Combining TGGs with ILP solving for consistency checks

Erhan Leblebici, BX – Shonan 09/2016

ES Real-Time Systems Lab

Prof. Dr. rer. nat. Andy Schürr

Dept. of Electrical Engineering and Information Technology

Dept. of Computer Science (adjunct Professor)

www.es.tu-darmstadt.de

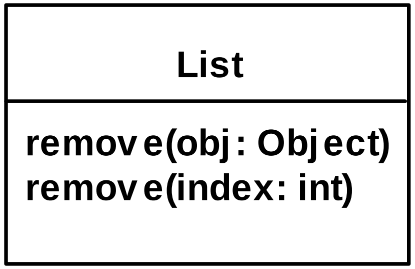
Erhan.Leblebici@es.tu-darmstadt.de

02.11.2016

The goals of a consistency check are to

- Provide a Yes or No indicating whether the models are consistent
- Provide traceability links referring to consistent parts (at least in case of TGGs)
- Point at inconsistent model parts

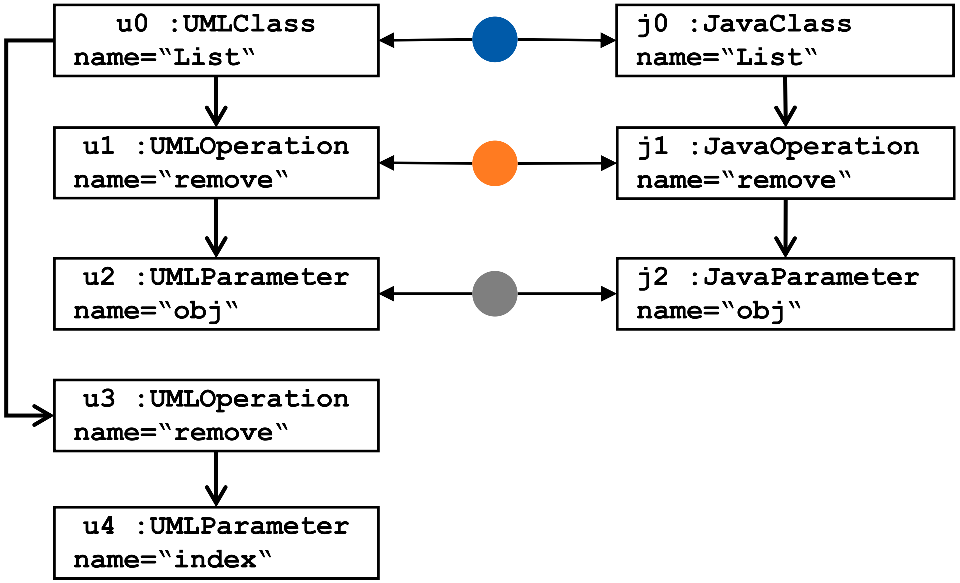
A simple 1:1 relation between UML and Java


List

remove(obj: Object)
remove(index: int)

```
class List
{
    void remove (Object obj){…}
}
```

What to do in case of inconsistency?



u0 :UMLClass
name=‶List‶

j0 :JavaClass
name=‶List‶

u1 :UMLOperation
name=‶remove‶

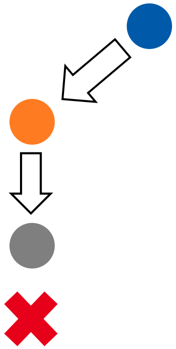
j1 :JavaOperation
name=‶remove‶

u2 :UMLParameter
name=‶obj‶

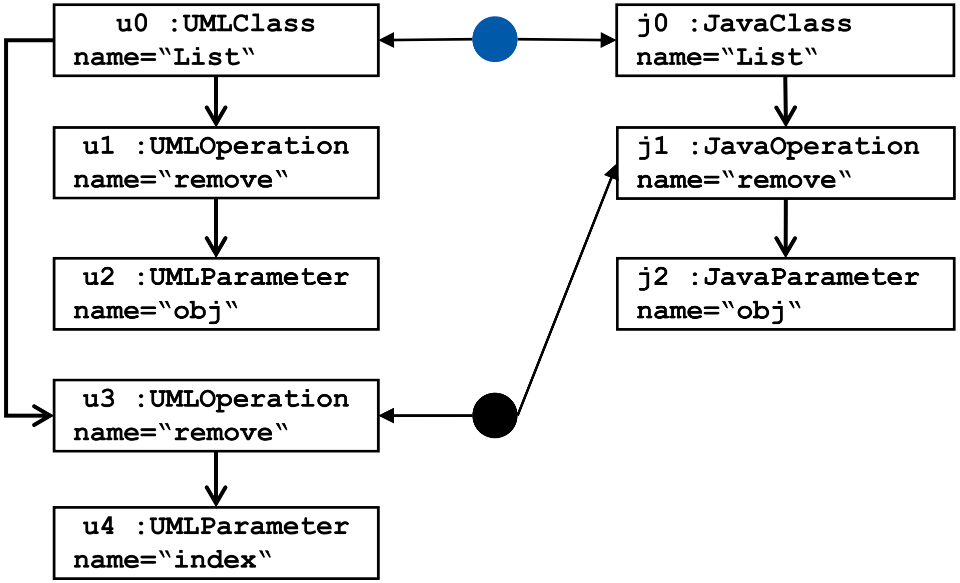
j2 :JavaParameter
name=‶obj‶

u3 :UMLOperation
name=‶remove‶

u4 :UMLParameter
name=‶index‶



A worse alternative



u0 :UMLClass
name=“List“

u1 :UMLOperation
name=“remove“

u2 :UMLParameter
name=“obj“

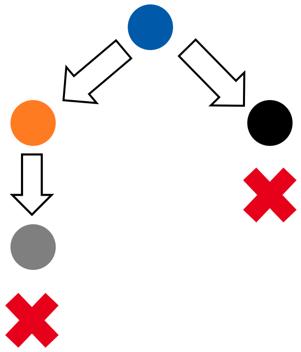
u3 :UMLOperation
name=“remove“

u4 :UMLParameter
name=“index“

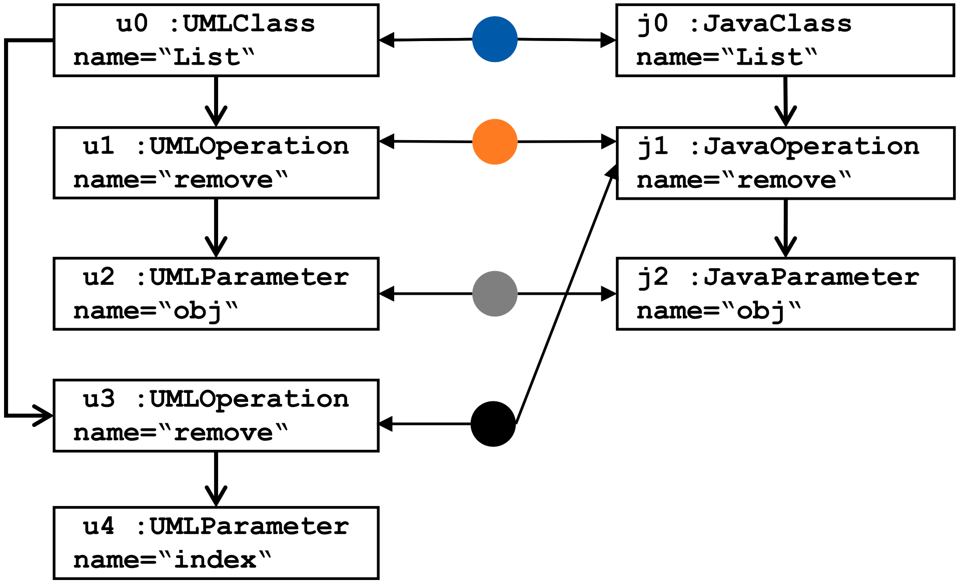
j0 :JavaClass
name=“List“

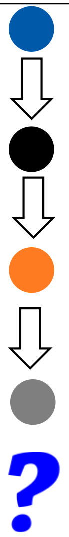
j1 :JavaOperation
name=“remove“

j2 :JavaParameter
name=“obj“



Formulate constraints between single steps

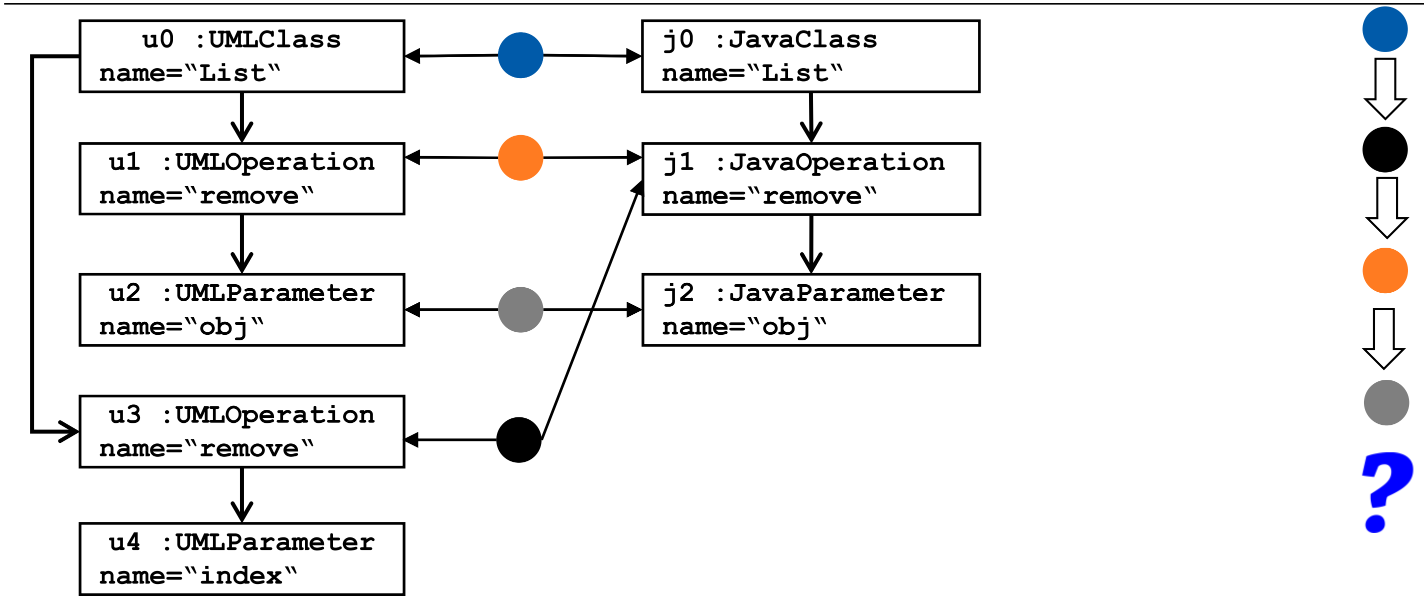




Exclusions	
j1	🟠 + ⚫ ≤ 1

Implications
🔘 ≤ 🟠

And define an objective

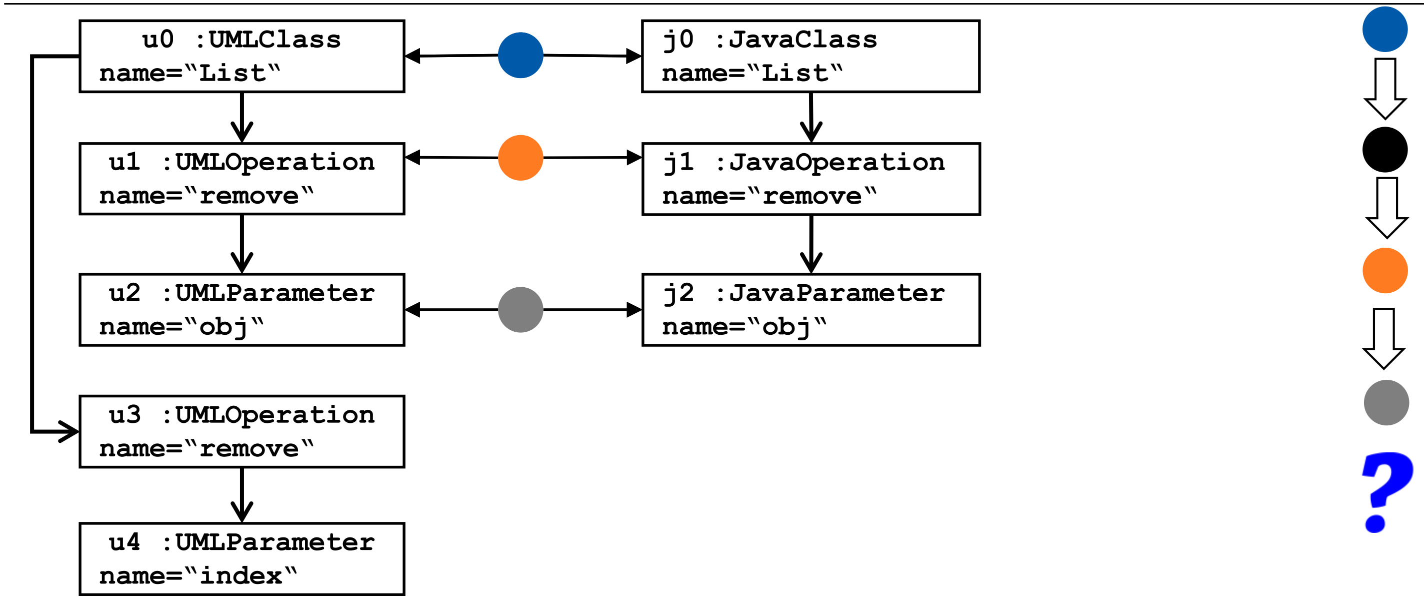


Exclusions	
j1	🟠 + ⚫ ≤ 1

Implications
🔘 ≤ 🟠

max 2*🔵 + 4*🟠 + 4*⚫ + 4*🔘

Retain chosen ones w.r.t. the objective



u0 :UMLClass
name="List"

j0 :JavaClass
name="List"

u1 :UMLOperation
name="remove"

j1 :JavaOperation
name="remove"

u2 :UMLParameter
name="obj"

j2 :JavaParameter
name="obj"

u3 :UMLOperation
name="remove"

u4 :UMLParameter
name="index"

Exclusions	
j1	🟠 + ⚫ ≤ 1

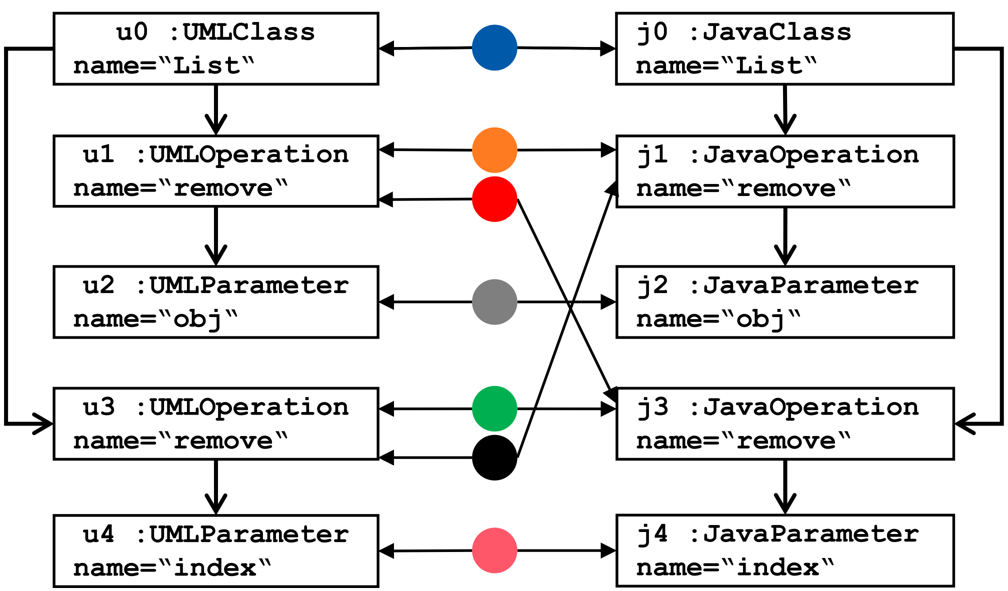
Implications
● ≤ 🟠

max 2*🔵 + 4*🟠 + 4*⚫ + 4*●

What are reasonable objectives of such an optimization problem?

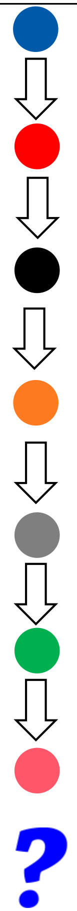
- Default: Maximize the number of related source and target model elements
  - Consistency can be conclued if all elements on both sides are in a relation
- Prefer covering source (target) elements
  - E.g., I‘m still happy if all UML elements are related to my Java code but not necessarily vice versa
- Prefer elements with a certain type, attribute, ...
- …

Example for consistent models (all mappings)



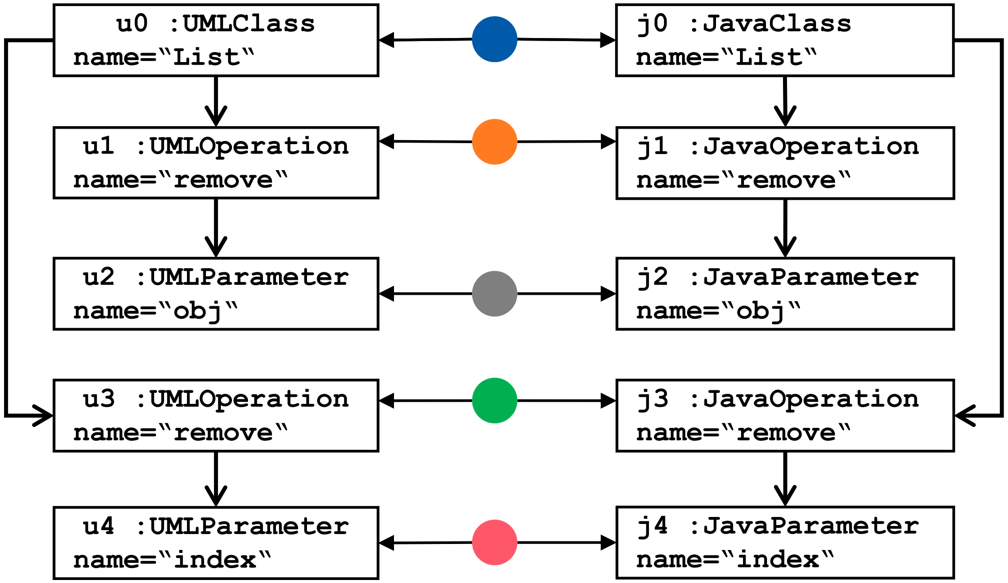
Exclusions	
u1	🟠 + 🔴 ≤ 1
j1	🟠 + ⚫ ≤ 1
u3	🟢 + ⚫ ≤ 1
j3	🟢 + 🔴 ≤ 1

Implications
🩶 ≤ 🟠
🩷 ≤ 🟢



max 2*🔵 + 4*🟠 + 4*⚫ + 4*🩶 + 4*🔴 + 4*🩷 + 4*🟢

…and retain correct mappings



Exclusions	
u1	🟠 + 🔴 ≤ 1
j1	🟠 + ⚫ ≤ 1
u3	🟢 + ⚫ ≤ 1
j3	🟢 + 🔴 ≤ 1

Implications
⚪ ≤ 🟠
🩷 ≤ 🟢

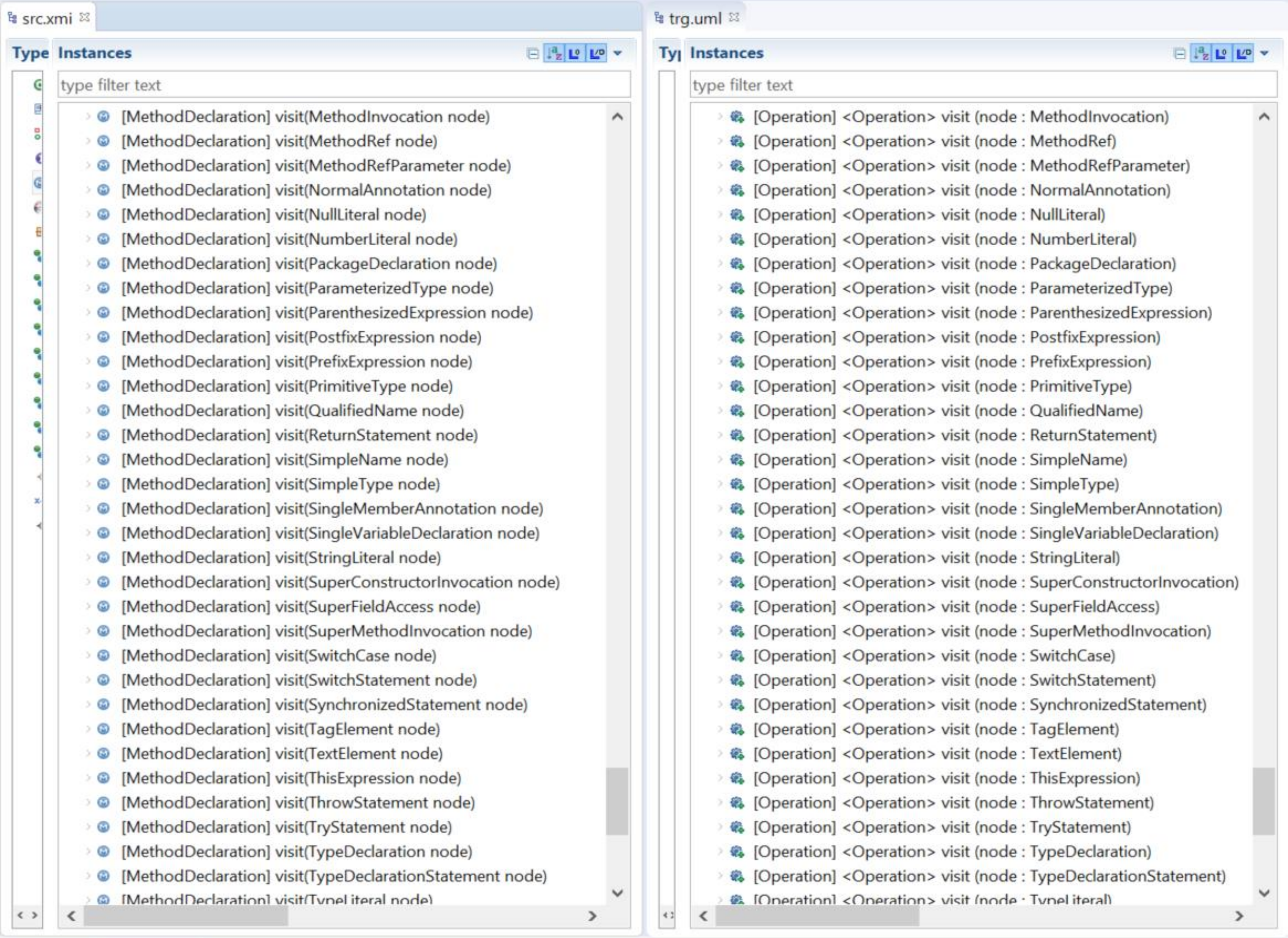
🔵
⇩
🔴
⇩
⚫
⇩
🟠
⇩
⚪
⇩
🟢
⇩
🩷

?

max 2*🔵 + 4*🟠 + 4*⚫ + 4*⚪ + 4*🔴 + 4*🩷 + 4*🟢

Runtime measurements

name	# class	# method	# param	ALL MAPPINGS	ELIMINATED MAPPINGS	Gra-Tra	ILP
org.moflon.ide.core	294	546	505	1853	72	8 sec	1 sec
modisco.java.discoverer	415	1356	1057	16610	13204	20 sec	11 sec
graphiti	467	2246	2780	7828	1472	76 sec	15 sec
org.eclipse.compare	977	3160	3528	10175	816	109 sec	26 sec



```
[MethodDeclaration] visit(MethodInvocation node)
[MethodDeclaration] visit(MethodRef node)
[MethodDeclaration] visit(MethodRefParameter node)
[MethodDeclaration] visit(NormalAnnotation node)
[MethodDeclaration] visit(NullLiteral node)
[MethodDeclaration] visit(NumberLiteral node)
[MethodDeclaration] visit(PackageDeclaration node)
[MethodDeclaration] visit(ParameterizedType node)
[MethodDeclaration] visit(ParenthesizedExpression node)
[MethodDeclaration] visit(PostfixExpression node)
[MethodDeclaration] visit(PrefixExpression node)
[MethodDeclaration] visit(PrimitiveType node)
[MethodDeclaration] visit(QualifiedName node)
[MethodDeclaration] visit(ReturnStatement node)
[MethodDeclaration] visit(SimpleName node)
[MethodDeclaration] visit(SimpleType node)
[MethodDeclaration] visit(SingleMemberAnnotation node)
[MethodDeclaration] visit(SingleVariableDeclaration node)
[MethodDeclaration] visit(StringLiteral node)
[MethodDeclaration] visit(SuperConstructorInvocation node)
[MethodDeclaration] visit(SuperFieldAccess node)
[MethodDeclaration] visit(SuperMethodInvocation node)
[MethodDeclaration] visit(SwitchCase node)
[MethodDeclaration] visit(SwitchStatement node)
[MethodDeclaration] visit(SynchronizedStatement node)
[MethodDeclaration] visit(TagElement node)
[MethodDeclaration] visit(TextElement node)
[MethodDeclaration] visit(ThisExpression node)
[MethodDeclaration] visit(ThrowStatement node)
[MethodDeclaration] visit(TryStatement node)
[MethodDeclaration] visit(TypeDeclaration node)
[MethodDeclaration] visit(TypeDeclarationStatement node)
[MethodDeclaration] visit(TypeLiteral node)
```

```
[Operation] <Operation> visit (node : MethodInvocation)
[Operation] <Operation> visit (node : MethodRef)
[Operation] <Operation> visit (node : MethodRefParameter)
[Operation] <Operation> visit (node : NormalAnnotation)
[Operation] <Operation> visit (node : NullLiteral)
[Operation] <Operation> visit (node : NumberLiteral)
[Operation] <Operation> visit (node : PackageDeclaration)
[Operation] <Operation> visit (node : ParameterizedType)
[Operation] <Operation> visit (node : ParenthesizedExpression)
[Operation] <Operation> visit (node : PostfixExpression)
[Operation] <Operation> visit (node : PrefixExpression)
[Operation] <Operation> visit (node : PrimitiveType)
[Operation] <Operation> visit (node : QualifiedName)
[Operation] <Operation> visit (node : ReturnStatement)
[Operation] <Operation> visit (node : SimpleName)
[Operation] <Operation> visit (node : SimpleType)
[Operation] <Operation> visit (node : SingleMemberAnnotation)
[Operation] <Operation> visit (node : SingleVariableDeclaration)
[Operation] <Operation> visit (node : StringLiteral)
[Operation] <Operation> visit (node : SuperConstructorInvocation)
[Operation] <Operation> visit (node : SuperFieldAccess)
[Operation] <Operation> visit (node : SuperMethodInvocation)
[Operation] <Operation> visit (node : SwitchCase)
[Operation] <Operation> visit (node : SwitchStatement)
[Operation] <Operation> visit (node : SynchronizedStatement)
[Operation] <Operation> visit (node : TagElement)
[Operation] <Operation> visit (node : TextElement)
[Operation] <Operation> visit (node : ThisExpression)
[Operation] <Operation> visit (node : ThrowStatement)
[Operation] <Operation> visit (node : TryStatement)
[Operation] <Operation> visit (node : TypeDeclaration)
[Operation] <Operation> visit (node : TypeDeclarationStatement)
[Operation] <Operation> visit (node : TypeLiteral)
```

Final remarks

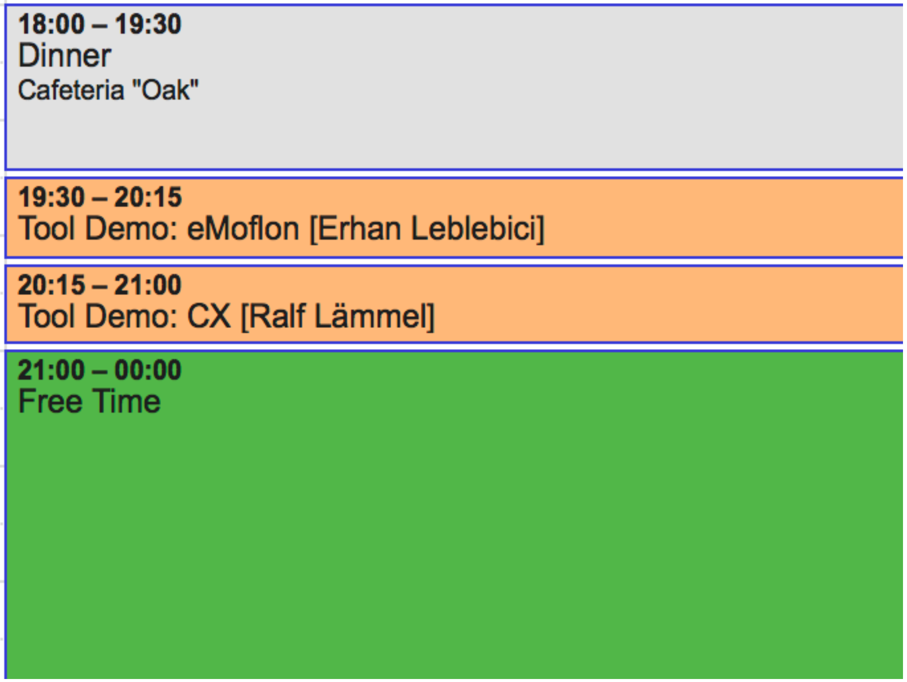
We actually try to kill two birds with one stone:

- Outsourcing decision making to constraint solving (implementation is arguably more managable than complex algorithms)
- Formulating consistency as an optimization problem

First results seem promising

The idea can be transfered to other use cases (model synchronization)

Hope to see you in the demo session



18:00 – 19:30
Dinner
Cafeteria "Oak"

19:30 – 20:15
Tool Demo: eMoflon [Erhan Leblebici]

20:15 – 21:00
Tool Demo: CX [Ralf Lämmel]

21:00 – 00:00
Free Time

I will perform some consistency checks as well