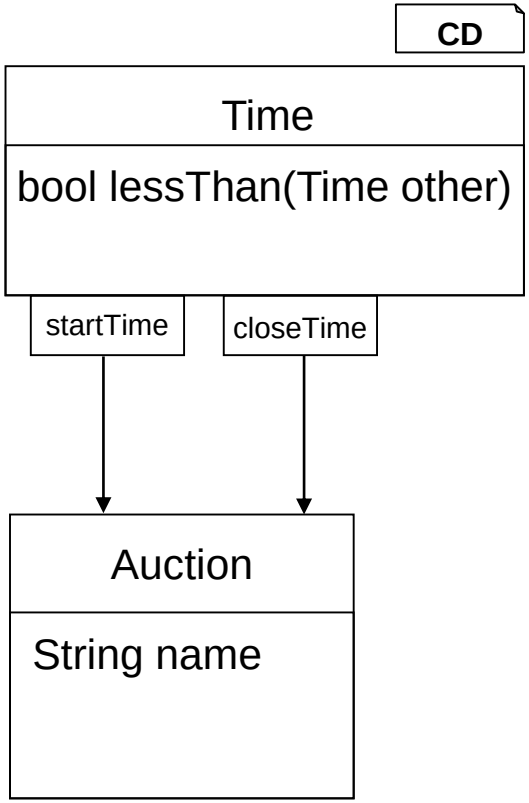
Improvements to OCL Implementation within the Monti-Core workbench

Ferdinand Mehlan
Seminar
am Lehrstuhl für Software Engineering
RWTH Aachen

Introduction to OCL

- Object Constraint Language (OCL)
- Part of UML
- Define invariants on runtime objects
- Uses information of classdiagrams



OCL

```
context Auction a inv:
  a.startTime.lessThan(a.closeTime)
```

- Formulated on class-level, semantics applied on object-level

Outline

1. Improvements: Grammar
2. Improvements: Types in OCL
3. Online Tool

Improvements: Grammar

- MontiCore language workbench
- For Domain Specific Languages (DSLs)
- Generate from Grammar (Parser, AST, Symboltable, Tools)

grammar

```
grammar OCL {
    CompilationUnit =
        ("package" package:(Name || ".")+ ";")?
        (ImportStatement)*
         OCLFile;
      // ...
}
```

OCL

```
package example.ocl;
import example.cd.Auction;

ocl myOCLDefs{
    // ...
}
```

- Minor Improvements:
  - Don‘t parse empty strings
  - Fix disparities to specification

Improvements: Grammar

- Expressions
  - Rules that can be nested
  - E.g. expr + expr , !expr
- MontiCore 3.x
  - No left recursion
  - Nodes dependent on each other
  - Changes break tools
  - Strict heirarchy
  - Specific Nodes for different expressions

grammar

```
grammar OCL {

  PlusMinusExpr implements Expression =
    left:PlusMinusExpr operator:["+" | "-"] right:PlusMinusExpr
    |
    ("(" paren: PlusMinusExpr ")" | nextHigherPrec: MultDivModExpr)
    ;

  MultDivModExpr implements Expression =
    left:MultDivModExpr operator:["*" | "/" | "%"] right:MultDivModExpr
    |
    ("(" paren: MultDivModExpr ")"  | nextHigherPrec:LogicalORExpr)
    ;
}
```

Improvements: Grammar

- MontiCore 4.x
  - Direct left recursion
  - Cleaner grammar
  - All expressions in one rule
  - Not extensible
  - No visitor pattern on different expressions for tools

grammar

```
grammar OCL {

  Expression =
    Expression operator:["*" | "/" | "%"] Expression
    |
    Expression operator:["+" | "-"] Expression;
}
```

Improvements: Grammar

- MontiCore 4.5.3+
  - Indirect left recursion
  - Easy to read
  - Priorities for expressions
  - Visitor pattern on each node for tools
  - Extensible

grammar

```
grammar OCL {
 interface Expression;

   PlusExpression implements Expression<20> =
     Expression "+" Expression;

   MinusExpression implements Expression<20> =
     Expression "-" Expression;

   MultExpression implements Expression<30> =
     Expression "*" Expression;

   DivExpression implements Expression<30> =
     Expression "/" Expression;
}
```

Improvements: Types in OCL

1. Improvements: Grammar
2. Improvements: Types in OCL
3. Online Tool

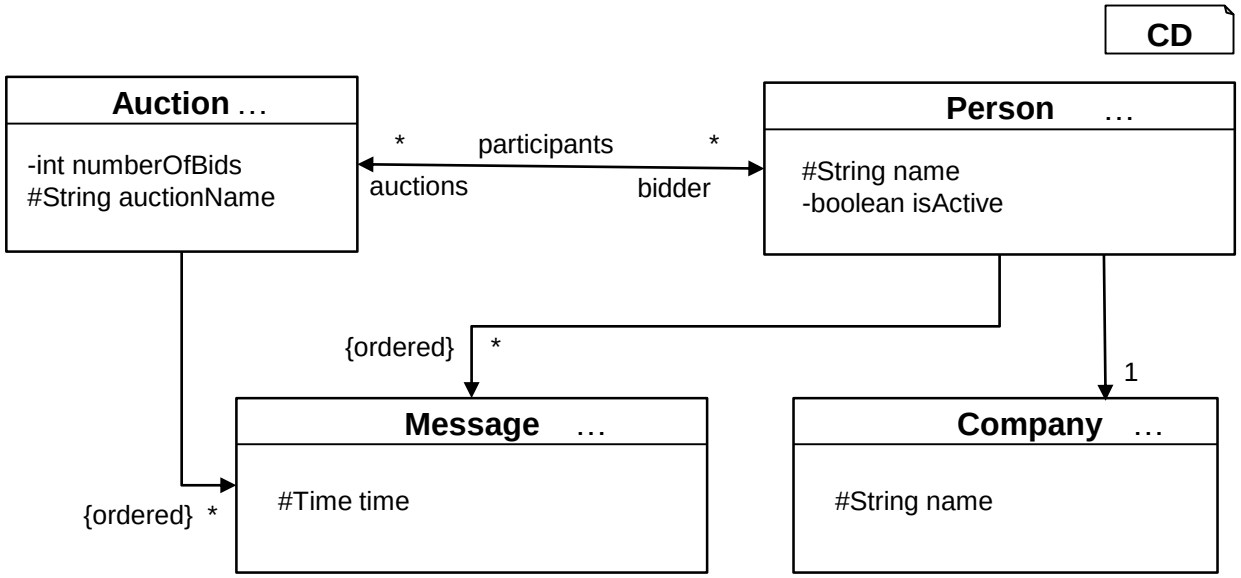
Types in OCL

- OCL a type-less language
- Uses CD information to navigate
- Concept: Automatically check correct navigation

OCL

```
context Auction a inv:
  let
    b = a.bidder;  // Set<Person>
    m = b.message;    // List<Message>
    t = {mess.time | mess in m}; // Collection<Time>
  in
    b == this.message
```

Types in OCL



OCL

```
context Auction a inv:
  let
    b = a.bidder;  // Set<Person>
    m = b.message;    // List<Message>
    t = {mess.time | mess in m}; // Collection<Time>
  in
    b == this.message
```

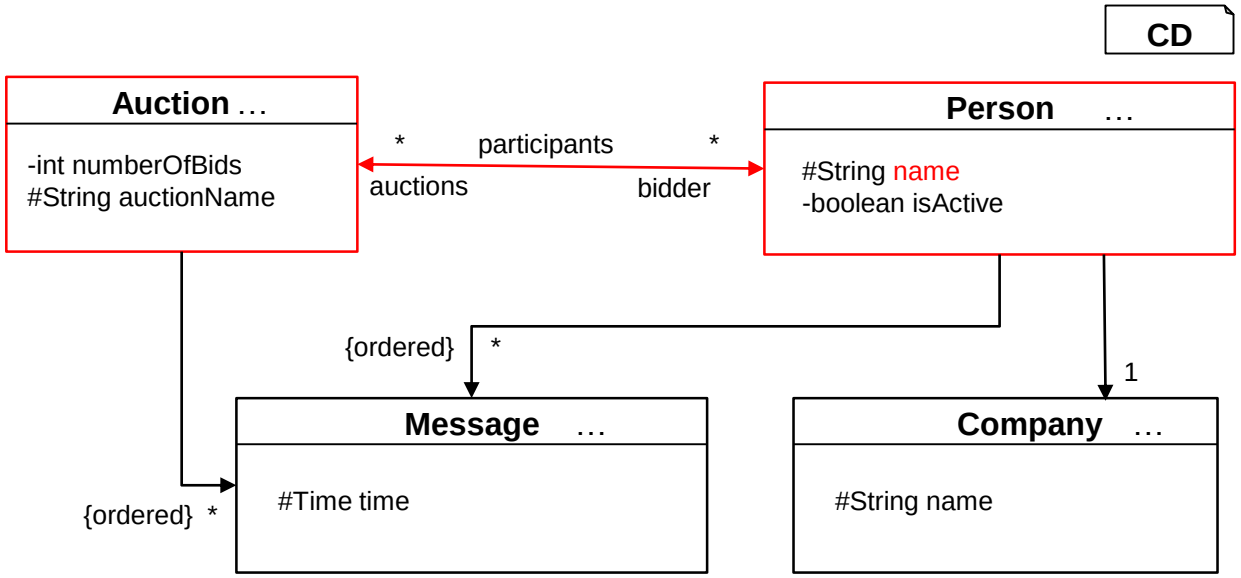
Types in OCL

- Navigation with the point operator using:
  - Association names, role names, class names, class fields
  - Specification mode (arrow direction is ignored)
  - Implicit variables (this, super)
  - Implicit flattening

OCL

```
context Auction a inv:
    a.participants.name != Set{}
context Auction a inv:
    a.bidder.name != Set{}
context Auction a inv:
    a.person.name != Set{}
```

Types in OCL



OCL

```
context Auction a inv:
    a.participants.name != Set{}
context Auction a inv:
    a.bidder.name != Set{}
context Auction a inv:
    a.person.name != Set{}
```

Types in OCL

- Build OCL symboltable and infer types
- Symbol OCLVariable for variable name and type
- Symbol CDType for type
- Automatically resolve types from CD symboltable
  - Check for illegal navigation

```
1  package ocl;
2  import cd.AuctionCD;
3  ocl myRule {
4
5    context Auction a inv:
7      let
8        m1 = a.message;
9        m2 = List{ m | m in a.bidder.message } ;
10     in
11       m1 == m2
13 }
```

OCLVariable

CDType

Types in OCL

- Inferring type of m2:
  - Infer type of m
  - Infer type of a.bidder.message

- M2 is of type List<Message>

```
1  package ocl;
2  import cd.AuctionCD;
3  ocl myRule {
4
5    context Auction a inv:
7      let
8        m1 = a.message;
9        m2 = List{ m | m in a.bidder.message } ;
10     in
11       m1 == m2
13 }
```

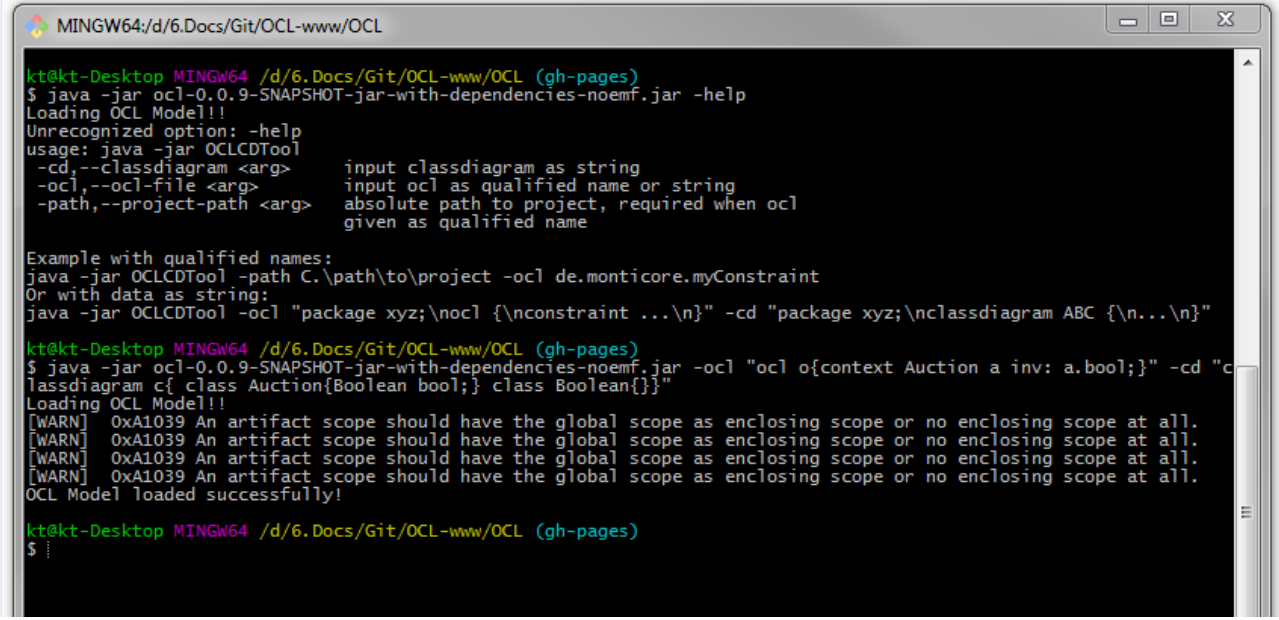
OCLVariable

CDType

Online Tool

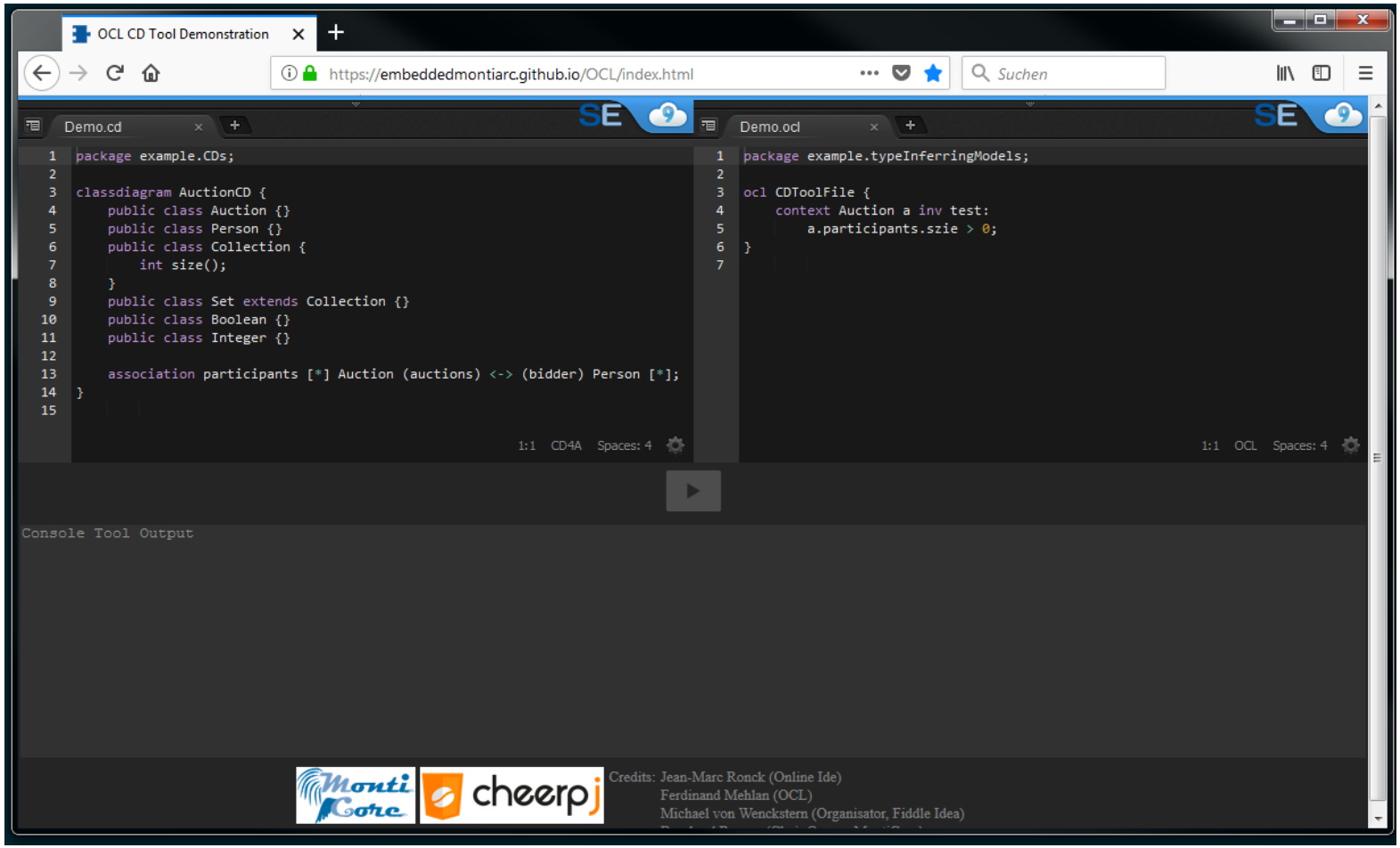
1. Improvements: Grammar
2. Improvements: Types in OCL
3. Online Tool

Online Tool



- CLI Tool to load and verify correct OCL
- Idea: Integrate tool into online interface
- CheerpJ by learningTech: Java -> JS transpiller
- Online IDE by Jean-Marc Ronck

Online Tool



https://embeddedmontiarc.github.io/OCL/index.html

Thank you for your attention!