Part II

ANATLYZER

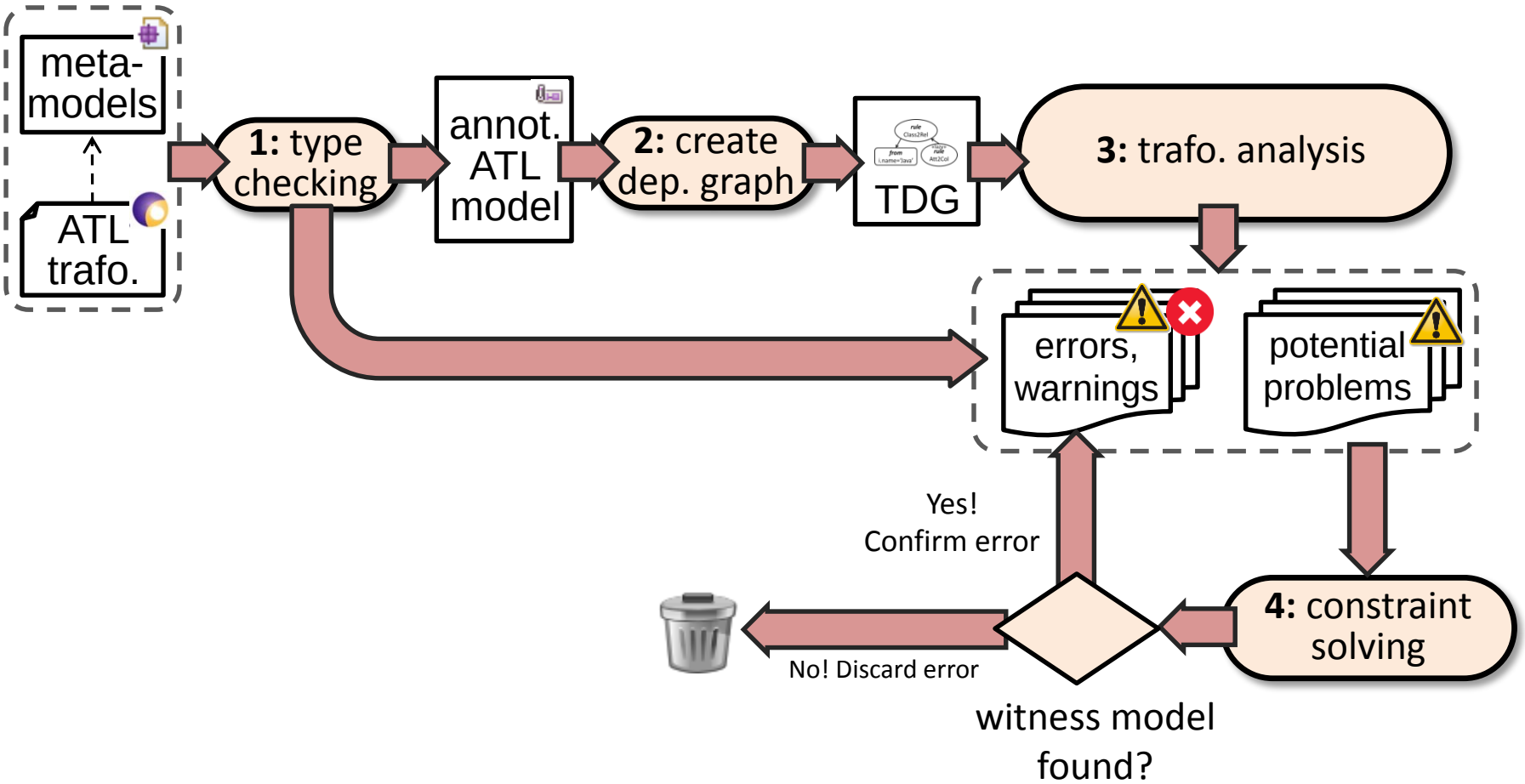
AnATLyzer

- A static analyser for ATL model transformations
- Static analysis
  - Detect problems before executing the transformation
  - Goal:
    - Be precise: few false positives
    - Be complete: few false negatives

Precision and completeness

- There is a trade-off between precision and completeness (recall).
  - Static analysis typically is over-restrictive
  - You can improve precision by sacrificing completeness and vice-versa
- Our approach is to detect “possible problems” and let the constraint solver find a witness model that confirms, or discard it otherwise.

AnATLyzer


meta-models
ATL trafo.
1: type checking
annot. ATL model
2: create dep. graph
TDG
3: trafo. analysis
errors, warnings
potential problems
Yes! Confirm error
No! Discard error
4: constraint solving
witness model found?

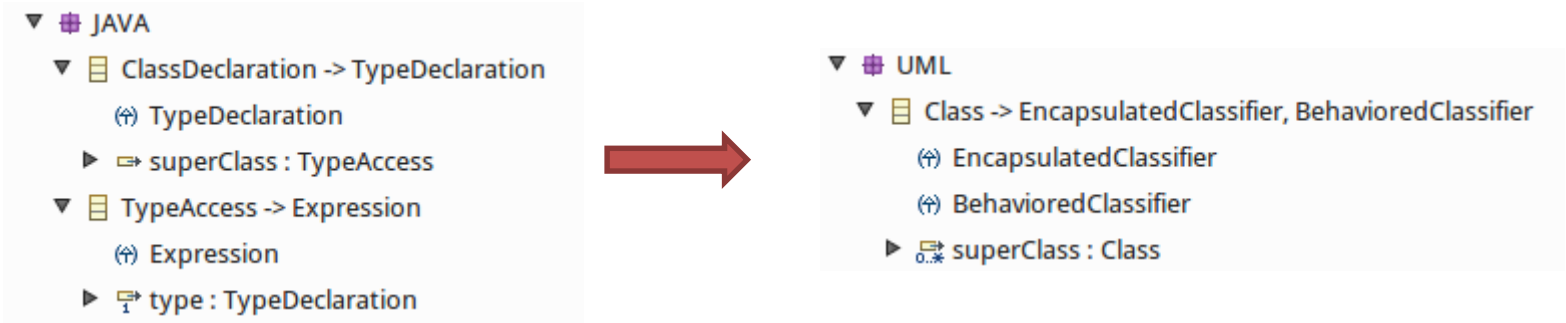
Motivation

- Writing a model(-to-model) transformation is a complex task
  1. You must handle every possible input configuration
  2. You must ensure the target model is syntactically correct (and satisfies the target constraints)
  3. There are many (sometimes implicit) relationships between the rules
  4. The mapping itself must be semantically correct

Motivation

- There are also accidental details due to the transformation language.
- In ATL:
  - It is dynamically typed
  - Hidden features not well documented
  - Design decisions may not be optimal

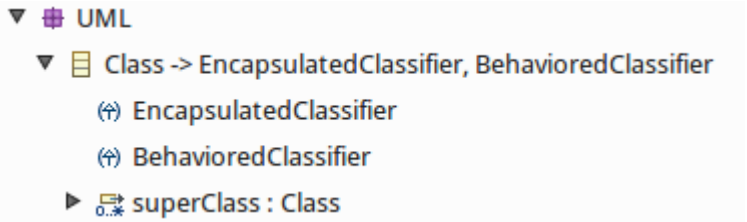
Is rule class2class correct?




```
-- Transforms Java classes (e.g., obtained with MoDISCO)
-- into UML classes and sets the inheritance links
module java2uml;
create CD : UML from CODE: JAVA;

helper context JAVA!ClassDeclaration
        def : getSuperClass : JAVA!ClassDeclaration = ... ;

rule class2class {
  from s1 : JAVA!ClassDeclaration
    to t1 : UML!Class (
        name <- s1.name,
        superClass <- s1.getSuperClass
    )
}
```

Is rule2class correct?

- Problem difficult to detect
  - Seasoned developers make mistakes like this
- A test may uncover a problem, but
  - The developer would need to know UML very well to identify the problem location



It is a derived feature:

```
self.generalization->
   collect(g | g.general)->
   select(c | c.oclIsKindOf(Class))
```

Questions

- What makes a high-quality transformation?
  - **Q1:** is the transformation correctly typed w.r.t. to the source meta-models?
  - **Q2:** do the generated models conform to the target meta-model?
  - **Q3:** do the transformation rules cover all cases?
  - **Q4:** is the transformation semantically correct?

Questions

- AnATLyzer helps you ask
  - Q1: typing
  - Q2: target conformance
  - Q3: rule coverage

- For Q4 (semantic correctness)
  - Testing is typically used
  - Code reviews, etc.

What AnATLyzer do for you?

- AnATLyzer detects more that 50 types of problems
- Additional features
  - IDE Integration (+ quick fixes, quick assist, explanations)
  - Source and target constraint handling
  - Visualizations
  - Support for UML profiles
- Utilities around AnATLyzer
  - Meta-model slicing
  - Constraint satisfaction for OCL
- Programmatic API

What AnATLyzer do for you?

- Useful for developing from scratch
  - Improved ATL editor
  - Quick fixes
- Useful during maintenance
  - A transformation has may implicit relationships among the rules
  - The static analysis may spot problems in a given change
  - Visualizations to understand rule relationships

Technical information

- Installation
  - Requirements:
    - Java 8
    - ATL 3.x
    - UML support (optional)
    - Visualization support (optional) – Eclipse Zest, Graphviz
    - Tested on Eclipse Luna, Mars and Neon
  - Web site and source code:
    - https://github.com/jesusc/anatlyzer
  - Update site:
    - http://sanchezcuadrado.es/projects/anatlyzer/sites/anatlyzer.updatesite/

Credits

- Built in the Miso team
  - Juan de Lara
  - Esther Guerra
  - Jesús Sánchez Cuadrado



http://miso.es

- Special thanks to the team behind USE/USE Validator
  - Martin Gogolla
  - Frank Hinkel ...

Research papers

- *Static analysis of model transformations*. Jesús Sánchez Cuadrado, Esther Guerra and Juan de Lara. IEEE Transactions on Software Engineering (2016).
- *Quick fixing ATL transformations with speculative analysis*. Jesús Sánchez Cuadrado, Esther Guerra and Juan de Lara. Software and Systems Modeling, 2016.
- *Translating Target to Source Constraints in Model-to-Model Transformations*. Jesús Sánchez Cuadrado, Esther Guerra, Juan de Lara, Robert Clarisó, Jordi Cabot. MoDELS'17, 2017.

* Papers available at: http://miso.es and http://sanchezcuadrado.es

AnATLyzer

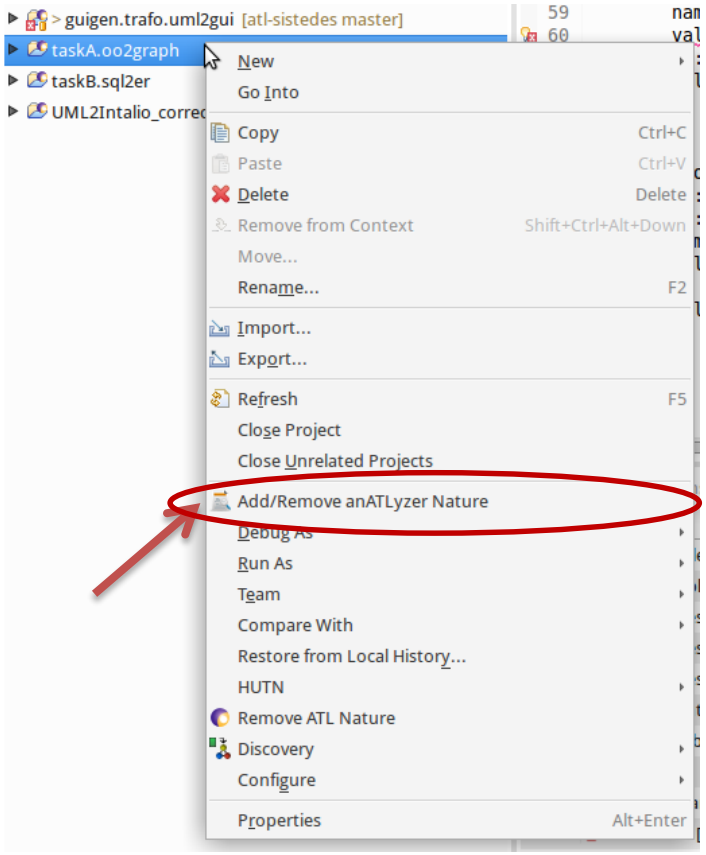
Using AnATLyzer

Installing the example projects

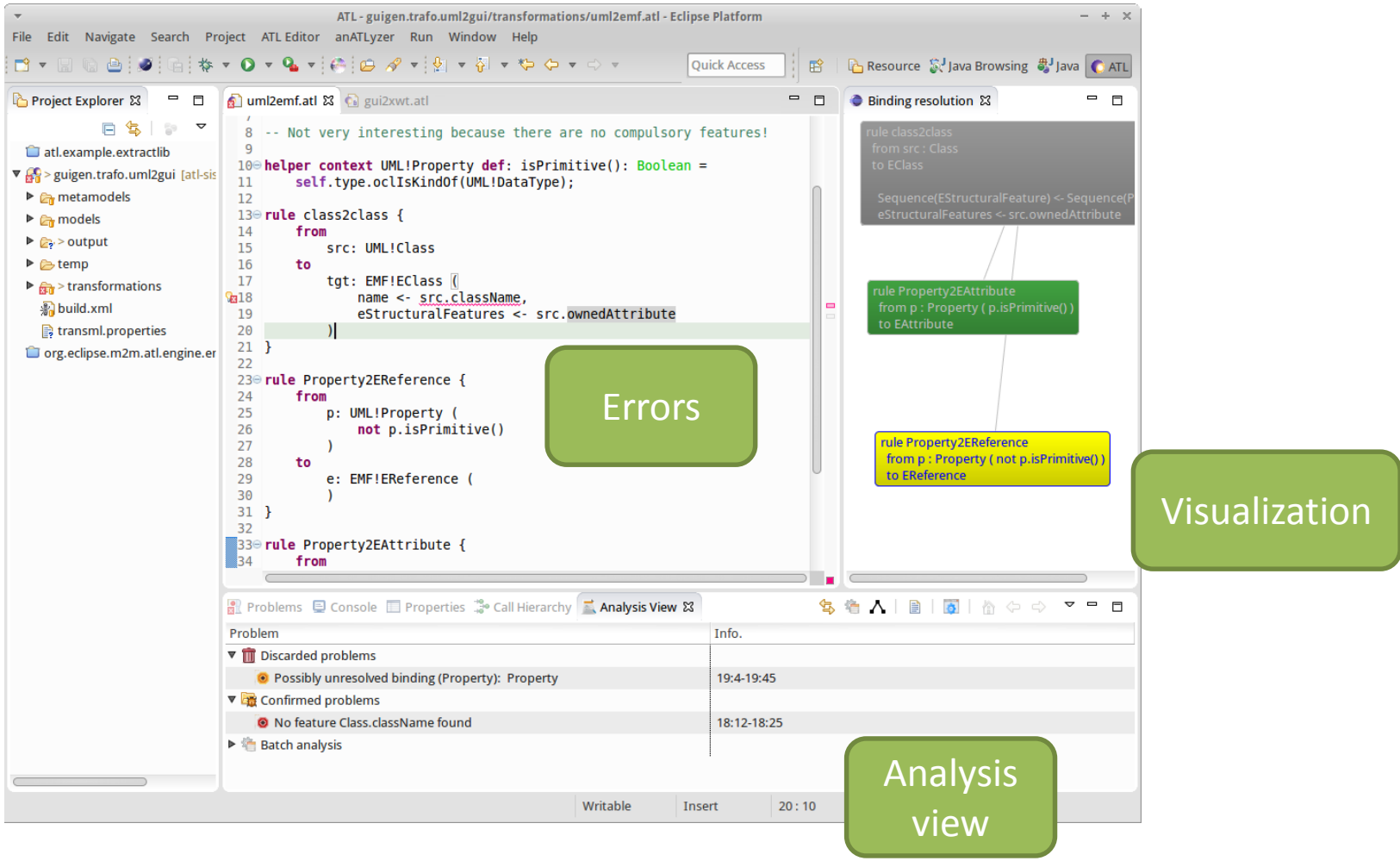
- Download from the tutorial Github page
  - http://github.com/jesusc/anatlyzer-models17
  - File → Import …
  - Existing projects into workspace
  - Select archive file (example-projects.zip)

Setting up the project

- Right-click on an ATL project
- Select “Add/Remove” AnATLyzer feature
- Transformations in the project will automatically be analysed (when opened in the editor)



User interface



The Analysis View

- Show list of detected problems
- Provide access to batch analysis
  - Regular problems scheduled for batch mode
  - Rule conflict
  - Target invariants analysis
  - Child stealing (experimental!)
  - Unconnected components (experimental!)
- Show the view
  - Window -> Show view -> Other ...
  - AnATLyzer -> Analysis View

The Analysis View

Warning

Error

Reload analysis

Problems | Console | Properties | Call Hierarchy | Analysis View | Search

Problem	Info.
Discarded problems	
Confirmed problems	
Possibly unresolved resolveTemp (Feature): Attribute, Reference	47:41-47:71
Possibly unresolved binding (Feature): Attribute, Reference	46:7-46:31
Possibly unresolved binding (Classifier): DataType, Class	31:3-31:68
No feature IntegerValidator.validators found	60:3-60:23
Binding may be resolved by rule with invalid target type (src : Feature). 46:7-46:31	46:7-46:31
Batch analysis	
Rule conflict analysis	Some conflicts: 1/3
Reference: [MultiRef2Widget, MonoRef2Widget] : Confirmed (by solver)	
DataType: [int2validator, DataType2StringValidator] : Discarded (by solver)	
Attribute: [TextProperty2Widget, IntProperty2Widget] : Discarded (by solver)	
Child stealing analysis	Some conflicts: 6/11
widgets (46:7-46:31) and widgets (46:7-46:31) and rule TextProperty2Widget : Confirmed (by solver)	
widgets (46:7-46:31) and widgets (46:7-46:31) and rule MultiRef2Widget : Confirmed (by solver)	
widgets (46:7-46:31) and widgets (46:7-46:31) and rule MonoRef2Widget : Confirmed (by solver)	

Double-click on “Rule conflict analysis” or “Child stealing” to execute

The Analysis View

- Confirmed
  - It is a true error. Should be fixed somehow.
  - Try some quick fix! CTRL + 1
- Discarded
  - We used model finding to ensure it is not an error
  - Can be ignored
- Unknown
  - It is a smell but we cannot check if it is an error.
- Running
  - Errors which are currently being processed
  - Most of the time the user does not see this.
- Time out
  - If it takes to long to confirm the problem

Keyboard shortcuts

- CTRL + 1
  - Over an error, show quick fix
  - Over a normal statement, show quick assists
- Be ready to use CTRL-Z to undo...
- CTRL + S to save and re-analyse
  - The analysis is mostly incremental

Keyboard shortcuts (Inherited from ATL Editor)

- Auto-complete / Template proposals
  - CTRL+SPACE
  - Not completely precise
- Go to definition (e.g., helper, definition)
  - CTRL + Click
  - F3 with the keyboard
- Comment / Uncomment
  - CTRL+SHIFT+C

Quick fixes

Boosting transformation development

Quick fixes

cd2gui.atl

```
15 create OUT: GUI from IN: CD;
16
17
18 helper context CD!Attribute def: isText() : Boolean = self.type.oclIsKindOf(CD!DataType) and self.type.name = 'String';
19 helper context CD!Attribute def: isInt()  : Boolean = self.type.oclIsKindOf(CD!DataType) and self.type.name = 'Integer';
20 helper context String def: toLabelName() : String = self.toLower();
21
22 helper context CD!Class def: allFeatures : Sequence(CD!Feature) =
23     self.superclasses->collect(c | c.allFeatures)->flatten()->union(self.features);
24
25
26 rule model2gui {
27     from m : CD!Model
28     to   w : GUI!Window (
29         name <- m.name,
30         name <- m.name,
31         widgets <- m.classifiers->reject(c | c.oclIsKindOf(CD!DataType) ),
32         layout <- hflow
33     ), g : GUI!GUI (
34         windows <- w,
35         validators <- CD!Attribute.allInstan
36     ), hflow : GUI!HFlow (
37
38     )
39
40 }
41
42 rule class2frame {
43    from c : CD!Class ( not c.isAbstract )
44      to w : GUI!Frame (
45         title <- c.name,
46         widgets <- c.allFeatures,
47         widgets <- c.allFeatures->collect(f
48         -- Show the idiomatic way...
```

Possibly unresolved binding (Classifier): DataType, Class

- Add rule...
- Add rule: Classifier
- Add filter expression to binding
- Remove binding
- Generate most general pre-condition
- Generate precondition
- Add ignore annotation: unresolved-binding
- Visualize problem

Access to detailed Information about the problem

Quick fixes

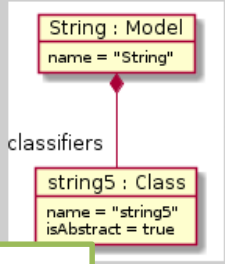
Visualization quick access

Problem information

Possibly unresolved binding (Classifier): DataType, Class

Problem explanation

There are some configurations of objects (in the right-hand side of the binding) which are not handled by any of the resolving rules



Witness

Speculative quick fixes

Possible fixes

Quick fixes

#	Quickfix
3	Add filter expression to binding
3	Remove binding
3	Generate most general pre-condition
3	Generate precondition
4	Add ignore annotation: unresolved-binding
4	Visualize problem
9	Add rule: Classifier
U	Add rule...

Add filter expression to binding

The fix will solve the problem, produces 1 new problems, and it does not fix any additional problem.

Original problem fixed

Fixed problems (1) | New problems (1) | Remaining problems (3)

Possibly unresolved binding (Classifier): DataType, Class

```
m.classifiers->reject(c | c.oclIsKindOf(CD!DataType))->select(_v | if ( _v.oclIsKindOf(CD!DataTyp
  _v.name = 'String' or _v.name = 'Integer'
else
  if ( _v.oclIsKindOf(CD!Class) ) then
  not  v.isAbstract
```

Cancel | OK

Types of quick fixes

- Modify the transformation
  - E.g., add a rule filter
- Modify the meta-model
  - E.g., Add a new class
- Generate pre-condition
  - E.g., To document that certain configurations are not allowed
- Ignore the problem

Developing from scratch

- Make errors on purpose
- Save the file to re-execute the analysis
  - Should be fast
- CTRL + 1
- Know the quick fix to apply on advance

Developing from scratch

- Typical scenarios
  - Create the helper signature automatically
  - Create rules
  - Create new meta-model classes/features
  - As a an auto-completion mechanism
    - E.g., the variable name in a resolveTemp

Static analysis with AnATLyzer

Types of problems

Types of problems

- AnATLyzer detects more than 50 types of problems
- Classification:
  - Typing and navigation
    - Typing w.r.t. meta-models and use of OCL
  - Transformation integrity
    - Checks related to the transformation structure
  - Target meta-model conformance
    - Does the output model conforms to the target meta-model?
  - Transformation rules
    - Issues related to (matched) rule usage

Typing and navigation

- OCL expressions should be well typed w.r.t. the source meta-model
- AnATLyzer detects problems like:
  - Invalid references to classes and features
  - Invalid iteration expressions
  - Invalid variable declarations / return types
  - “Null pointer exceptions”
  - Invalid downcasting (“Feature found in subtype”)

uml2gui_errors_01.atl

Typing and navigation

- Invalid class name and invalid feature

```
helper context CD!Model
          def: classifiers : Sequence(CD!Clasifier) =
  self.ownedTypes->select(c | c.oclIsKindOf(CD!Classifier));
```

  - `Clasifier` : is a typo
  - `ownedTypes` : does not exist, it is ownedType

uml2gui_errors_02.atl

Typing and navigation

- Incoherent return type

```
helper context CD!Class
          def: superclasses : Sequence(CD!Generalization) =
   self.generalization->
         collect(g | g.general)->
         select(c | c.oclIsKindOf(CD!Class));
```

uml2gui_errors_03.atl

Typing and navigation

- Access to undefined value
  - This is the “Null pointer exception” of OCL

```
helper context CD!Property def: isText(): Boolean =
   self.type.oclIsKindOf(CD!DataType) and
   self.type.name = 'String';
```

  - Possible quick fixes:
    - Surround with “if”
    - Generate pre-condition

The type checker uses the condition to rule out the problem

Constraint solving is then used to discard the problem

Typing and navigation

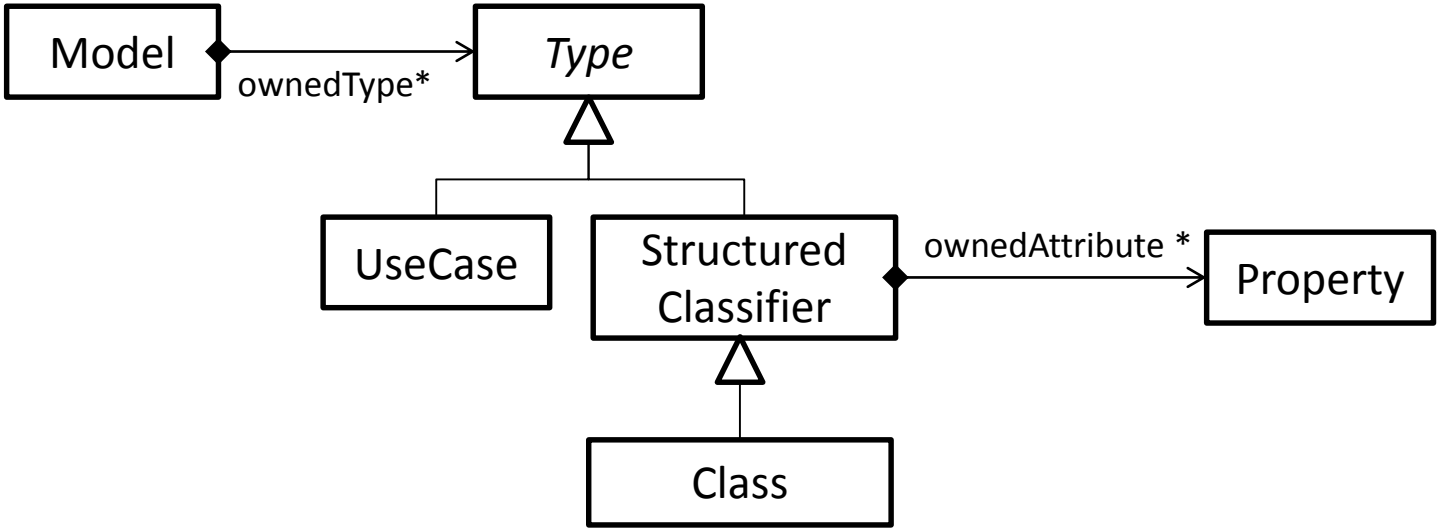
- Implicit downcasting
  - In ATL there is no **oclAsType**
  - An attribute/operation must defined for all possibles dynamic types of an expression
    - Duck typing

uml2gui_errors_04.atl

Typing and navigation

- Implicit downcasting

```
helper context CD!Model def: notEmptyClasses: Sequence(CD!Class) =
   self.ownedType->select(t | t.ownedAttribute->notEmpty());
```



uml2gui_errors_04.atl

Typing and navigation

- Implicit downcasting

```
helper context CD!Model def: notEmptyClasses: Sequence(CD!Class) =
    self.ownedType->select(t | t.ownedAttribute->notEmpty());
```

- Possible fixes

```
helper context CD!Model def: notEmptyClasses: Sequence(CD!Class) =
    self.ownedType->select(t | t.oclIsKindOf(CD!Class))->
                    select(t | t.ownedAttribute->notEmpty());
```

AnATLyzer uses this to infer the proper typing

Special operations

- `oclAsType`
  - ATL does not have a downcasting operation
  - If you implement this dummy operation:

```
helper context OclAny def: oclAsType(t : OclType) : OclAny = self;
```

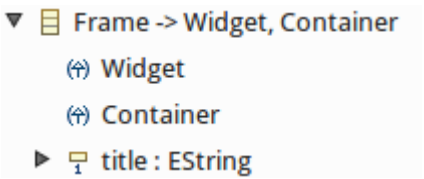
  - AnATLyzer recognizes it to avoid so many nested ifs.

- `fail_(str : message)`
  - OclUndefined.fail_("Pattern match error")
  - To indicate an impossible path in your code

uml2gui_errors_05.atl

Target conformance problems

- No binding for compulsory feature



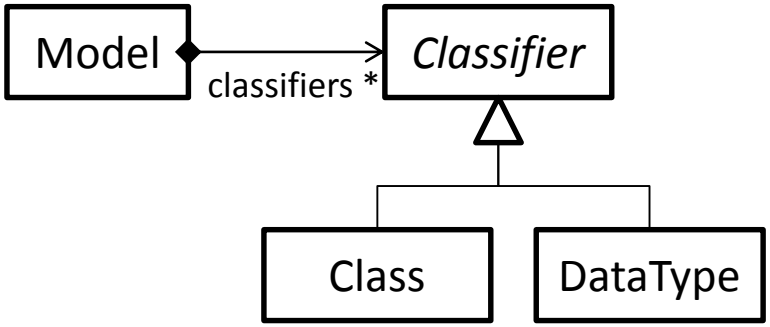

```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
      widgets <- c.ownedAttribute,
      ...
  )
}
```

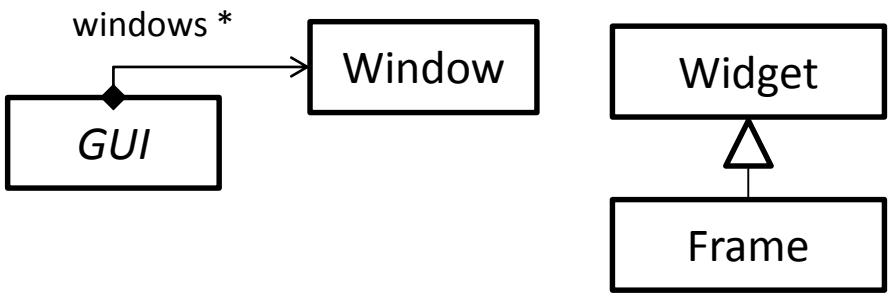
  - Feature `title` is compulsory, but the rule is not setting it.
  - It will cause problems in other transformations relying on the existence of a value for `title`.

uml2gui_errors_06.atl

Target conformance problems

- Binding resolved by rule with invalid target
  - A binding gets assigned a target object whose type is incompatible with the feature type






```
rule model2gui {
  from m : CD!Model
  to g : GUI!GUI (
      windows <- m.nonEmptyClasses()
    )
}
```

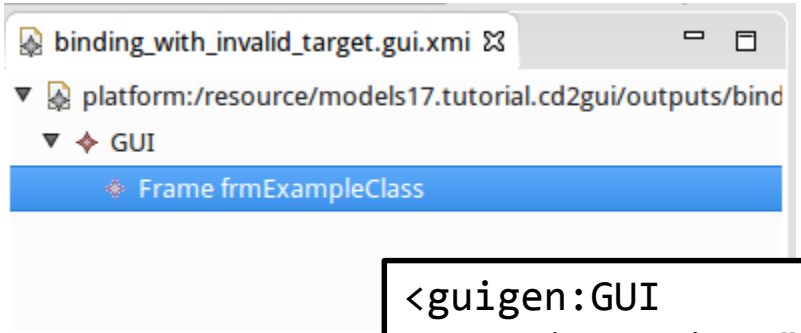
```
rule class2frame {
 from c : CD!Class ( not c.isAbstract )
 to w : GUI!Frame  ( … )
}
```

Frame is not compatible with Window!

uml2gui_errors_06.atl

Target conformance problems

- Binding resolved by rule with invalid target




```
<guigen:GUI
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:guigen="http://guigen/gui">

  <!-- There is a Frame object where
       only Window is permitted -->
  <windows xsi:type="guigen:Frame"
      name="frmExampleClass"/>

</guigen:GUI>
```

Transformation integrity

- ATL code which is syntactically correct but leads to unexpected behaviour.
- Example
  - Are filters in lazy rules allowed?

```
lazy rule property2text {
  from p : CD!Property ( p.isText() )
  to   t : GUI!Text
}
```

  - The lazy rule will be executed regardless of the filter.
  - Filters in lazy rules only work with rule inheritance

uml2gui_errors_07.atl

Transformation rules problems

- Unresolved binding
  - What happens when there is no rule to resolve an element appearing in the right part of a binding?
  - Example:

```
rule class2frame {
 from c: CD!Class ( not c.isAbstract )
 to   f: GUI!Frame (
     widgets <- c.ownedAttribute, ...
```

  - Rules `property2text` and `property2date` only handles a subset of the possible Property objects

Cannot set feature widgets to value [org.eclipse.emf.ecore.impl.DynamicEObjectImpl@4a12c7a0 (eClass: org.eclipse.emf.ecore.impl.EClassImpl@54087d0d (name: Frame) (instanceClassName: null) (abstract: false, interface: false)), org.eclipse.emf.ecore.impl.DynamicEObjectImpl@632e536 (eClass: org.eclipse.emf.ecore.impl.EClassImpl@789537ef (name: Class) (instanceClassName: null) (abstract: false, interface: false)), org.eclipse.emf.ecore.impl.DynamicEObjectImpl@f99ae63 (eClass: org.eclipse.emf.ecore.impl.EClassImpl@54087d0d (name: Frame) (instanceClassName: null) (abstract: false, interface: false)), org.eclipse.emf.ecore.impl.DynamicEObjectImpl@6704dd1e (eClass: org.eclipse.emf.ecore.impl.EClassImpl@54087d0d (name: Frame) (instanceClassName: null) (abstract: false, interface: false))], inter-model references are forbidden. Configure launching options to allow them.

Transformation rules

- Unresolved binding
  - Should be treated appropriately
  - It is a smell of incompleteness in the transformation
    - Not all cases are covered
  - If the cases don’t need to be considered:
    - Filter the right-hand side of the binding
    - Write a pre-condition
    - Ignore (but documenting)

uml2gui_errors_08.atl

Transformation rules

- resolveTemp with invalid output pattern

```
rule class2frame {
  from c: CD!Class
    to f: GUI!Frame (
        ...
    ), grid: GUI!GridLayout (
        numColumns <- 2,
        info <- c.ownedAttribute->
            collect(a | thisModule.resolveTemp(a, 'grid1')),
        info <- c.ownedAttribute->
            collect(a | thisModule.resolveTemp(a, 'g2'))
    )
}
```

It should be 'g1'

Pre-conditions

- Useful to document the conditions under which the transformation actually works
- Used by anATLyzer to filter out problems
  - Need to be written formally in OCL
- Written as:
  - Module annotations (`--@pre`)
  - Helpers annotated with `-- @precondition`

Pre-conditions

- Module annotations

```
-- @pre CD!Property.allInstances()->forAll(v |
--          not v.type.oclIsUndefined())

module “uml2gui”;
create OUT : GUI from IN : CD;
```

- Helpers

```
-- @precondition
helper def: supportedDataTypes() : Boolean =
  CD!Property.allInstances()->forAll(p |
       p.isText() or p.isInt() );
```

Annotations

- Ignore annotations
  - They are used to remove problems of a certain type in a rule or helper
  - Easy access via a quick fix
  - Examples:
    - **-- @ignore unresolved-binding**
    - **-- @ignore no-binding-compulsory-feature**

uml2gui_errors_09.atl

Annotations

- Force return type
  - To prefer declared type over inferred
  - Type inference is typically precise, but false positives may arise
  - **-- @force-declared-return-type**

```
-- @force-declared-return-type
helper context UML!Element def: getContainingModel() : UML!Model =
        if self.refImmediateComposite().oclIsTypeOf(UML!Model) then
                self.refImmediateComposite()
        else
                self.refImmediateComposite().getContainingModel()
        endif;
```

Libraries

- Library support via @lib annotation

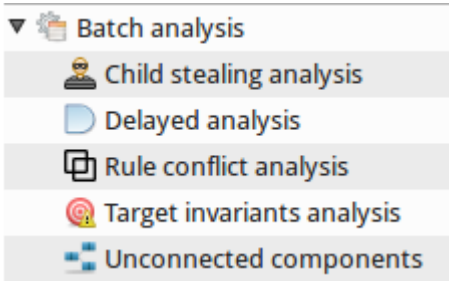
```
-- @nsURI CD=http://www.eclipse.org/uml2/5.0.0/UML
-- @path GUI=/models17.tutorial.cd2gui/metamodels/gui.ecore
--
-- @lib GUILIB=/models17.tutorial.cd2gui/transformations/guilib.atl

module “uml2gui”;
create OUT : GUI from IN : CD;

uses GUILIB
```

All helpers are inlined in the current module

Batch analysis

Batch analysis

- Analysis that may take too long to do them while editing
  - Delayed analysis
  - Rule conflict analysis
  - Child stealing analysis
  - Target invariant analysis
  - Unconnected component analysis

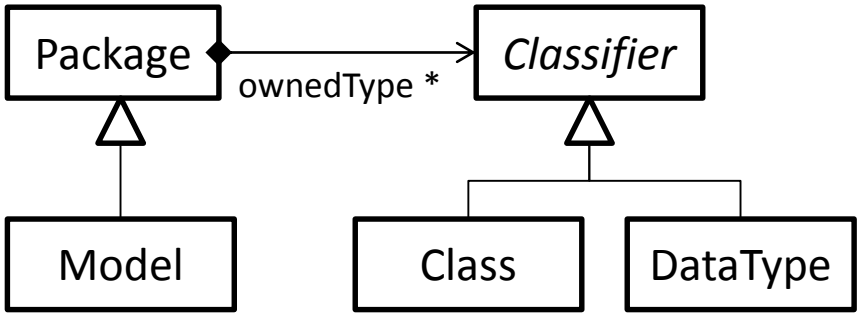


Delayed analysis

- Regular checks removed from the live analysis and scheduled to be executed in batch mode
  - See configuration options
    - Global AnATLyzer configuration
      - Window -> Preferences -> AnATLyzer
      - Default batch configuration
    - ATL file configuration
      - Right-click on the file -> AnATLyzer -> Configure AnATLyzer
      - Fine grained control over the checked problems

Rule conflicts

- Two matched rules should not match the same source element




```
rule model2window {
  from m1: CD!Model
  to   w: GUI!Window
}

rule package2window {
  from p: CD!Package
  to   w: GUI!Window
}
```

Solution #1. Make model2window inherit package2window
Solution #2. Add filter p.oclIsType(UML!Package)

uml2gui_rule_conflict.atl

Rule conflicts

```
rule property2text {
  from p : CD!Property ( p.isText() )
  to   t : GUI!Text
}

rule property2largeText {
  from p : CD!Property (
      p.type.name = ‘LString' or p.type.name = 'Text'
  )
  to   t : GUI!TextArea
}
```

Is p.isText() in conflict with the other filter?

uml2gui_rule_conflict.atl

Rule conflicts

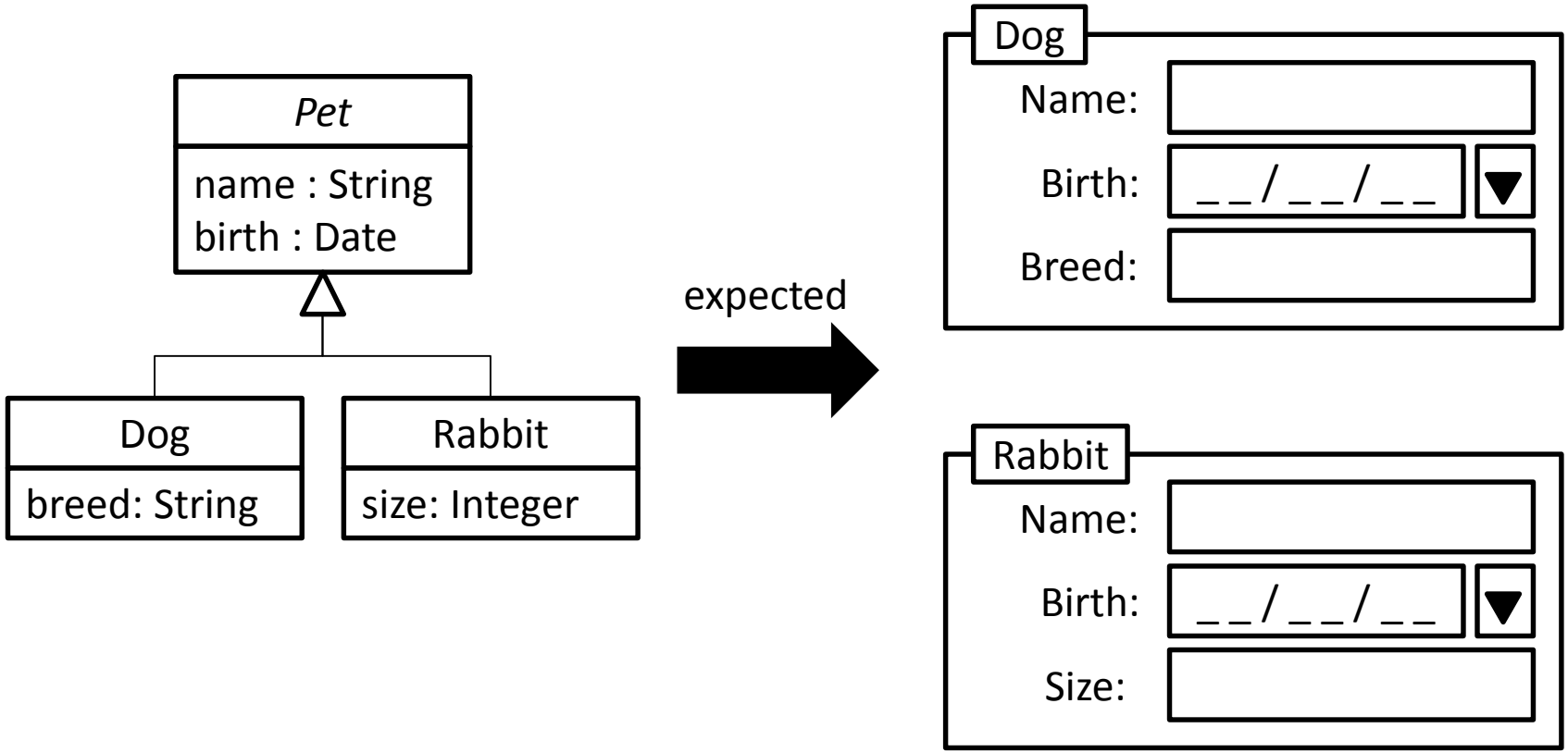
- Double click on “Rule conflict analysis”
  - Green means “that’s ok”
  - Red means “there’s a rule conflict!”

Rule conflict analysis	Some conflicts: 1/3
Property: [property2date, property2text] : Discarded (by solver)	
Property: [property2largeText, property2date] : Discarded (by solver)	
Property: [property2largeText, property2text] : Confirmed (by solver)	Rule conflict!

uml2gui_child_stealing.atl

Child stealing

- Example. Consider both owned and inherited features of a class.



uml2gui_child_stealing.atl

Child stealing

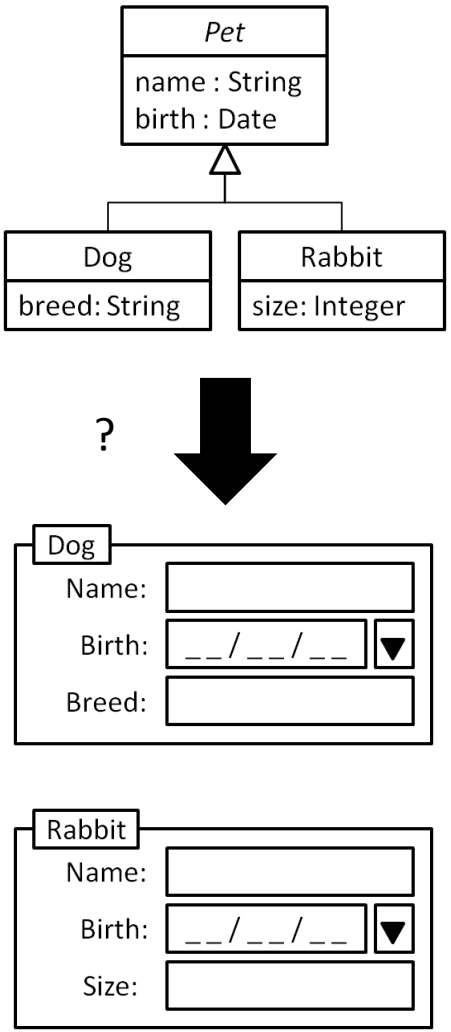
- Change is straightforward:

```
helper context CD!Class def: allAttributes : Sequence(CD!Feature) =
  self.superclasses->collect(c | c.allFeatures)->flatten()
                  ->union(self.ownedAttribute);


rule class2frame {
  from c: CD!Class ( not c.isAbstract )
  to   f: GUI!Frame (
       title <- c.name,
       widgets <- c.ownedAttribute,
       widgets <- c.allAttributes,
       ...
  ), ...
}
```

uml2gui_child_stealing.atl

Child stealing

```
rule class2frame {
  from c : CD!Class ( not c.isAbstract )
  to   f : GUI!Frame (
     title <- c.name,
     widgets <- c.allAttributes
  )
}

rule property2text {
  from p : CD!Property ( p.isText() )
  to   t : GUI!Text
}


rule property2date {
  from p : CD!Property ( p.isDatePicker() )
  to   t : GUI!DatePicker
}
```



uml2gui_child_stealing.atl

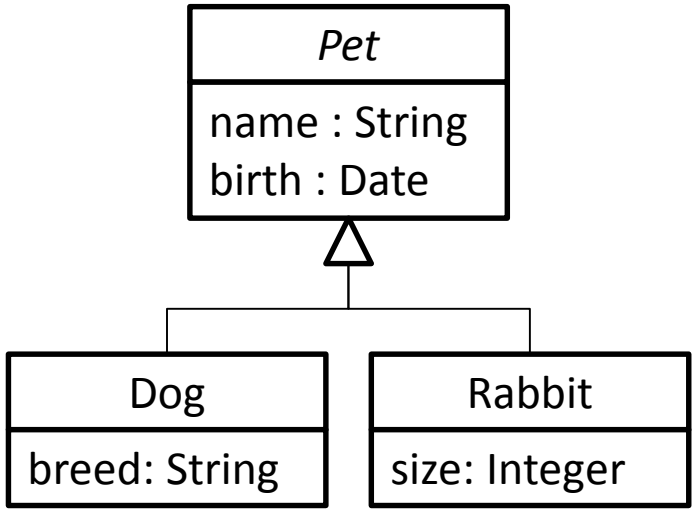
Child stealing

- Double click on “Child stealing”
  - It checks pair of binding which may “steal” objects to each other
  - It may not work with the default configuration because “Class.allAttributes” is recursive
    - Activate “unfold recursion” option

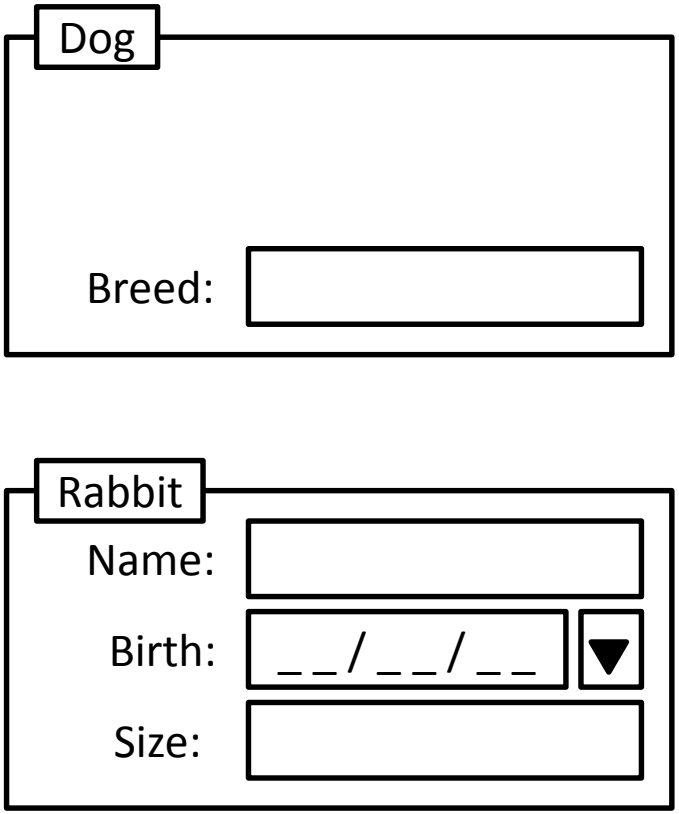
Child stealing analysis	Some conflicts: 2/3
windows (55:4-55:58) and windows (55:4-55:58) and rule class2frame : Discarded (by solver)	
widgets (66:4-66:30) and widgets (66:4-66:30) and rule property2text : Confirmed (by solver)	
widgets (66:4-66:30) and widgets (66:4-66:30) and rule property2date : Confirmed (by solver)	

uml2gui_child_stealing.atl

Child stealing



you get



Why?

Child stealing

- Explanation
  - Each instance of a Property generates one Widget instance.
  - Several activations of the class2frame rule put the same Property in the right part of the binding
    - The last binding wins
  - We need one instance per concrete subclass
- Solutions
  - Use lazy rules
  - Use rule with two input elements (Class + Property)
  - Forbid the scenario (e.g., forbid inheritance)

uml2gui.atl

Target constraint analysis

- Analyse if the transformation will satisfy a given constraint of the target meta-model

```
-- @target_constraint
helper def : rightColumn() : Boolean =
  GUI!GridInfo.allInstances()->forAll(info | info.column >= 1);
```

More details about this in our talk
"Translating source to target constraints in model-to-model transformations"

Wednesday 20th - 11:30

Unconnected components

- Attempts to check the number of (sub-)graphs generated by the transformation
  - E.g., if you forget to add a binding you will have an “unconnected element”
  - Still in development
    - Do not trust it yet

Visualizations

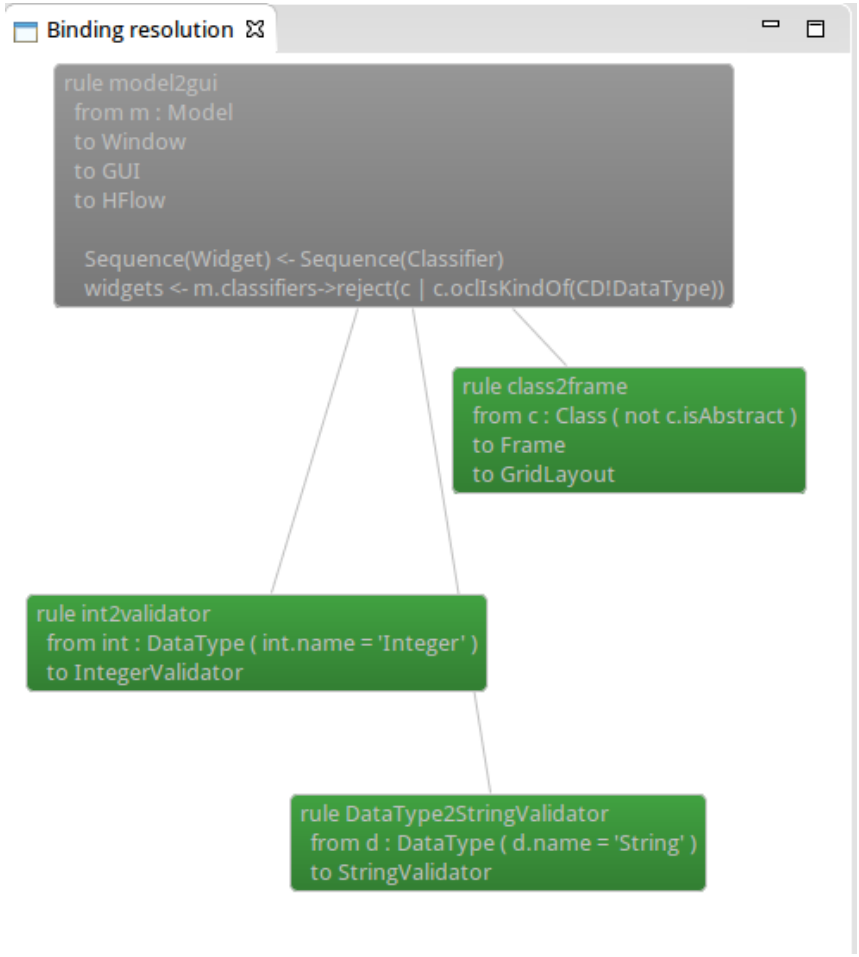
Rule relationships

- We make use of the TDG to provide visualizations about rule relationships
- In ATL,
  - Rules are connected via bindings
  - Relationships between rules are implicit
  - Visualization to make them explicit

Visualization

- Available as quick assist for bindings and also as quick fix for binding errors
  - Currently visualization does not use constraint solving to prune, you get all “possible” resolutions

Visualization



UML Support

Enabling UML support

- Additional analysis for UML meta-models
  - Proof of concept
  - Support for UML profiles

```
-- @nsURI CD=http://www.eclipse.org/uml2/5.0.0/UML
-- @path GUI=/models17.tutorial.uml2gui/metamodels/gui.ecore
--
-- @profile CD=/models17.tutorial.uml2gui/profiles/GUI.profile.uml

module “uml2gui”;
create OUT : GUI from IN : CD;
```

Enabling UML support

- Additional analysis for UML meta-models
  - Check stereotypes
  - Main drawback: model finder will not work

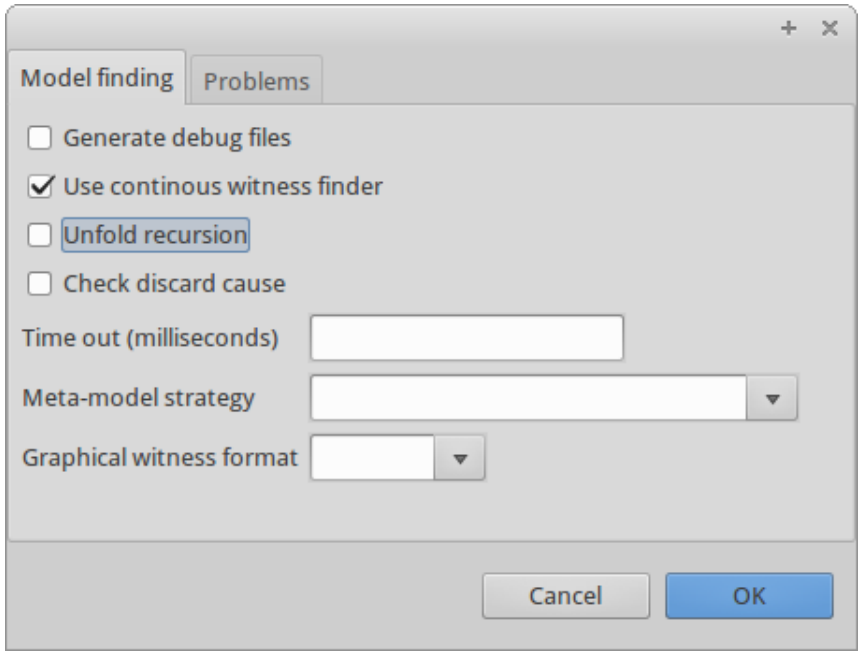
```
rule class2window {
  from c : CD!Class (
     c.getAppliedStereotype('GUI::isUI') <> OclUndefined
  )
  to   f : GUI!Window (
     title <- c.name,
     widgets <- c.ownedAttribute
  )
}
```

Configuration options

Configuration

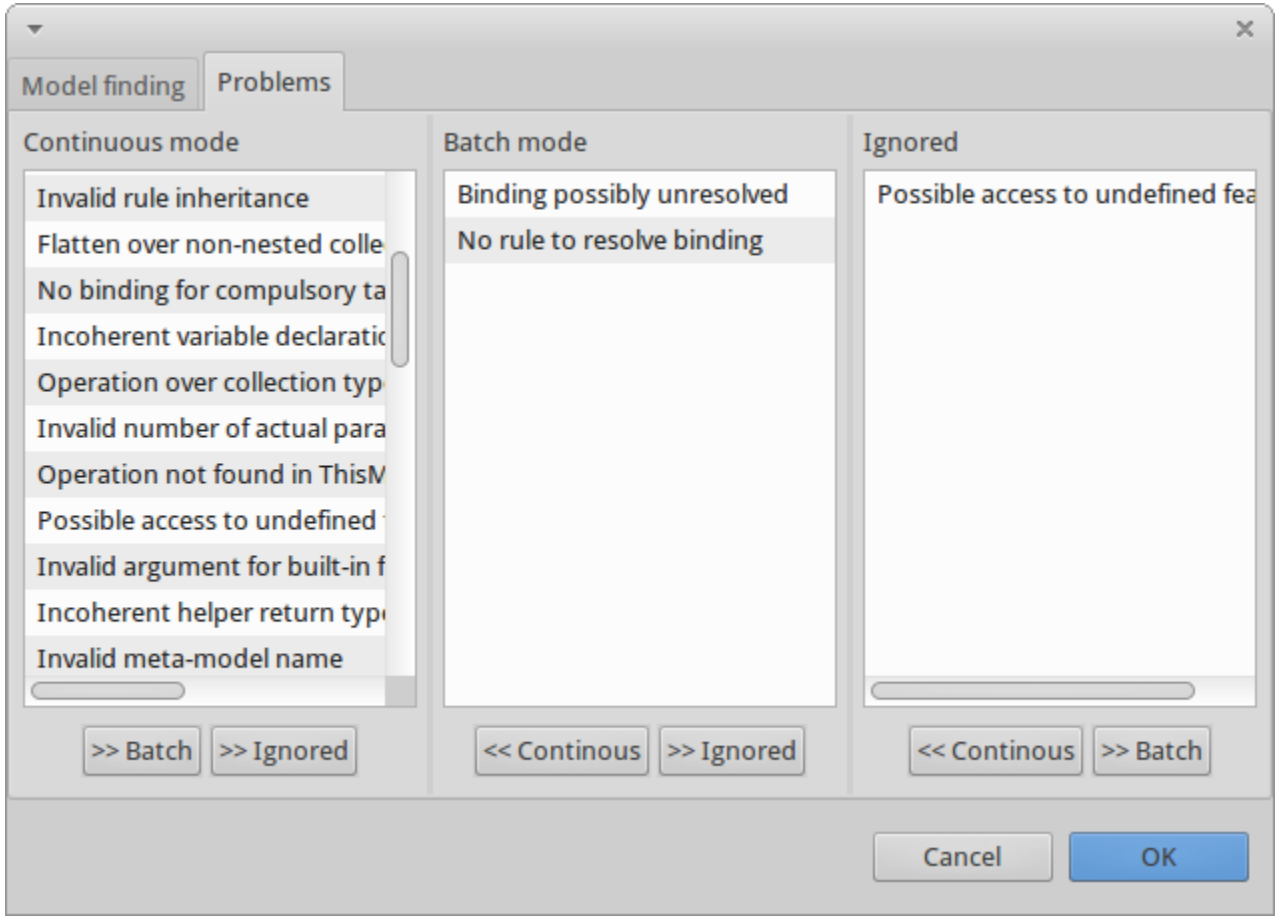
- Eclipse-wide configuration
  - Window -> Preferences -> AnATLyzer
- Transformation-specific configuration
  - Right-click on the ATL file
  - AnATLyzer -> Configure anATLyzer

Configuration

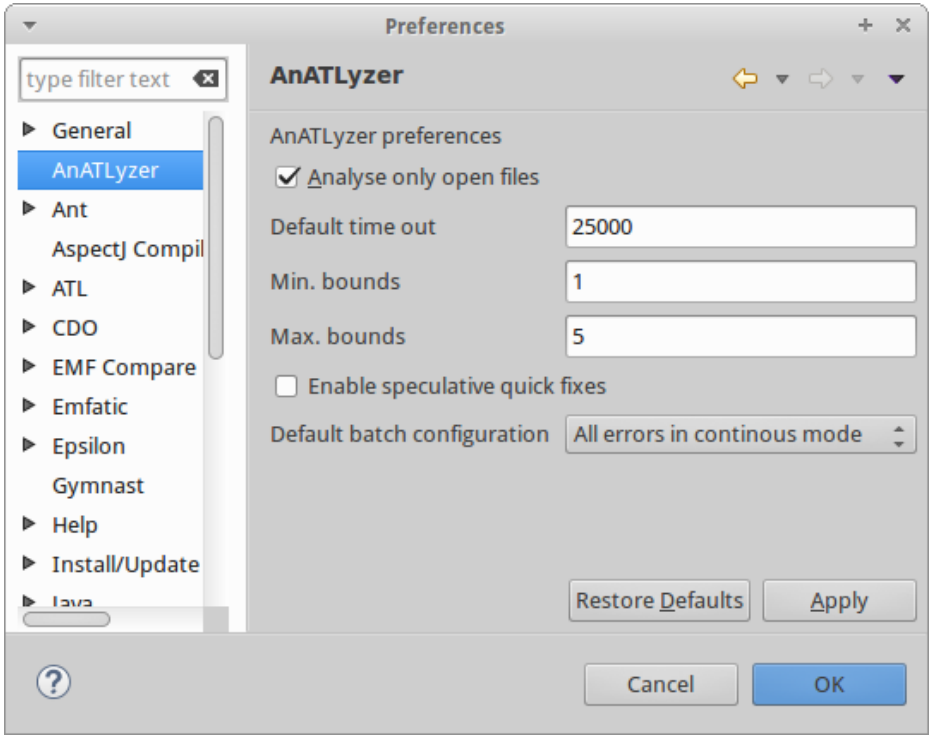


- Continous mode
  - Recommended
  - Untick to execute model finder on demand
- Unfold recursion
  - Experimental support for recursive helpers
- Check discard cause
  - Errors can be discarded due to meta-model issues
- Time out

Configuration



Configuration



- Analyse only open files
  - Recommended to avoid the Eclipse builder to force too many unneeded analysis
- Default time out
  - 2.5 seconds – 5 seconds
- Min./Max. bounds
  - 1..5 is typically enough but you can play according to your computing power
- Enable speculative qfx.
  - Needs too much memory

Configuration

- Default batch configuration
  - Which problems to check if there is no specific .atlc file
  - All errors in continuous mode
    - All errors are checked as the user is editing
  - Model finding on errors
    - Selects for continuous mode only severe errors that require model finding
  - No model finding
    - Delay to batch mode any analysis that requires model finding.

Troubleshooting

Troubleshooting

- The solver does not work (too many unknowns)
  - If there are errors in the error path the solver is likely to fail
  - There are features which are not supported by USE Validator
  - Bugs and limitations in the translation

Troubleshooting

- The solver takes too much memory
  - It is likely that there are some memory leaks
  - We will work harder, sorry!

  - Move some of the solver-based problems to batch mode. This is the recommended order:
    - Unresolved binding
    - Feature defined in subtype
    - Access to undefined value
    - Binding resolved by invalid target
      - Do not move this if you can afford the solver time

Troubleshooting

- Suddenly, all error markers are vanished and the Analysis View is empty
  - Probably there is an internal error in AnATLyzer
  - Please, send us the bug

Troubleshooting

- Typing
  - Type inference for (mutually) recursive helpers may lead to false positives sometimes
  - Try with @force-declared-return-type

Limitations

- Many!
  - Including fixing bugs
  - Technical issues:
    - After undo we need to reload…
- Mapping to USE Validator
  - We have good coverage but we have to work on e.g., Map and Tuple support
- Cannot re-analyse dependent transformations or changes in the meta-model
  - Lack of standard mega-model

More information

- Available at Github
  - https://github.com/jesusc/anatlyzer

- Send me an e-mail if:
  - You want to use it and have some problem
  - You have found a bug
  - You want to collaborate

jesus.sanchez.cuadrado@gmail.com