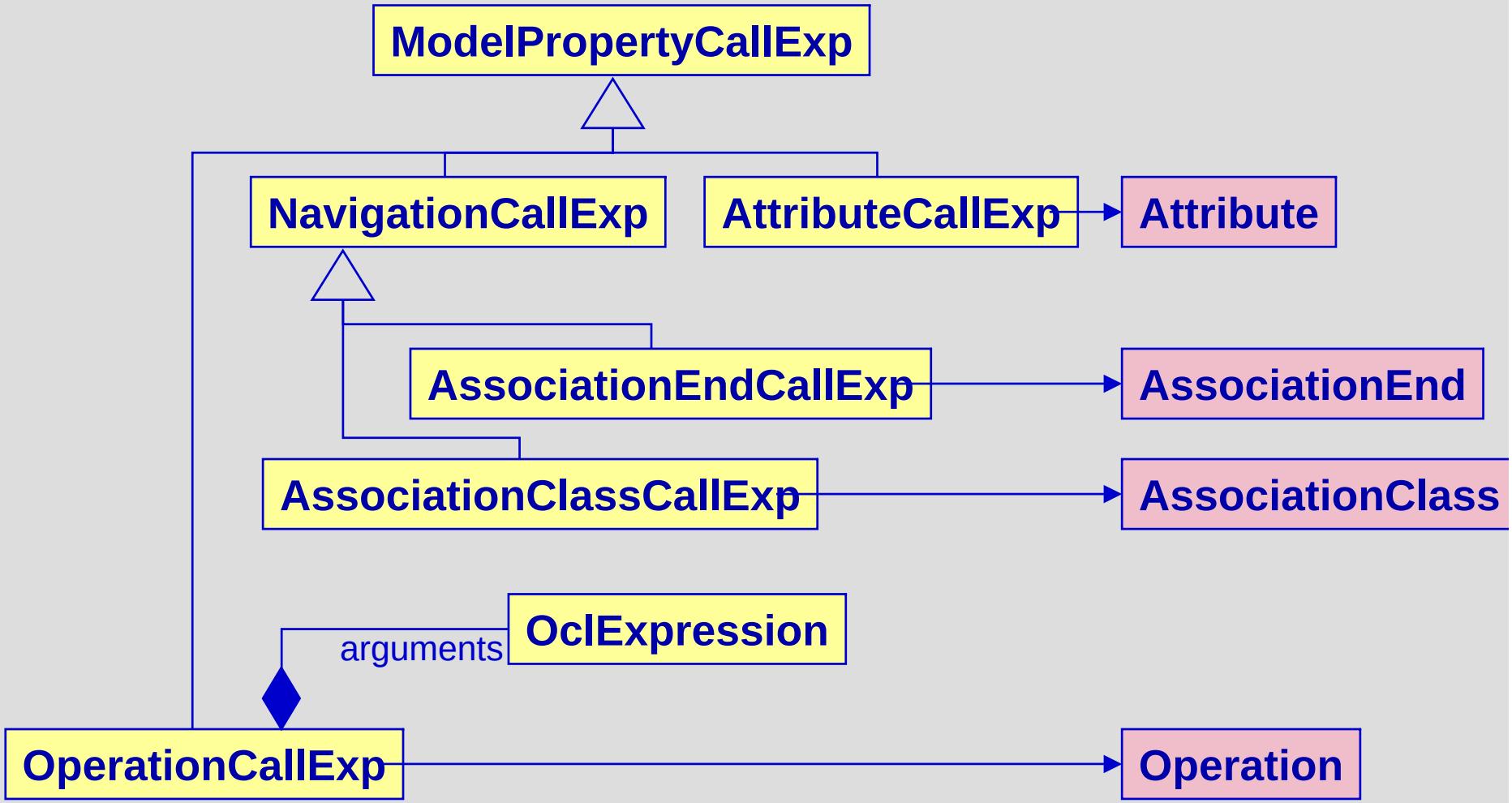
Expressions PACKAGE

University of Twente
Enschede - The Netherlands



Direct Integration With UML

University of Twente
Enschede - The Netherlands

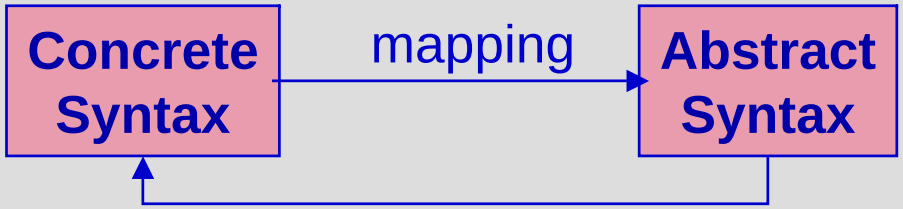


Abstract vs. Concrete Syntax

- This is an example of the concrete syntax:

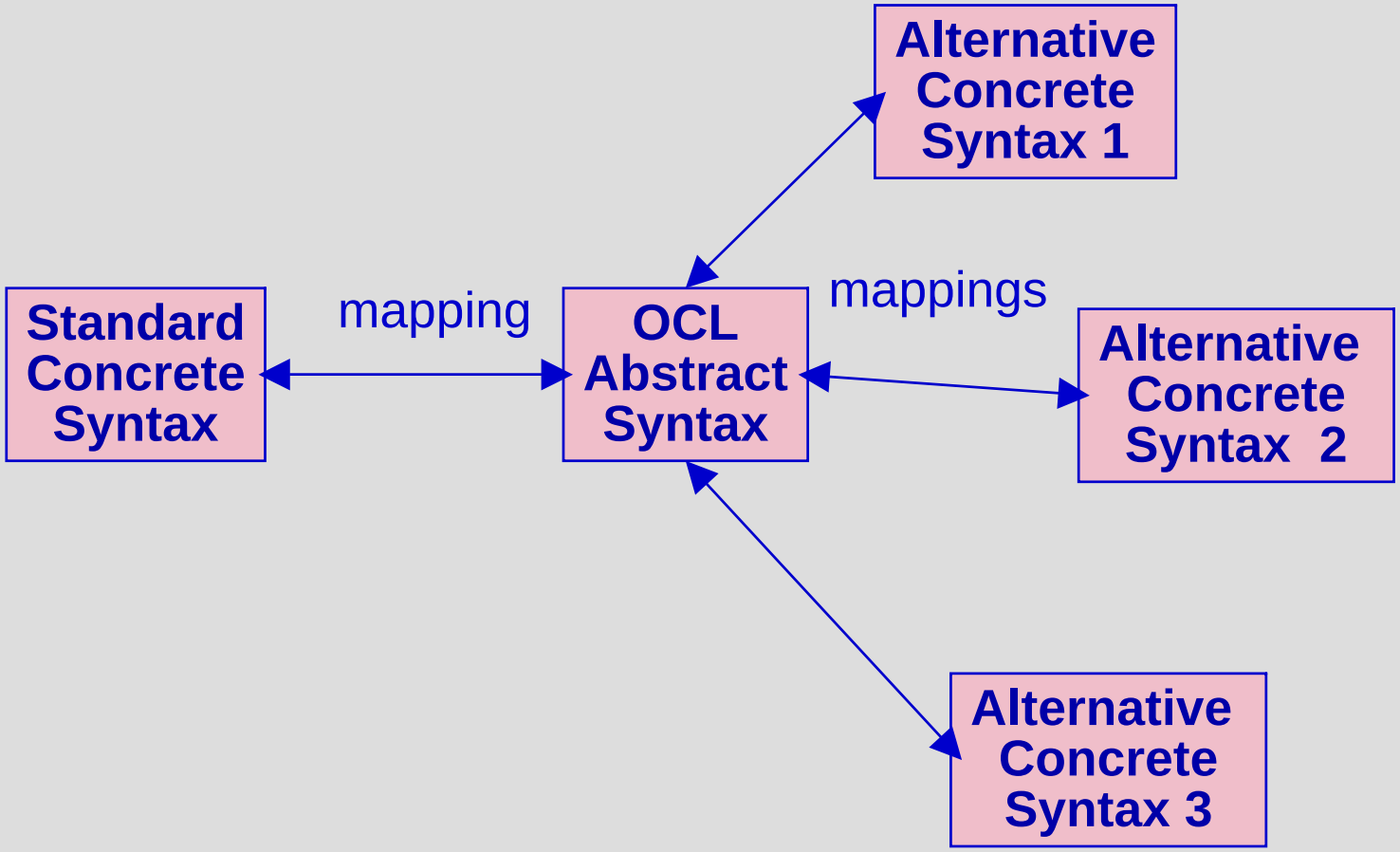
```
context getYoungCustomers() : Set(Customer)
body: customers->select( c | c.age < 18 )
```

- Formal mapping from concrete to abstract syntax using attribute grammar



Abstract vs. Concrete Syntax

University of Twente
Enschede - The Netherlands



Alternative Concrete Syntax

- Business Modeling Syntax:

```
context getYoungCustomers() : Set(Customer)
body: customers->select( c | c.age < 18 )

body: SELECT c : Customer FROM customers
       WHERE c.age < 18
```

- See:
  - The Object Constraint Language, Getting Your Models Ready for MDA
  - Octopus: an Eclipse based IDE for OCL

OCL Text Files

- Concrete syntax for OCL files formalized

```
package OclBoek::RandL

  context LoyaltyAccount::points
  init: 0

  context LoyaltyProgram::getServices(): Set(Service)
  body: partners.deliveredServices->asSet()

  context CustomerCard::myLevel : ServiceLevel
  derive: Membership.currentLevel

endpackage
```

Part 2

Using OCL in UML

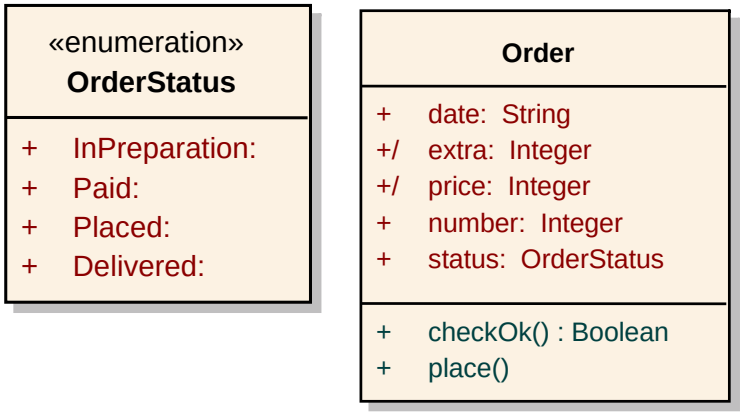
Where to use OCL in UML

- Anywhere UML talks about Expression
- Specific examples:
  - Pre- and post-conditions
  - Invariants
  - Initial values of attributes and associations
  - Derivation rules
  - Bodies of query operations
  - State invariants
  - Definition of additional attributes and operations
  - Guards in state machines
  - Choice or guard expressions in interaction diagrams
  - Choice expressions in activity diagrams
  - Etc. etc.

Pre- and Postconditions

- Used to specify operations

```
context Order::place()
pre : checkOk() and status = OrderStatus::InPreparation
post: status = OrderStatus::Placed
```

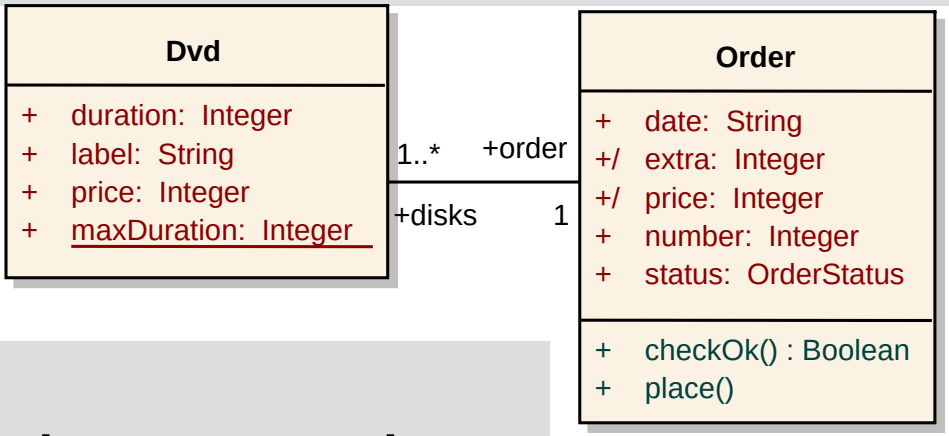


Invariants

University of Twente
Enschede - The Netherlands

- Used to specify invariants

```
context Order
inv: price > 0
inv: disks->isUnique(label)
```

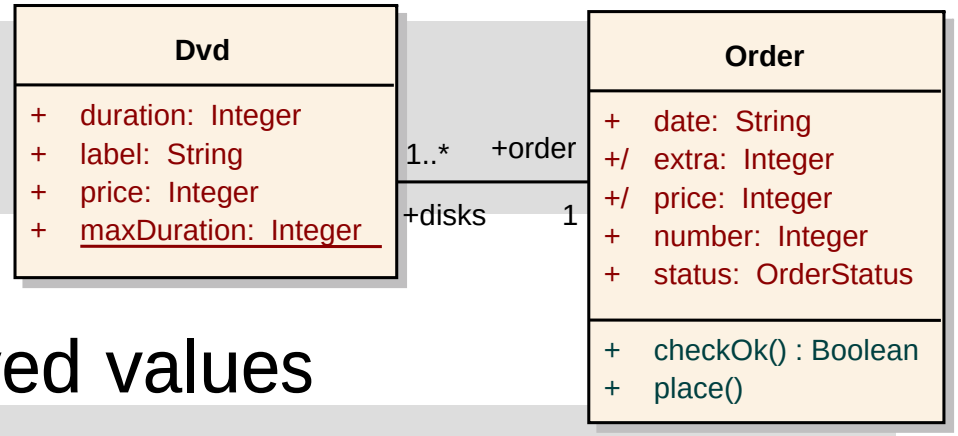



```
context Dvd
inv : duration <= Dvd::maxDuration
```

Initial values and derivation rules

- Used to specify initial values

```
context DVD::label
init: 'default title'
```



- Used to specify derived values

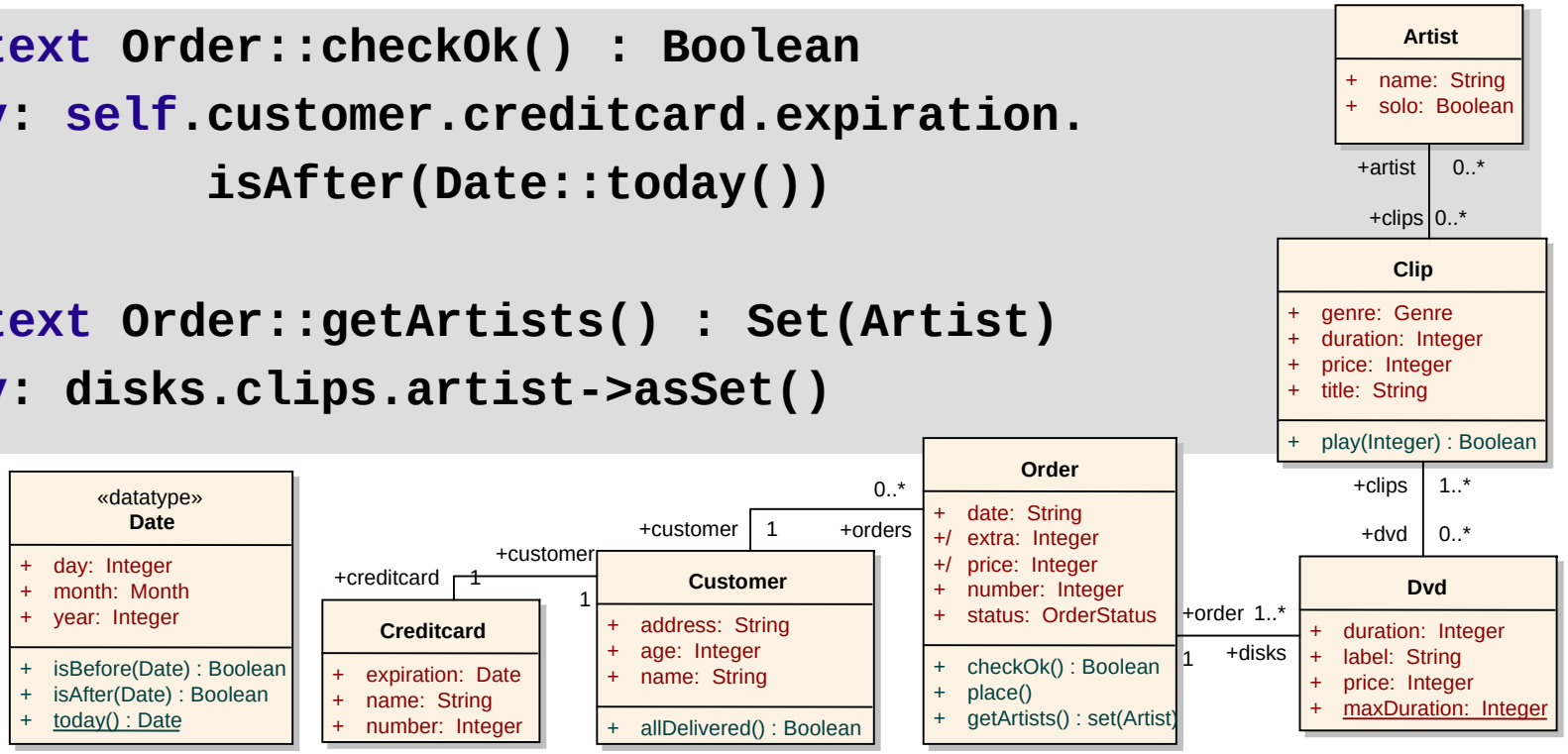
```
context Order::price
derive: disks.price->sum() + extra

context Order::extra
derive: if clips.price->sum() < 15 then 2
                                   else 0
        endif
```

Body values

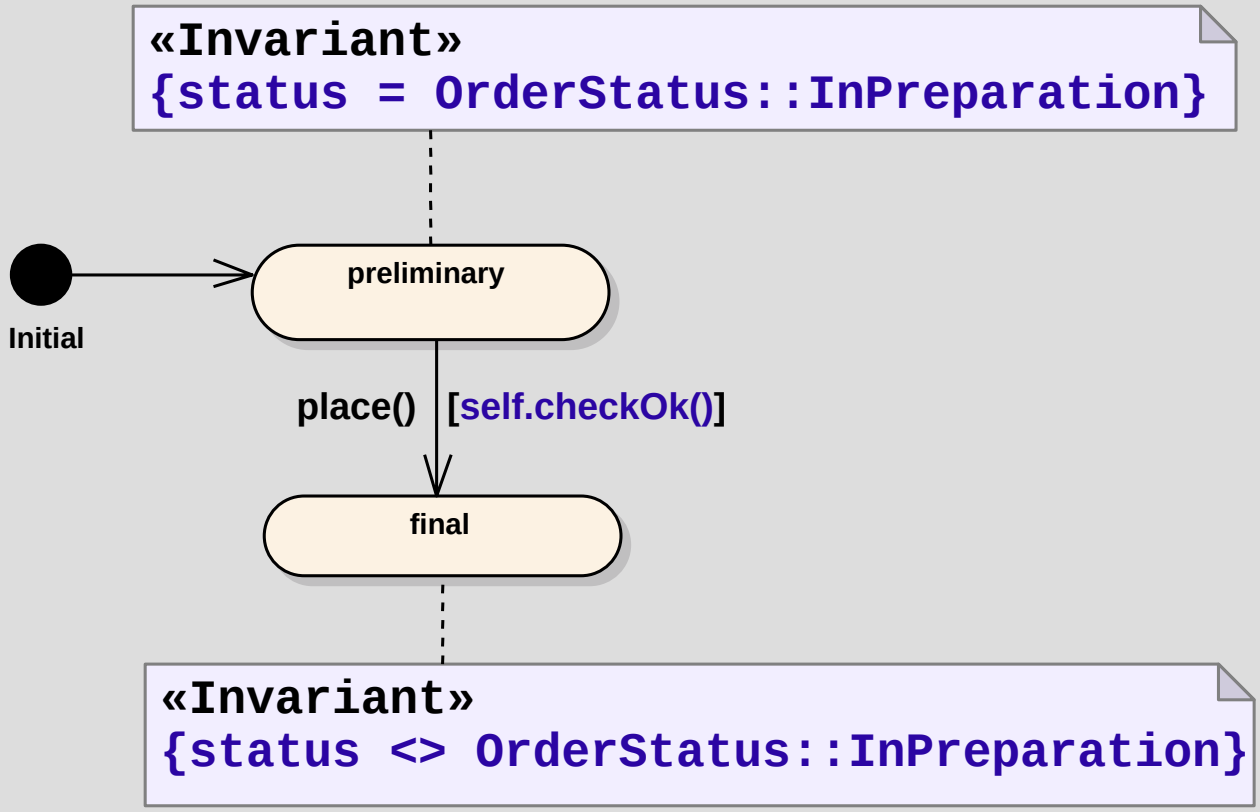
University of Twente
Enschede - The Netherlands

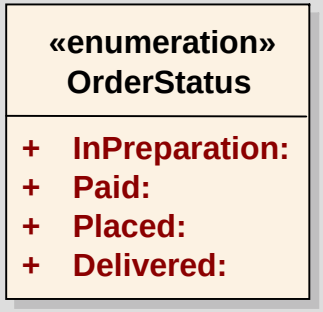
- Used to specify the full body of operations

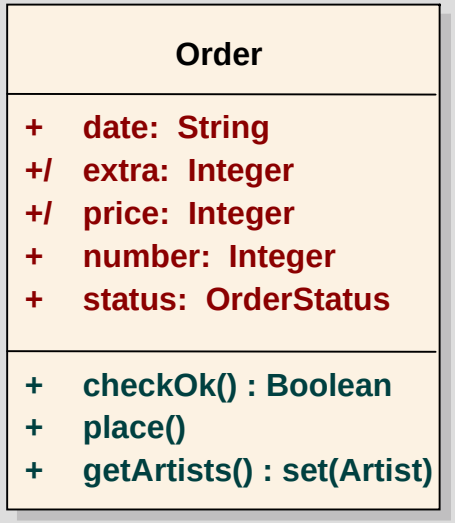
```
context Order::checkOk() : Boolean
body: self.customer.creditcard.expiration.
          isAfter(Date::today())

context Order::getArtists() : Set(Artist)
body: disks.clips.artist->asSet()
```



State charts

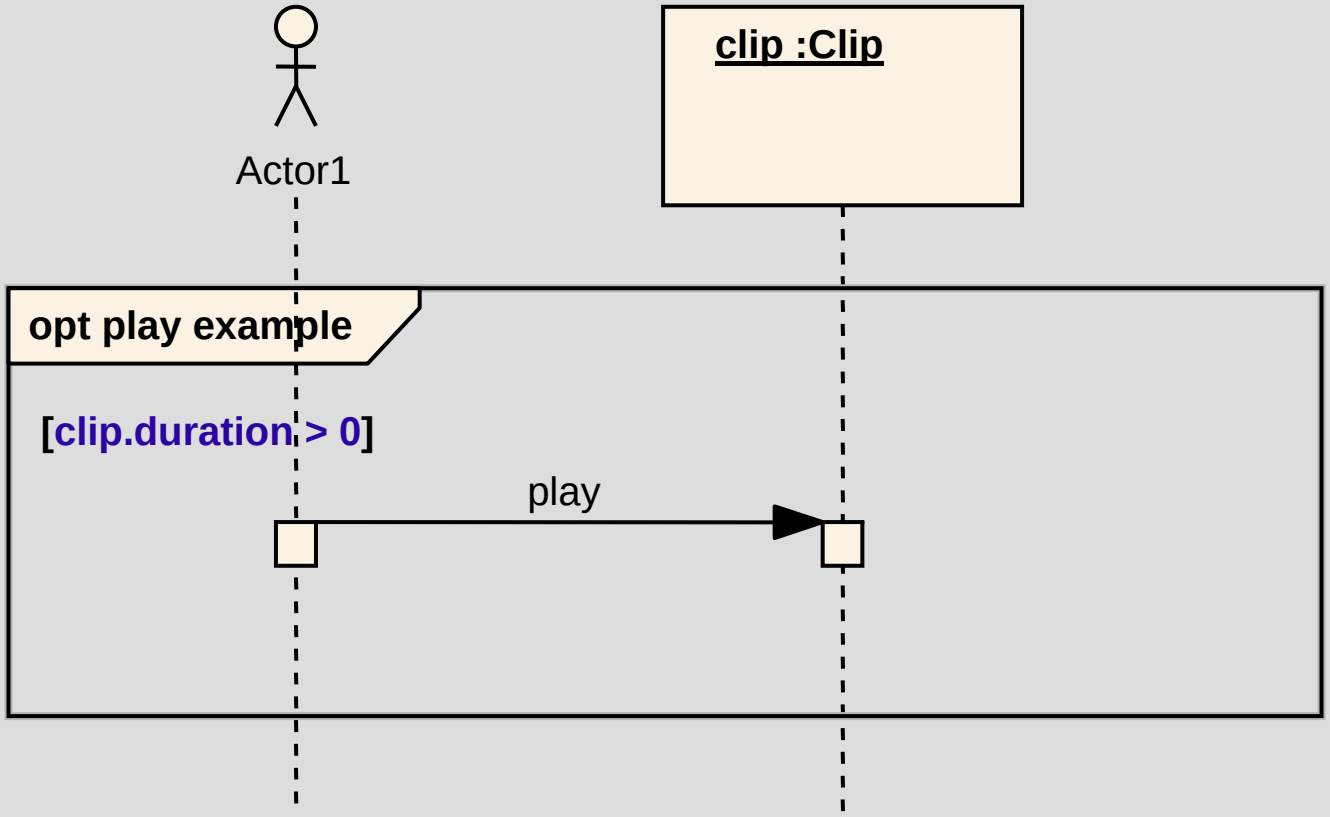






Sequence Diagrams

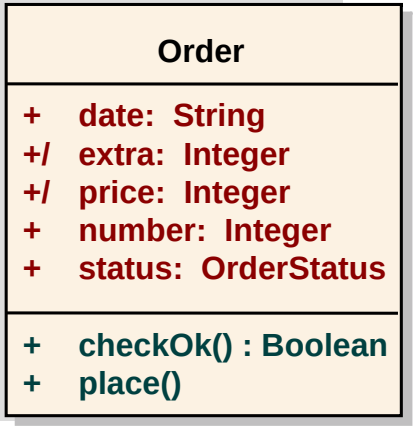
University of Twente
Enschede - The Netherlands



Defining additional attributes

University of Twente
Enschede - The Netherlands

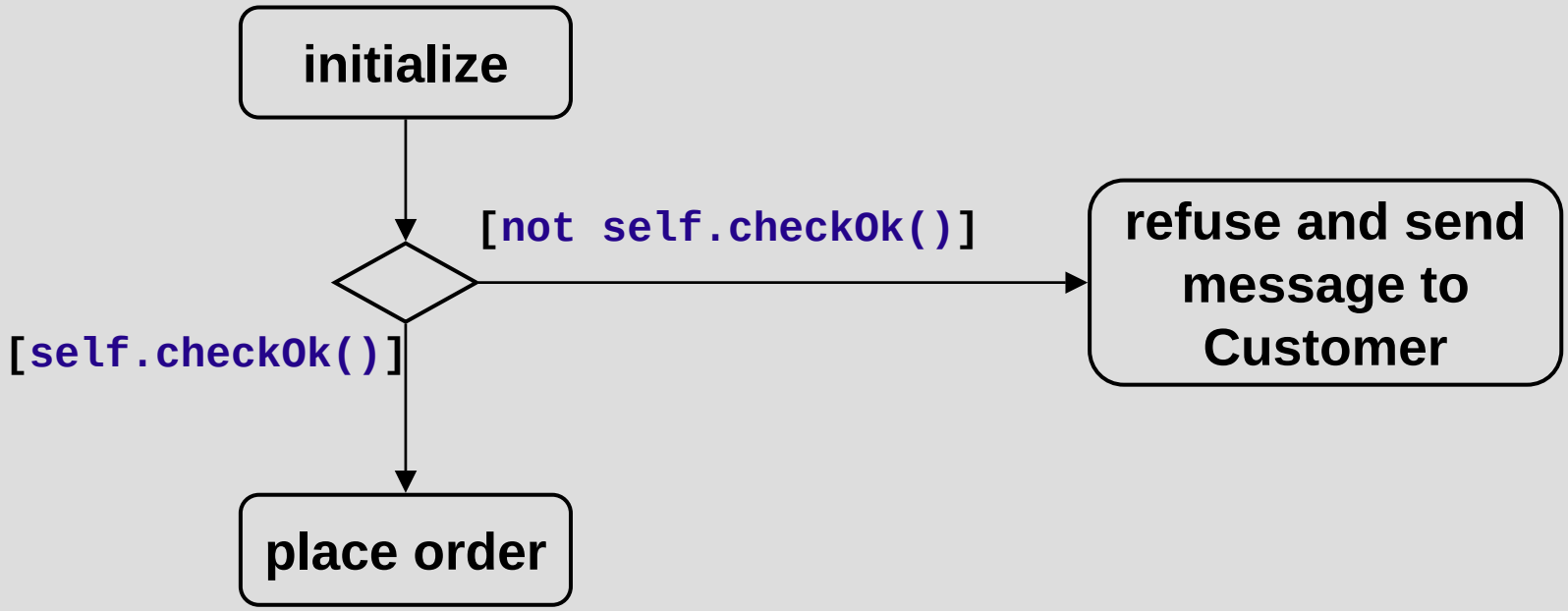
```
context Order::price
derive: disks.price->sum() + extra

context Order::extra
derive: if disks.price->sum() < 15 then 2
                                   else 0
        endif
```




```
context Order
def: basePrice : Integer = disks.price->sum()

context Order::price
derive: basePrice + extra

context Order::extra
derive: if basePrice < 15 then 2 else 0 endif
```

Activity Models

University of Twente
Enschede - The Netherlands



Part 3

Defining semantics

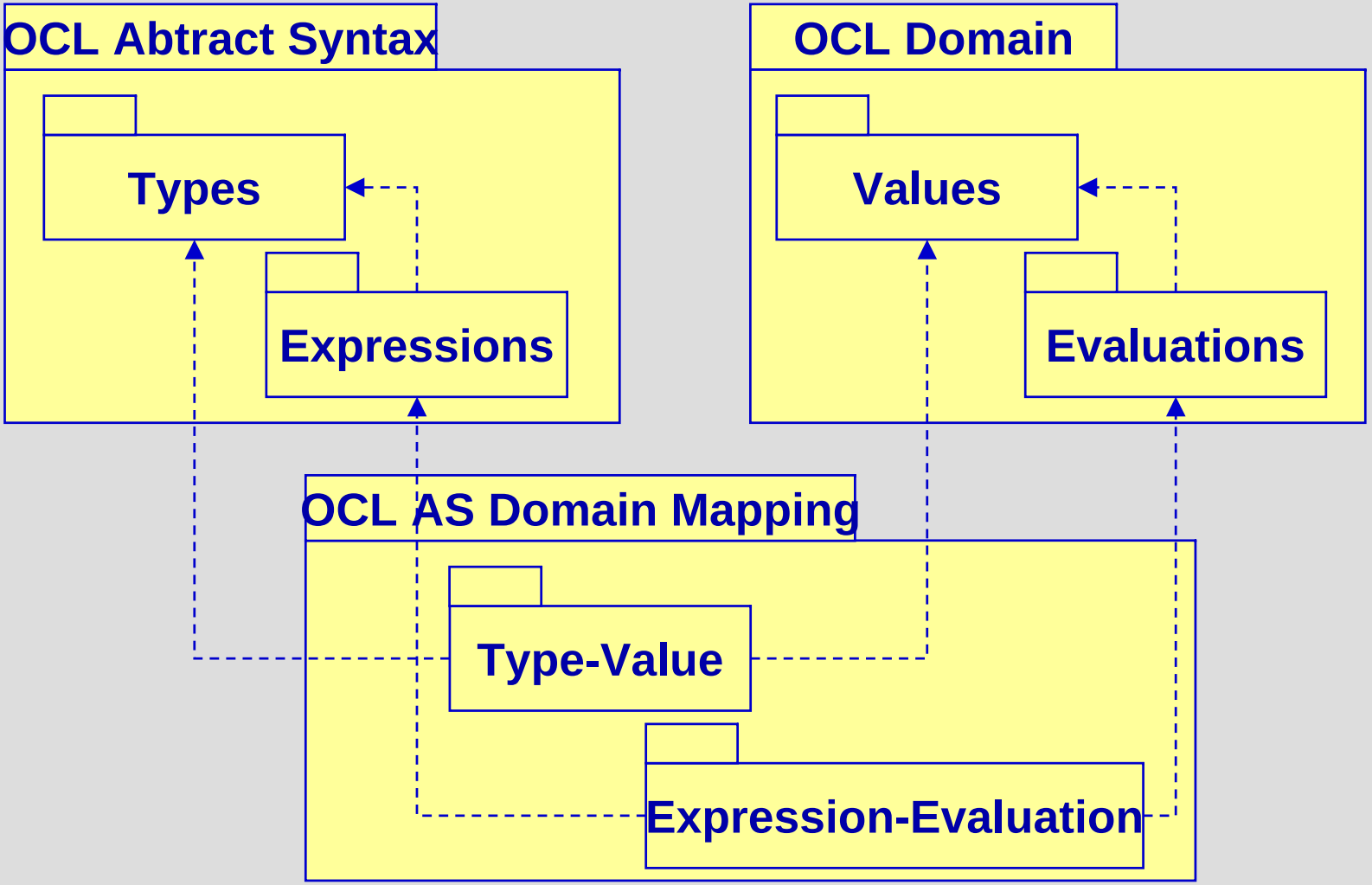
UML-based semantics

University of Twente
Enschede - The Netherlands

- Denotational metamodeling (pUML group)
  - Build UML model of semantic domain and map the abstract syntax to it using UML associations
  - Report: Unification of static and dynamic semantics of UML (http://www.klasse.nl/research/uml-semantics.html)

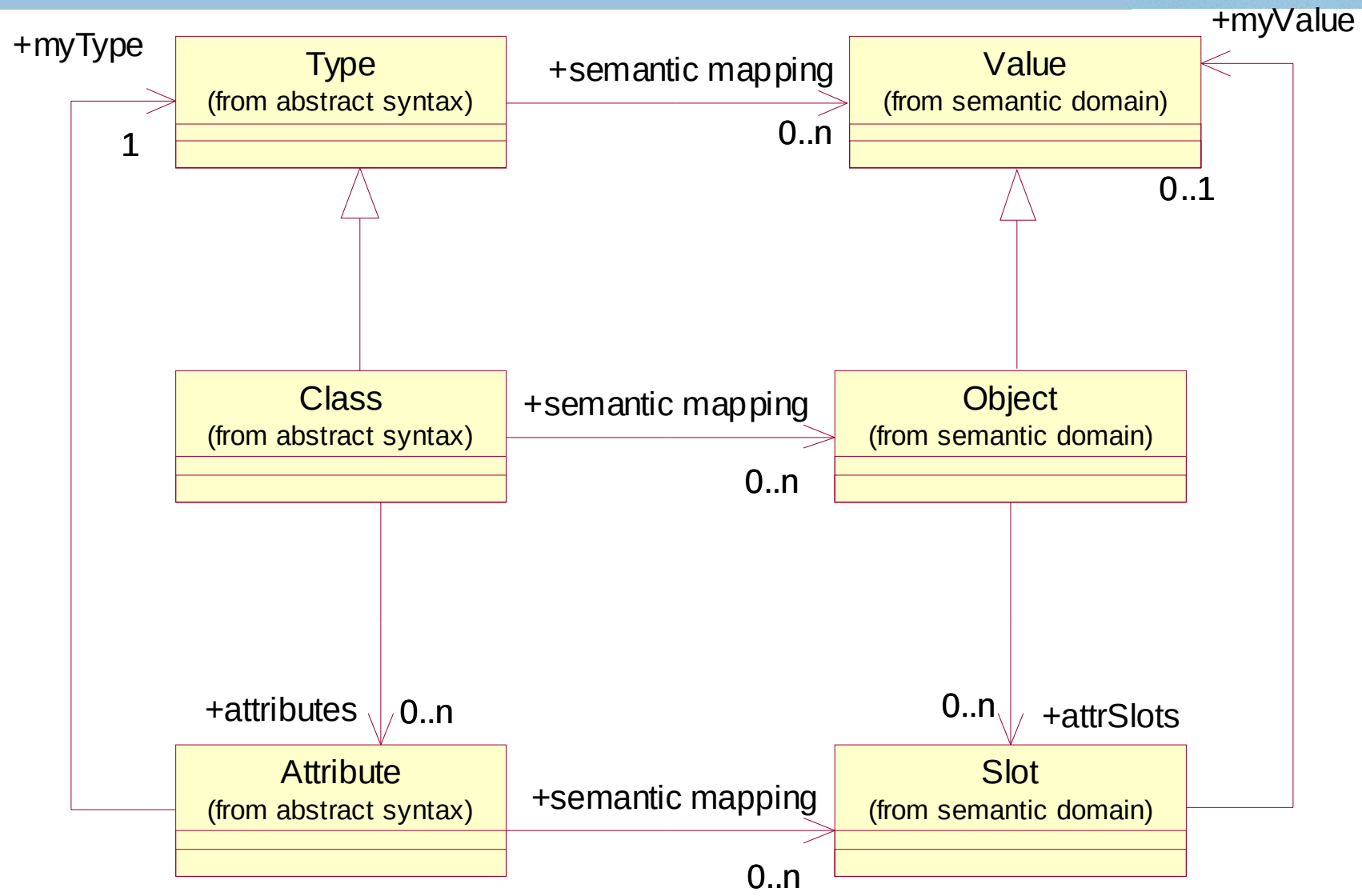
Semantics for OCL

University of Twente
Enschede - The Netherlands



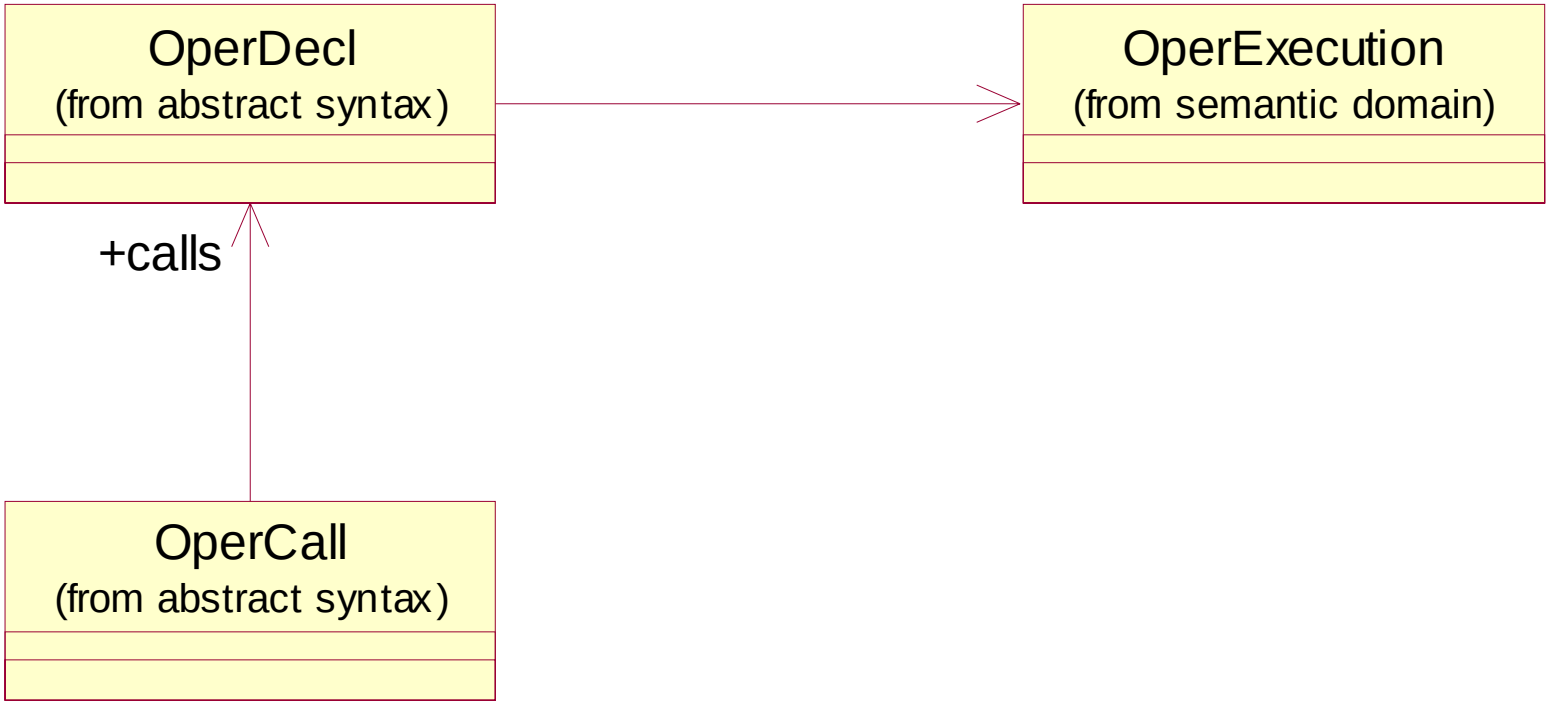
Example 1

University of Twente
Enschede - The Netherlands



Example 2

University of Twente
Enschede - The Netherlands



Weakness

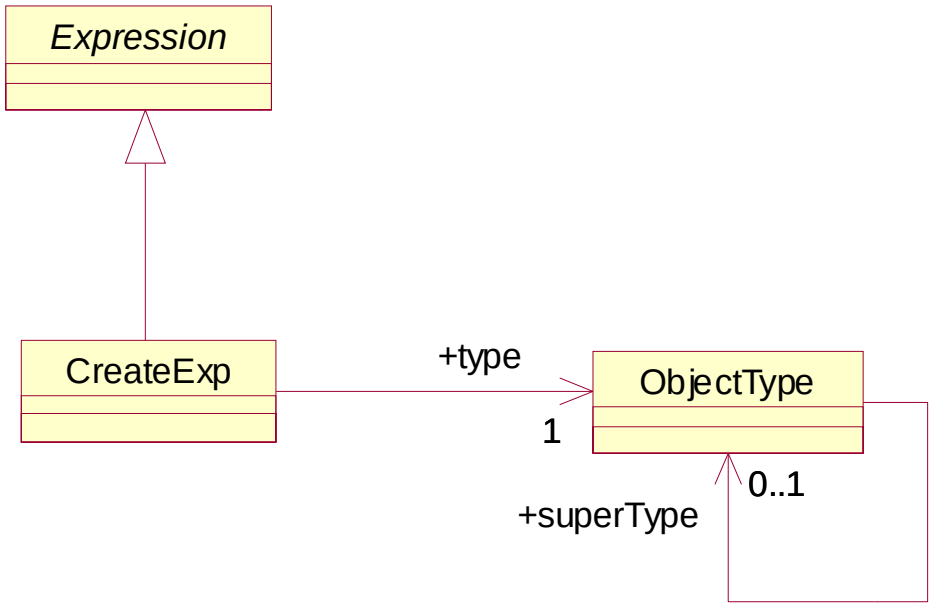
- How to model the dynamics of the semantic domain (SD)?
- Answer J.H. Hausmann: Dynamic Denotational Metamodeling
  - SD model contains operations which are defined by graph transformation rules
  - Needs explicit calling of these operations
- Our answer: Operational Denotational Metamodeling ???
  - TAAL project

TAAL project

University of Twente
Enschede - The Netherlands

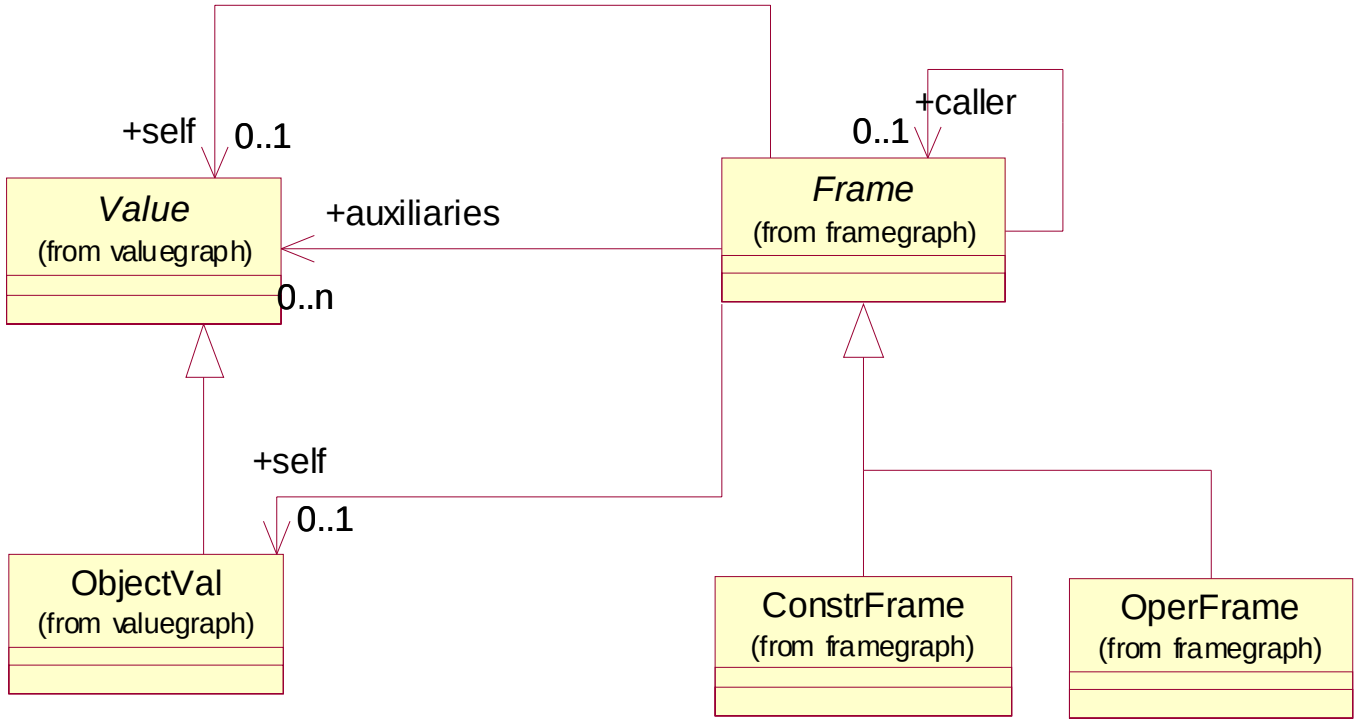
- Abstract syntax is captured in program graph
- Semantic domain is captured in execution graph
- Execution graph = value graph + frame graph
  - Value graph: objects and their links
  - Frame graph: info on running processes
- Graph transformation rules specify changes in execution graph
  - Based on occurences of nodes and edges in both program and execution graph

Example: abstract syntax

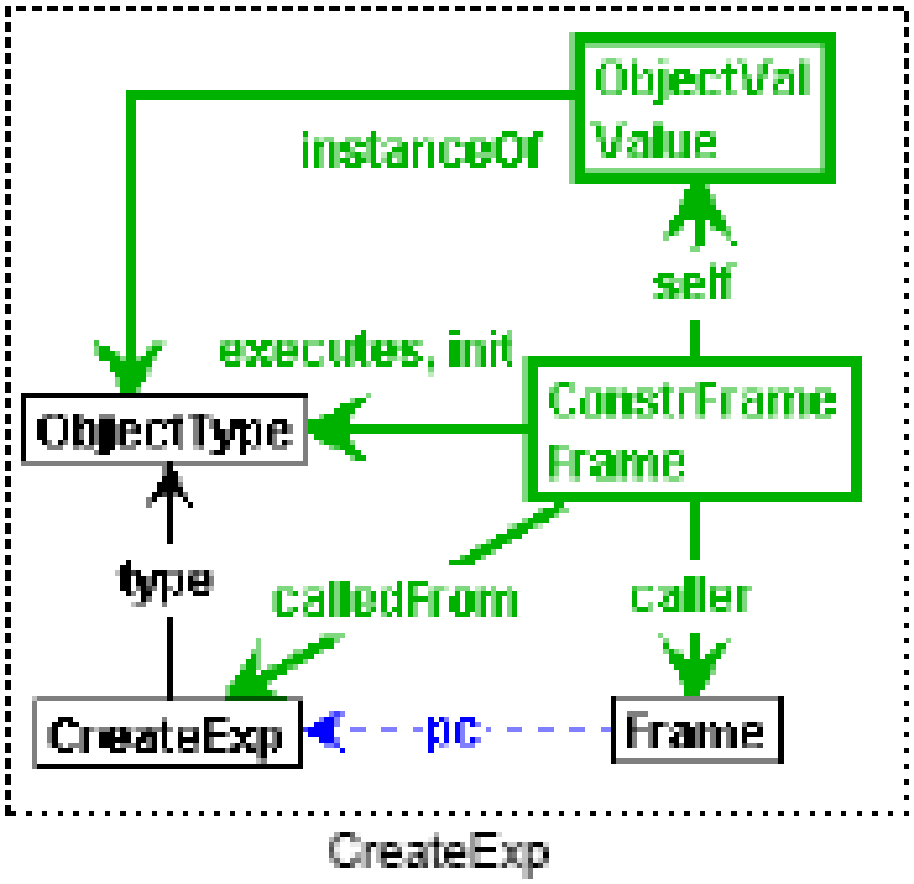


Example: semantic domain

University of Twente
Enschede - The Netherlands



Example rule



- Black: required
- Green: created
- Blue: deleted

More on TAAL

University of Twente
Enschede - The Netherlands

- Application of rules builds LTS that can be simulated and investigated
- Report at
  - http://www.cs.utwente.nl/~kastenbe/papers/taal.pdf
- Paper at FMOODS, June ’06, by Kastenberg, Kleppe & Rensink

Part 4

Defining Languages

Grasland, 2006 - 2008

University of Twente
Enschede - The Netherlands

- Graphs for Software Language Definitions
- Goal: to develop a formalism for defining software languages using graphs and graph transformations
  - And tool support

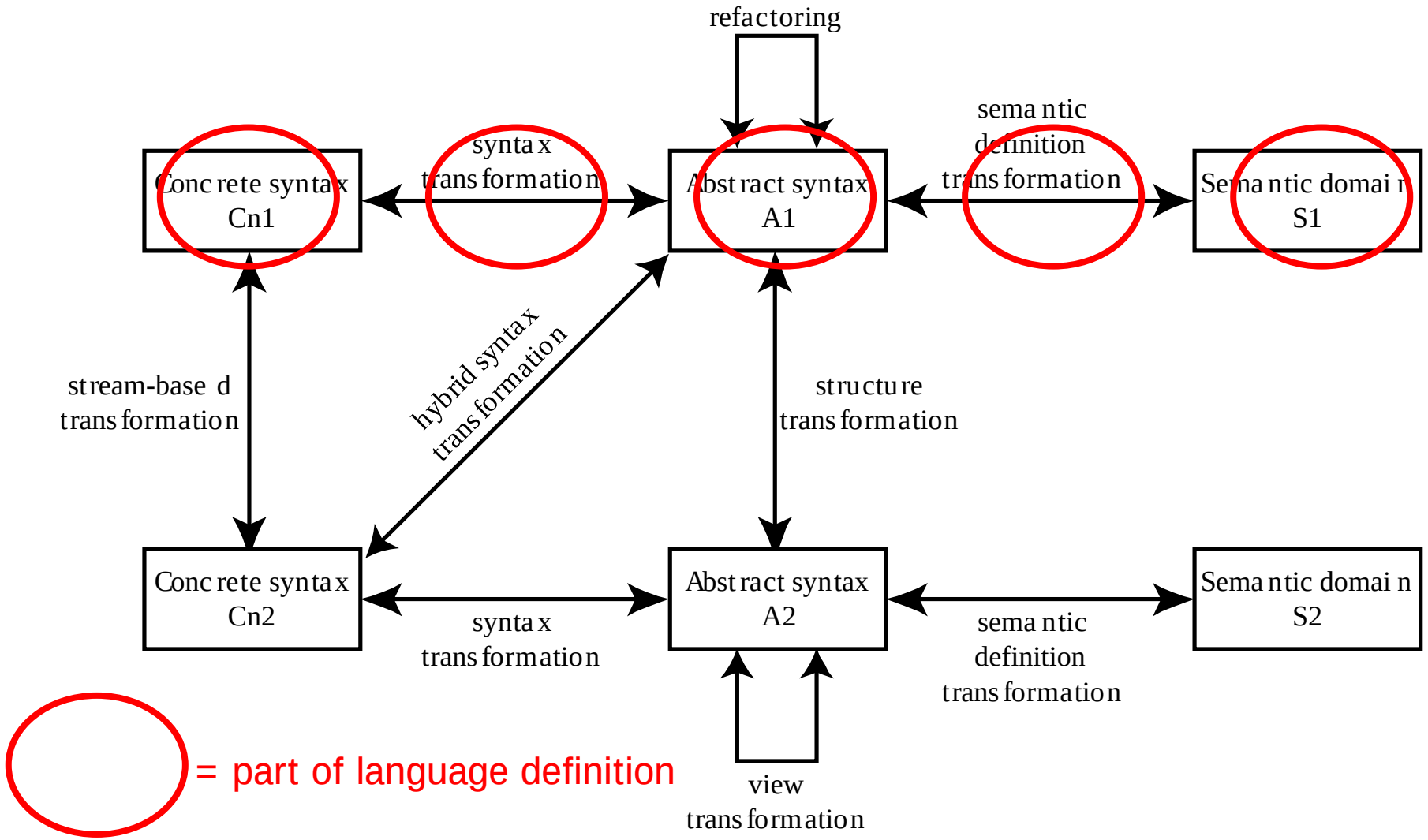
Language

- Definition *A language is a 5-tuple* $L = < AS, CSS, SD, M_S, SM_C >$ *consisting of*
  - *an abstract syntax (AS),*
  - *a set of concrete syntaxes (CSS),*
  - *a set of syntax mappings (SM_C),*
  - *a semantic domain (SD),*
  - *and a semantic mapping (M_S).*

  (from paper to appear in ECMDA ’06 by Kleppe)

Mappings and MDA

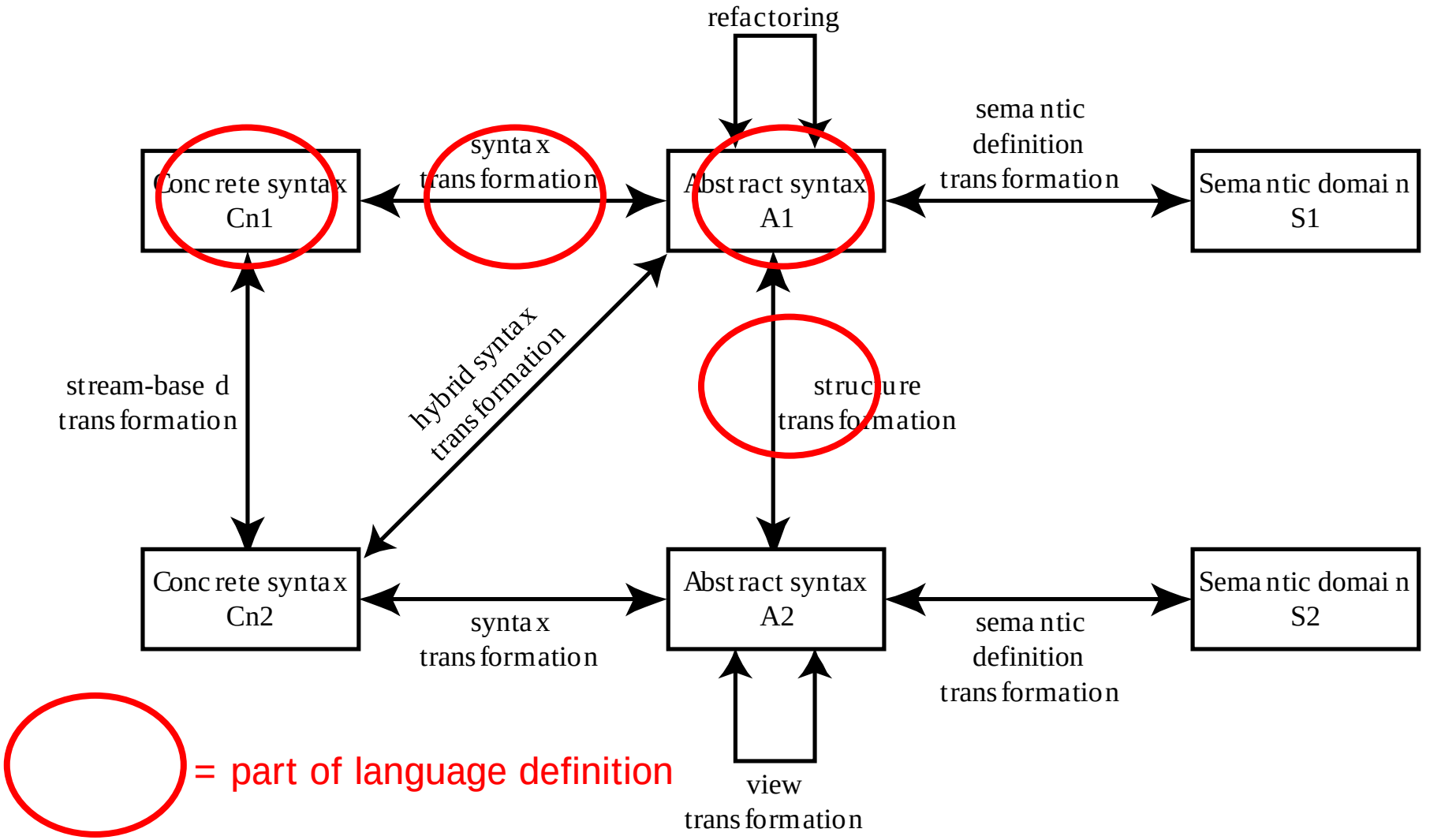
University of Twente
Enschede - The Netherlands



Two types of semantics

- Direct: semantic definition transformation
- Translational: structure transformation (L1 -> L2) + semantic definition transformation of L2

Language definition 2



Grasland formalism

- Metamodel to define CS, AS and SD (restricted form of UML)
  - Metamodel is type-graph of all possible models
- Graph transformations to define syntax and semantic mapping
  - or syntax and structure transformation

Part 5

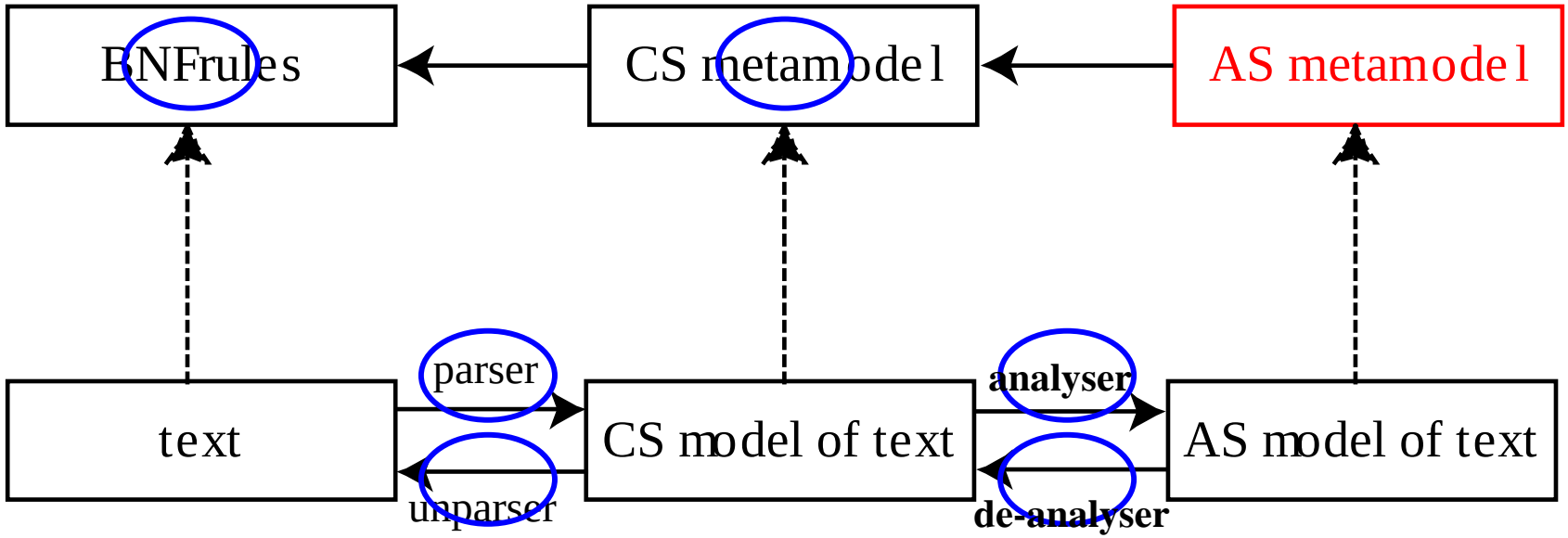
Concrete textual syntax

Generating textual CS

- Algorithm (and implementation) to transform an AS metamodel into a CS metamodel
  - CS metamodel defines a tree or forest
- Algorithm (and implementation) to transform a CS metamodel into parser generator input (JavaCC)
  - LL(n) parser

Generation of Language-IDE

- Only input is AS metamodel



is generated

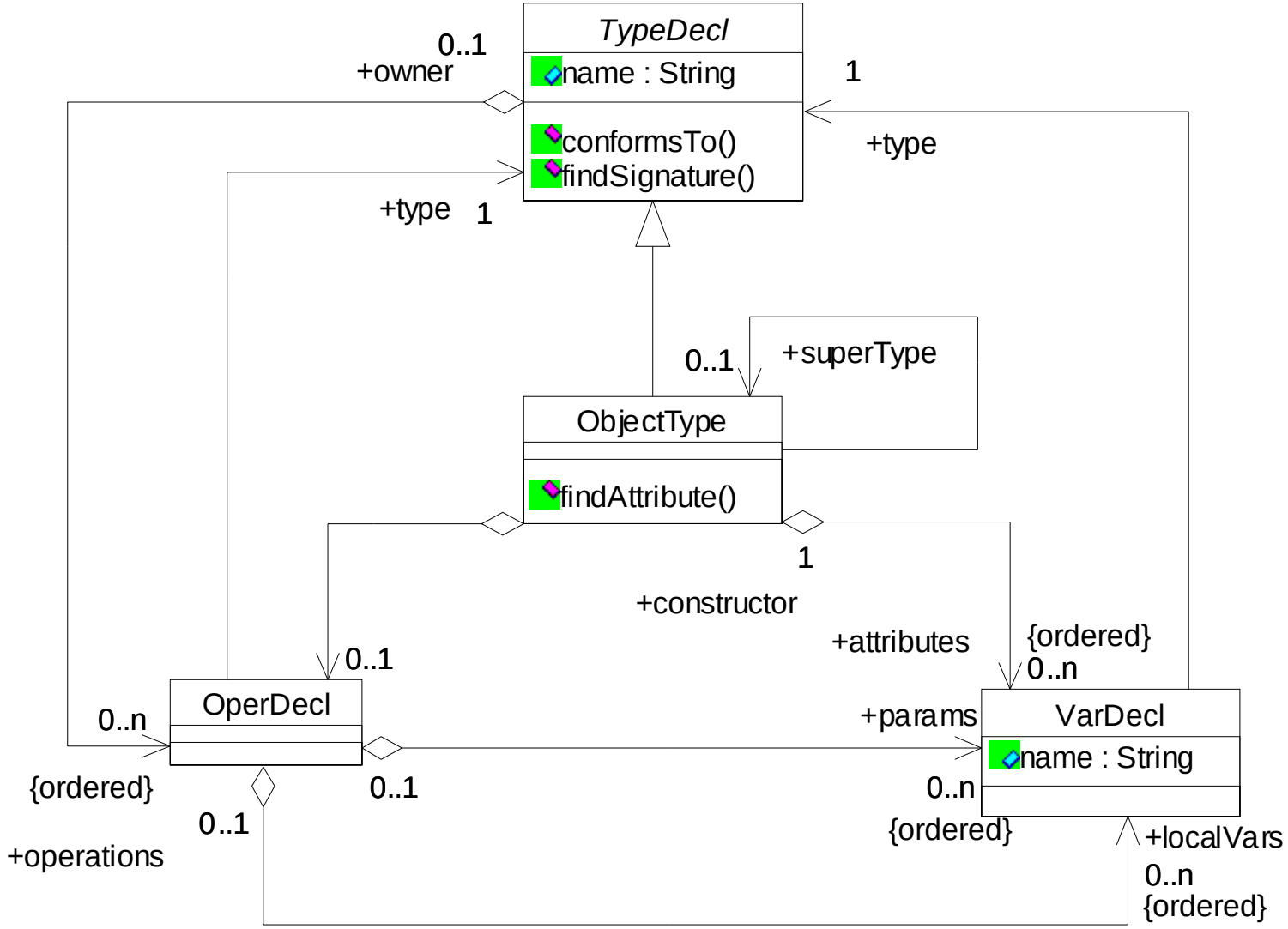
AS -> CS transformation

University of Twente
Enschede - The Netherlands

- Transformation rules:
  - AS metaclass -> CS metaclass
  - AS composite association -> CS composite association
  - AS non-composite association -> CS composite association to (new) Ref class
  - AS non-primitive attribute -> CS composite association
- Some classes and associations may be hidden
  - Rules needed for analyser to determine their value

Example AS metamodel

University of Twente
Enschede - The Netherlands



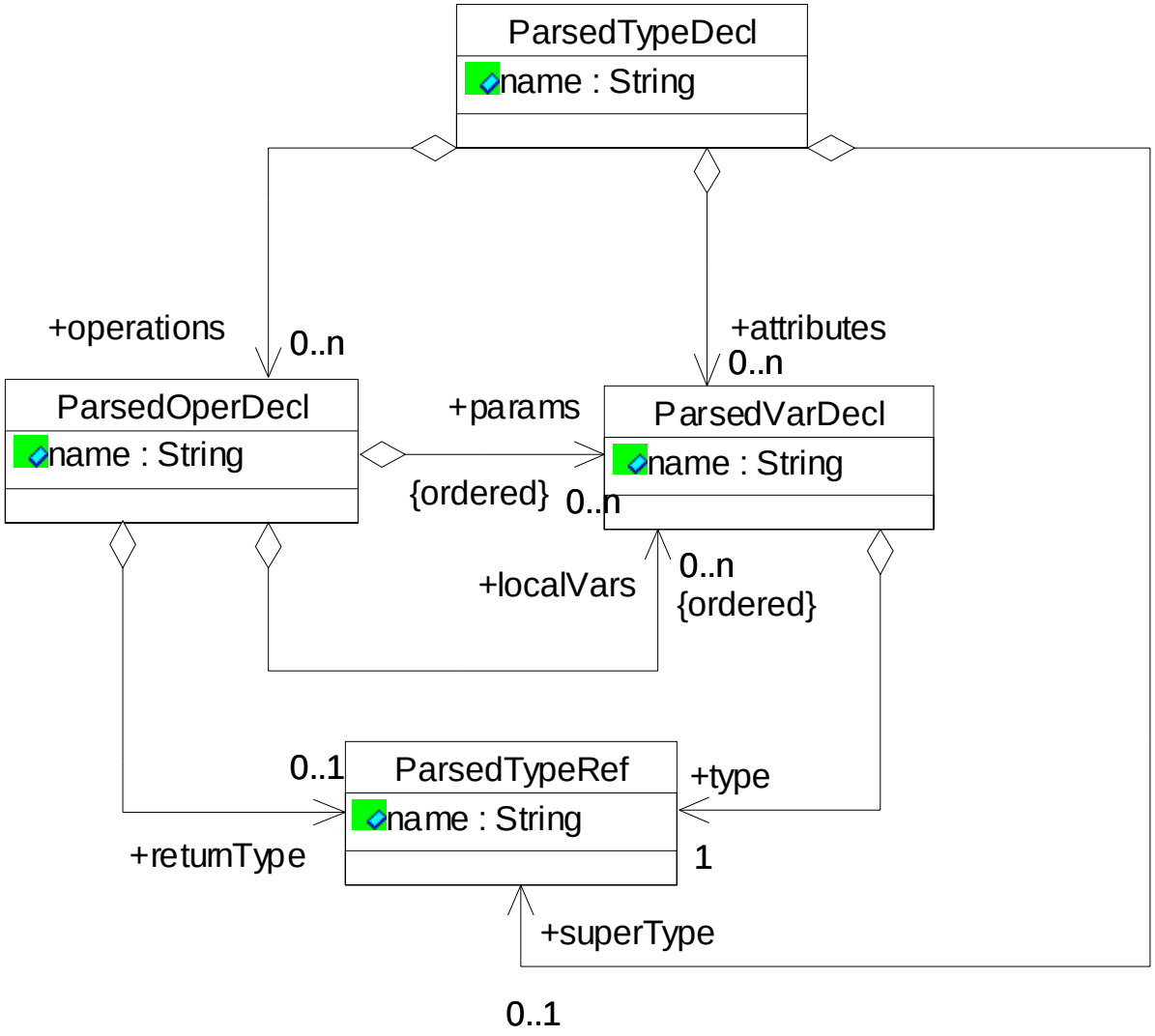
Restrictions on AS

- No association classes
- No interfaces
- No states

- Two-way associations allowed
- Non-composite associations allowed
- Non-primitive attribute types allowed
- Operations?

Example CS metamodel

University of Twente
Enschede - The Netherlands



Restrictions on CS for text

- Only composite associations
- All attributes have primitive type
- No interfaces, states, operations, and other fancy stuff

- Tooling uses property file for ordering, and for keywords in BNF rules
  - Class: begin_class, end_class
  - Association end: begin_list, end_list, separator

CS to BNF transformation

University of Twente
Enschede - The Netherlands

- Transformation rules:
  - CS metaclass -> BNF rule
  - CS inheritance -> BNF choice
  - CS (primitive) attribute -> BNF special token for strings, numbers etc.
  - Keywords -> BNF tokens
- Some keywords may be empty or mandatory
  - Lookaheads added
  - Different formats for lists (16?)

Example Properties file

University of Twente
Enschede - The Netherlands

- TYPEDECL_ORDER_1=name
- TYPEDECL_ORDER_2=superType
- TYPEDECL_ORDER_3=attributes
- TYPEDECL_ORDER_4=operations
- TYPEDECL_ATTRIBUTES_NAV_END=SEMICOLON
- TYPEDECL_ATTRIBUTES_NAV_SEPARATOR=SEMI COLON
- TYPEDECL_BEGIN=class
- TYPEDECL_END=endclass

Example BNF rules

University of Twente
Enschede - The Netherlands

ParsedTypeDecl_CS ::= <TYPEDECL_BEGIN> <IDENTIFIER>
[<TYPEDECL_SUPERTYPE_NAV_BEGIN> ParsedTypeRef_CS]
[ParsedVarDecl_CS (<SEMICOLON> ParsedVarDecl_CS)* <SEMICOLON>]
(ParsedOperDecl_CS)*
<TYPEDECL_END>

ParsedVarDecl_CS ::= <IDENTIFIER> <COLON> ParsedTypeRef_CS [
<ASSIGN> ParsedExpression_CS]

ParsedTypeRef_CS ::= <IDENTIFIER>

ParsedOperDecl_CS ::= <IDENTIFIER>
<OPERDECL_PARAMS_NAV_BEGIN> [ParsedVarDecl_CS (<COMMA>
ParsedVarDecl_CS)*] <OPERDECL_PARAMS_NAV_END>
[<COLON> ParsedTypeRef_CS]
[<OPERDECL_LOCALVARS_NAV_BEGIN> ParsedVarDecl_CS
(<SEMICOLON> ParsedVarDecl_CS)* <SEMICOLON>]

Part 6

Future work

2007 - ...

Visual concrete syntax

University of Twente
Enschede - The Netherlands

- Generation of input for visual editor generator
  - Parser often incorporated in editor
  - Separate CS metamodel

Semantic units

University of Twente
Enschede - The Netherlands

- Unit combining
  - Part of AS metamodel (1)
  - Part of SD model (2)
  - Mapping from 1 to 2
- Assumption: any language can be build from a combination of semantic units
- CS(es) added to this combination
- Goal: library of semantics units
- Problem: which combinations are allowed

Complete language-IDE

University of Twente
Enschede - The Netherlands

- Editor
- Parser/analyser
- Deparser/De-analyser
  - For each concrete syntax
- Simulator
  - Based on AS and mapping to SD
- Translator?
  - Based on two AS-es and their mappings???

Open question

- For textual languages:
  - Parse (derivation) tree represents application of production rules
  - CS metamodel is type graph of parse trees
- What is a metamodel w.r.t. other languages?