Static Analysis of Model Transformations for Effective Test Generation

Jean-Marie Mottu[2],**Sagar Sen[1]**, Massimo Tisi[3], Jordi Cabot[3]

[1]**Certus V&V Center, Simula Research Laboratory, Oslo**

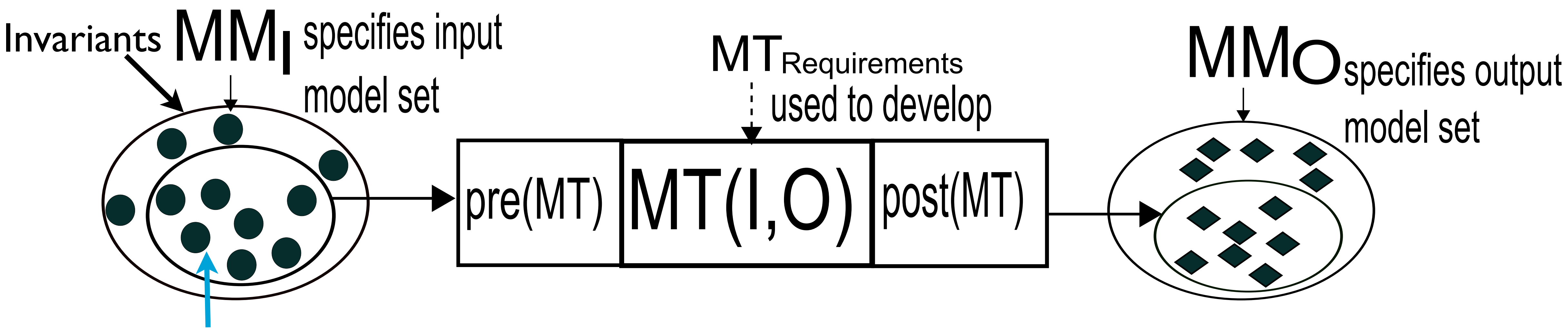
[2]AeLoS, Universite de Nantes, France

[3]ATLANMOD, Ecole des Mines, Nantes

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis

Introduction



Effective test models!

Model Transformation Testing

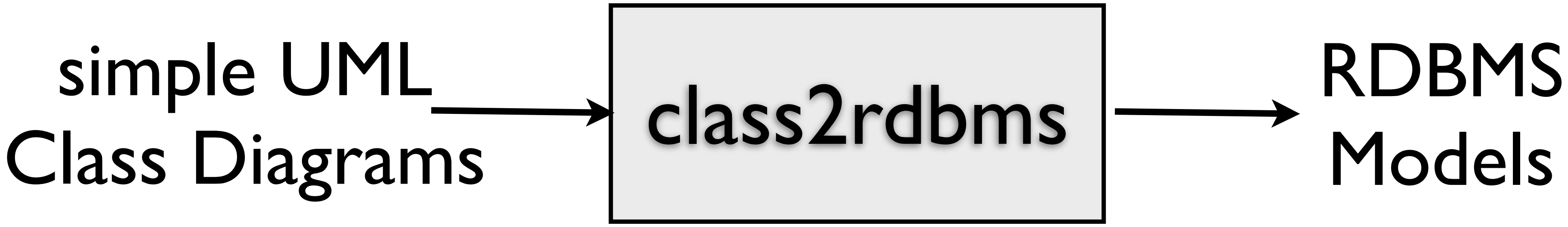
Examples

1. Compilers (Java to Bytecode)
2. Code generators (UML Statemachine to code)
3. Structured data format transformation (XML to XML/text)
4. Object persistence (Class to RDBMS)

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis

Case study: class2rdbms



simple UML Class Diagrams → class2rdbms → RDBMS Models

1. Object persistence **benchmark** proposed in the MTIP workshop, MoDELS 2005

2. Input domain spec. **covers all major metamodelling** concepts such as inheritance, composition, finite and infinite multiplicities.

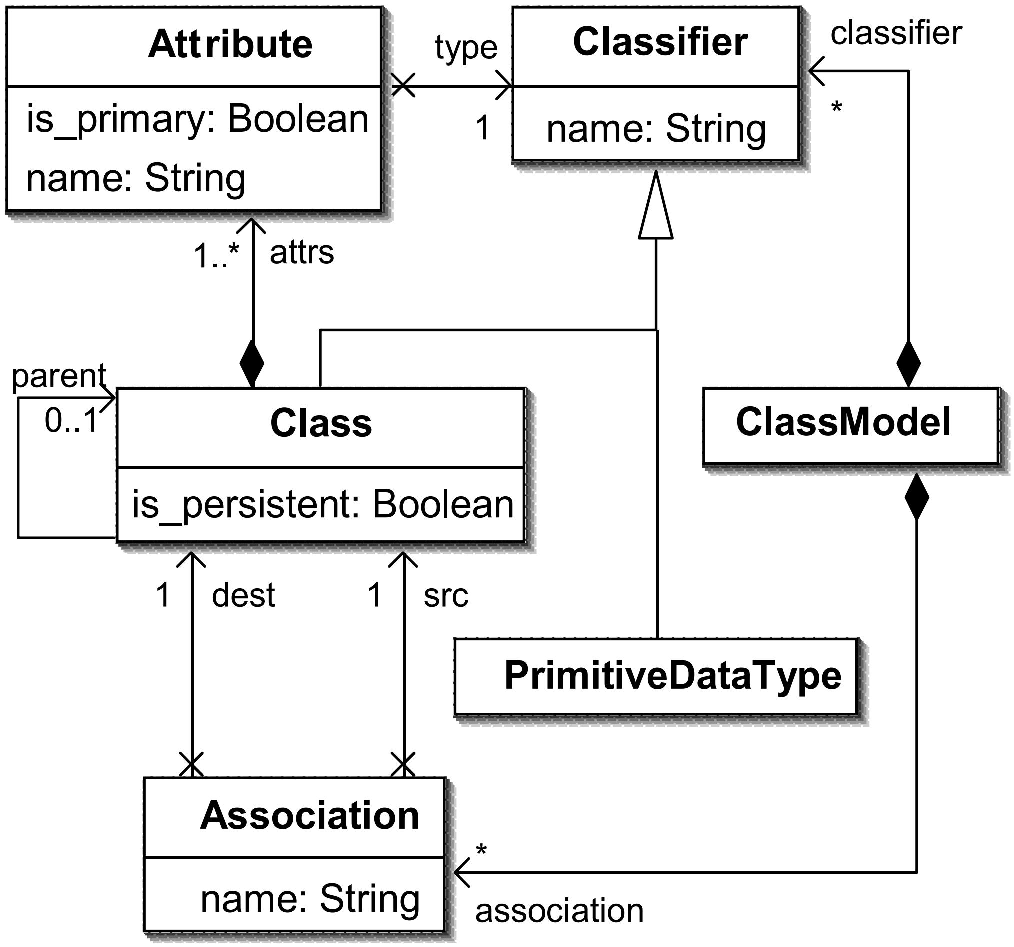
3. **Invariants** are both first-order and high-order, contains also transitive closure invariants

4. Transformation exercises most **major model transformation operators** such as navigation, creation, and filtering

5. Available in **many transformation languages** Kermeta, ATL, VIATRA, QVT

Input MM + Invariants

Ecore Meta-model



(a)

OCL Invariants

```
context Class

    inv noCyclicInheritance:
        not self.allParents()->includes(self)

    inv uniqueAttributesName:
        self.attrs->forAll(att1, att2 |
            att1.name=att2.name implies att1=att2)

context ClassModel

    inv uniqueClassifierNames:
        self.classifier->forAll(c1, c2 |
            c1.name=c2.name implies c1=c2)

    inv uniqueClassAssociationSourceName :
        self.association->forAll(ass1, ass2 |
            ass1.name=ass2.name implies
            (ass1=ass2 or ass1.src != ass2.src))
```

(b)

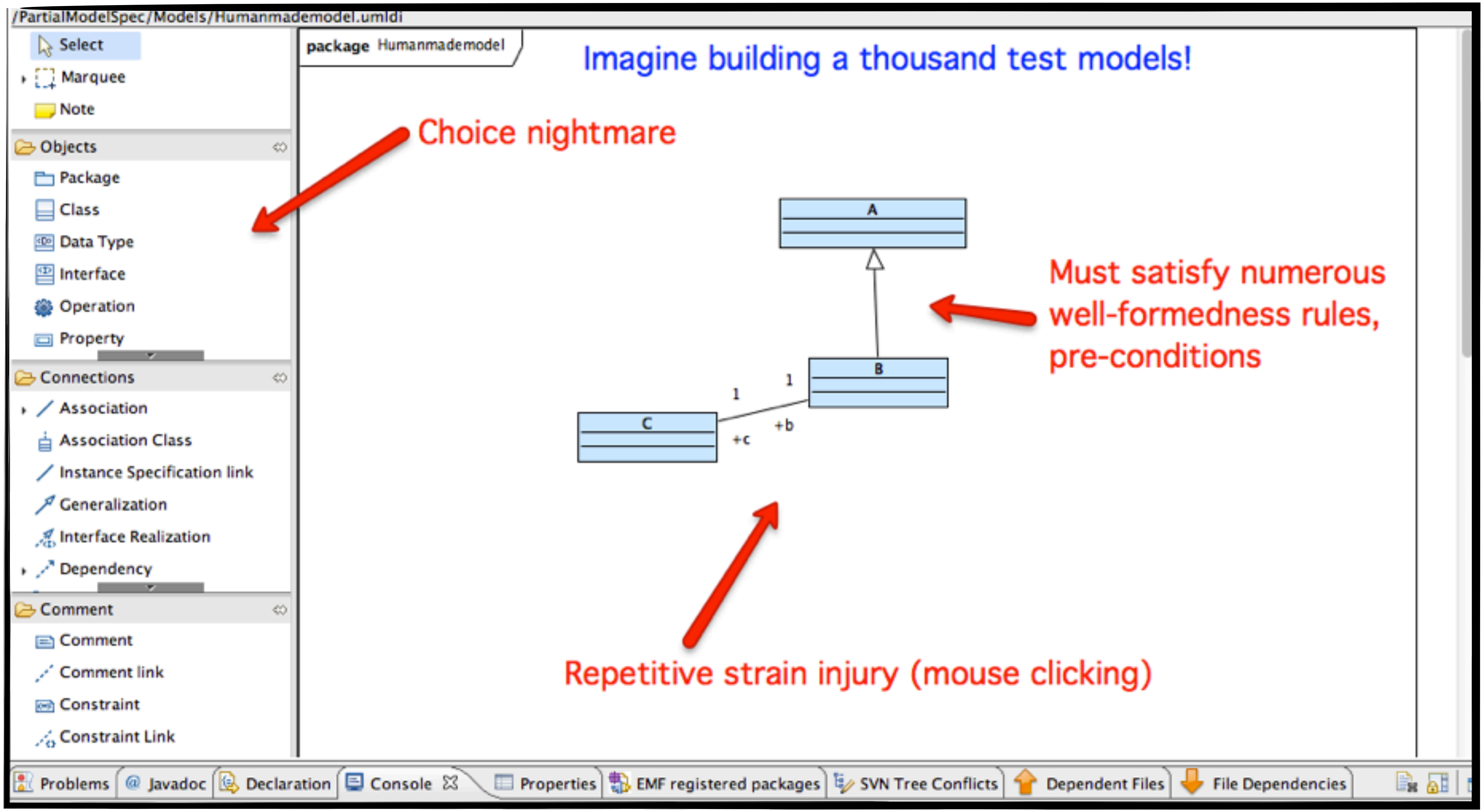
(a) Input metamodel MM_i : Simplified UML CD

(b) A subset of all invariants on MM_i (9 invariants)

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis

Tediousness of Creating Test Models



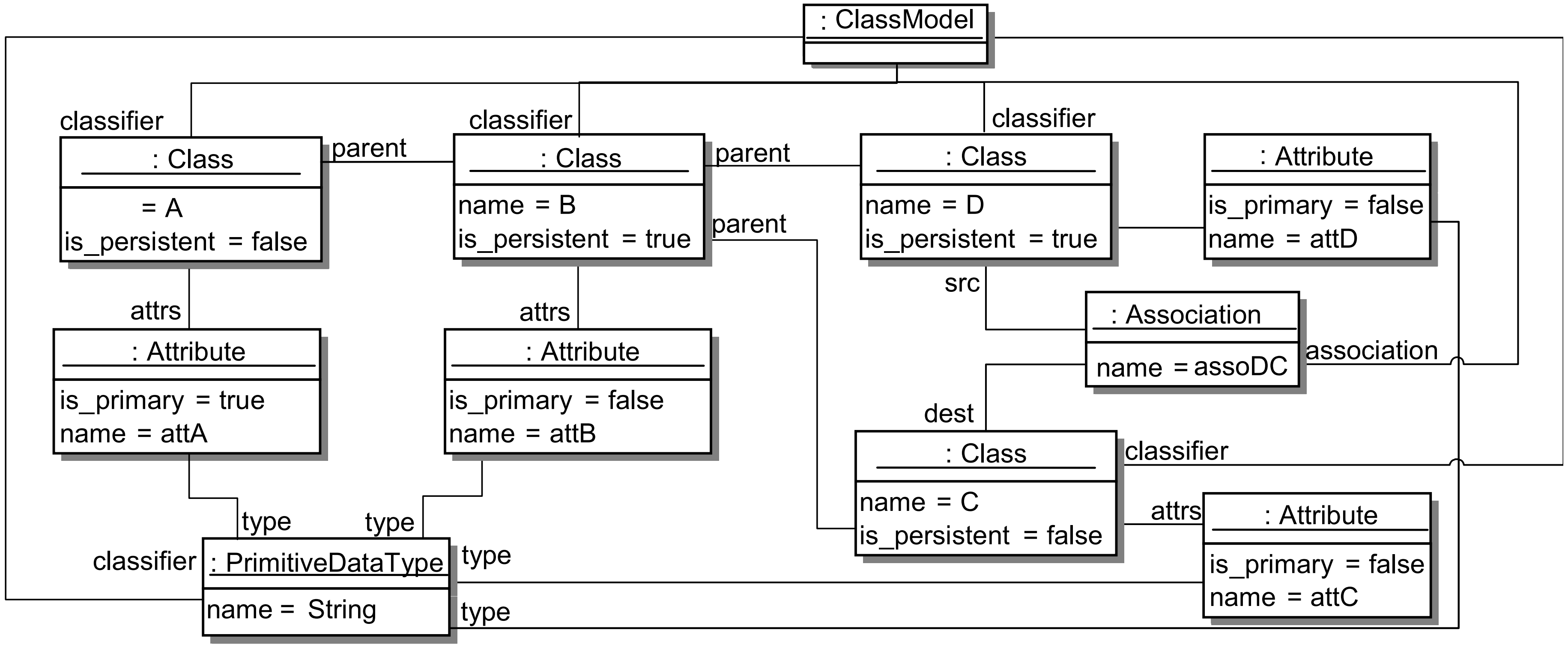
Imagine building a thousand test models!

Choice nightmare

Must satisfy numerous well-formedness rules, pre-conditions

Repetitive strain injury (mouse clicking)

Tediousness of Creating Test Models



A Human-made Test Model

Problems

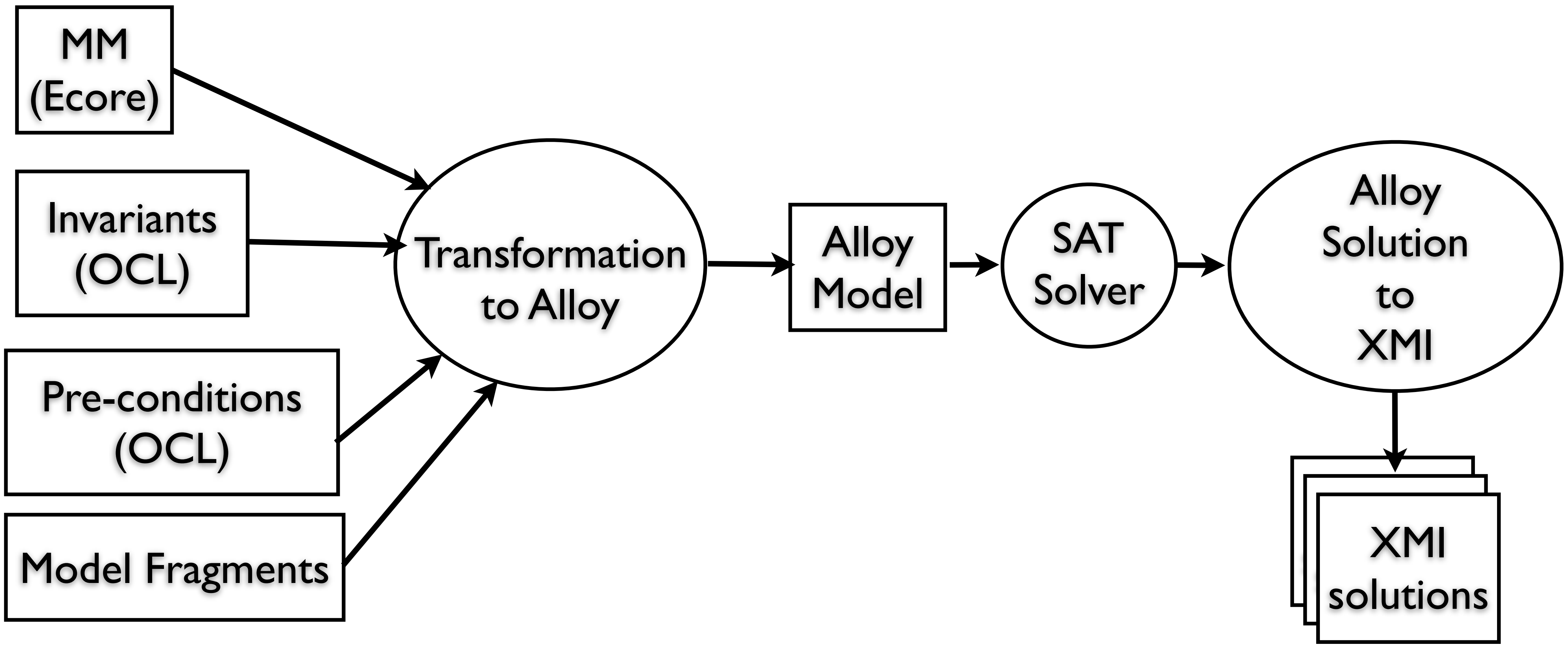
1. Must conform to metamodel **MM$_i$**
2. Must satisfy **MM$_i$** invariants (9 invariants)
3. Must satisfy pre-conditions **pre(MT)** on model transformation (class2rdbms in our case, with 22 pre-condition invariants)
4. Must contain **test knowledge** to find bugs

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis

Story so far!(1)

- **2008:** *How to generate models that satisfy knowledge from heterogeneous sources?*
- **Published in:** Sen et. al. On Combining Multi-formalism Knowledge to Select Test Models, ICST 2008



MM (Ecore)

Invariants (OCL)

Pre-conditions (OCL)

Model Fragments

Transformation to Alloy

Alloy Model

SAT Solver

Alloy Solution to XMI

XMI solutions

Story so far!(1)

- **2008:** *How to generate models that satisfy knowledge from heterogeneous sources?*
- **Published in:** Sen et. al. On Combining Multi-formalism Knowledge to Select Test Models, ICST 2008

Source Element | **Target Alloy Construct**

Classes to Alloy Signatures → auto

```
sig Class extends Classifier
{
        is_persistent: one Bool,
        parent : lone Class,
        attrs : some Attribute
 }
```

Implicit constraints to Alloy Facts → auto

```
fact associationContainment
{
all a:Association | a in ClassModel.association
}
```

Invariants/pre-conditions to Alloy Facts → manual

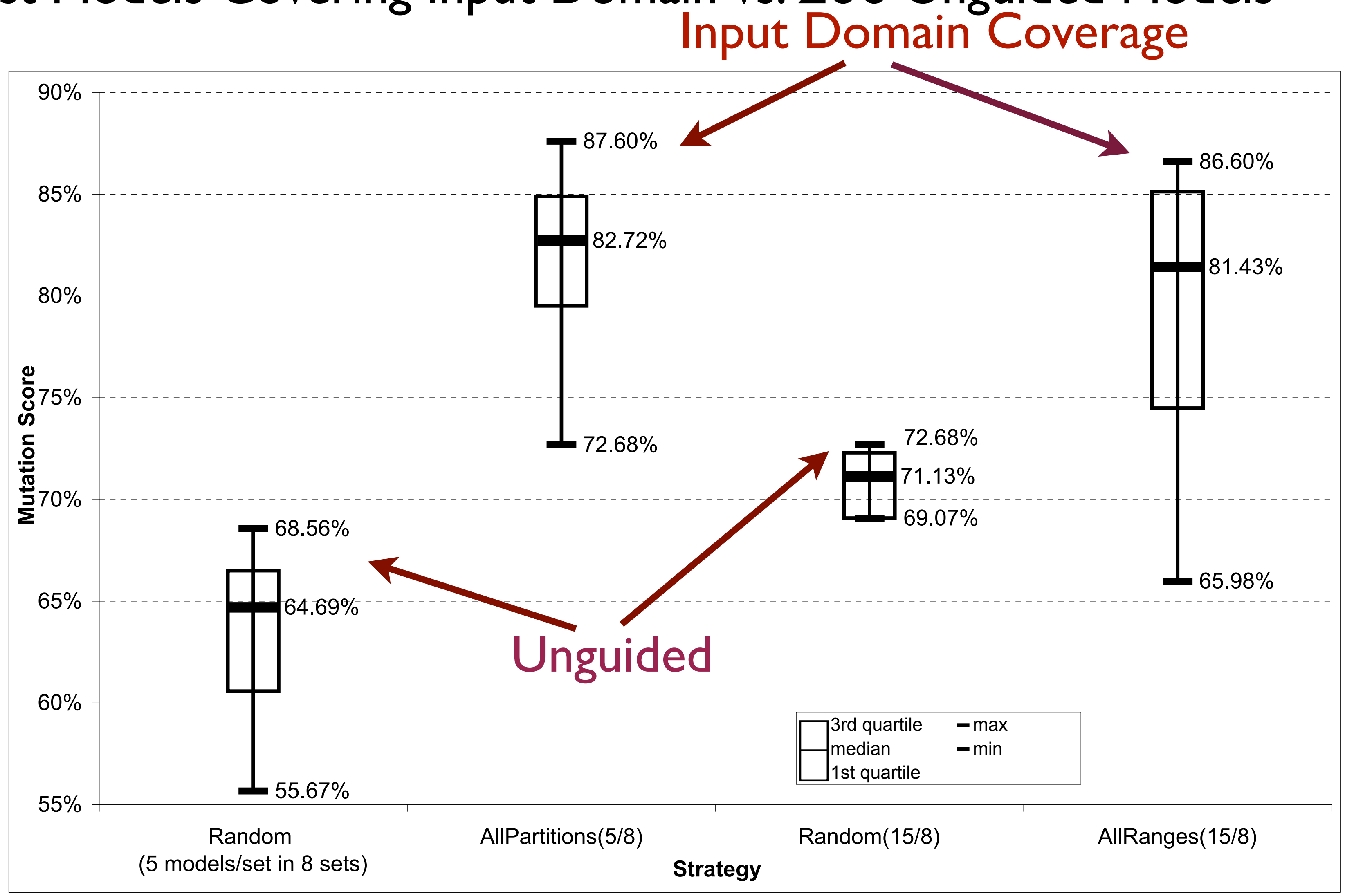
```
fact noCyclicInheritance
{
        no c: Class | c in c.^parent
}
```

Story so far!(2)

- **2009:** *How to test models satisfying coverage criteria and how to validate the quality of these test models?*

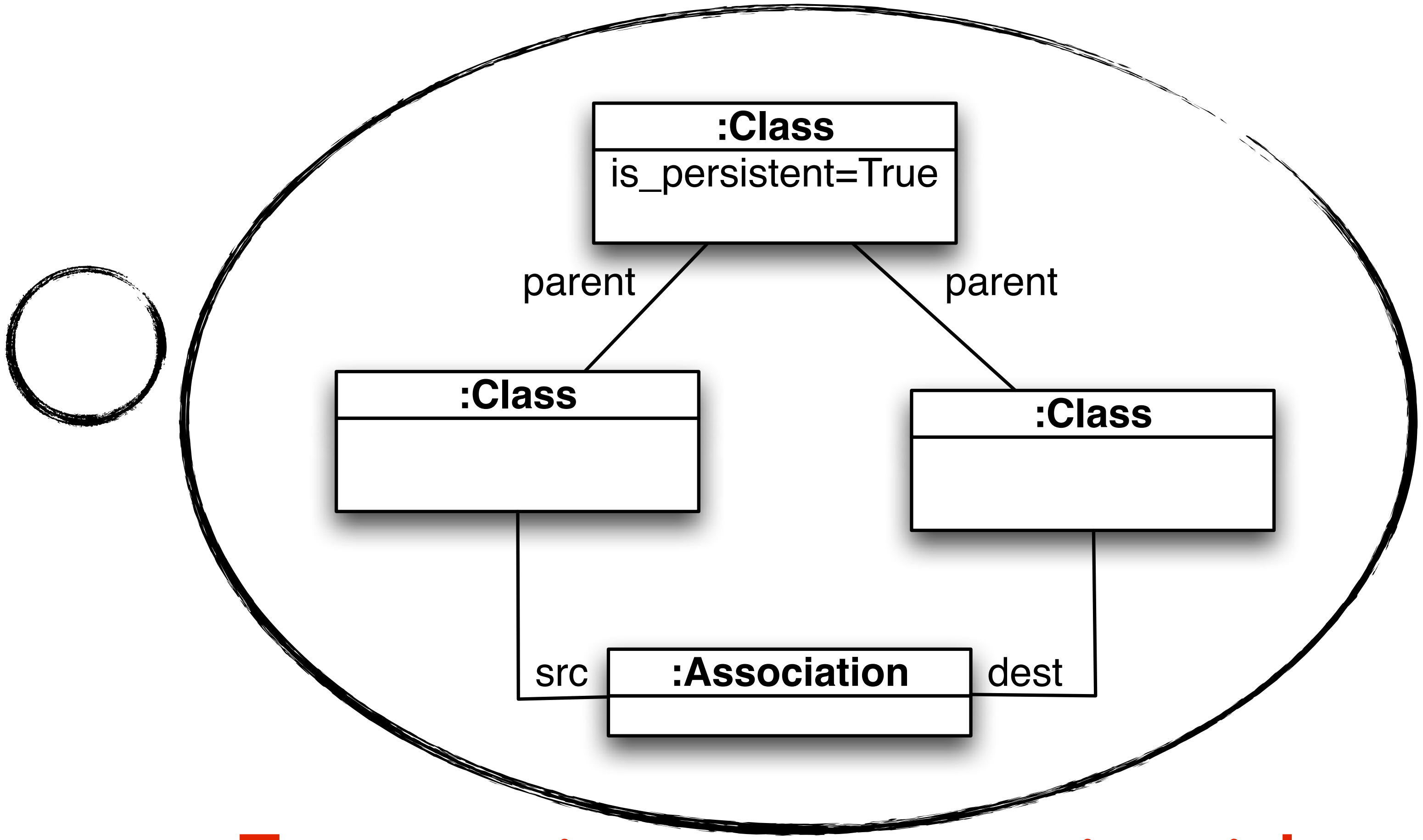
- **Published in:** Sen et. al. Automatic Model Generation Strategies for Model Transformation Testing. ICMT 2009

- **40** Test Models Covering Input Domain vs. **200** Unguided Models



Story so far! (3)

- **2011:** How use “partial knowledge” by introducing a human-in-the-loop for test model generation?
- Published in: Sen S.,et. al. Using Models of Partial Knowledge to Test Model Transformations. ICMT 2012



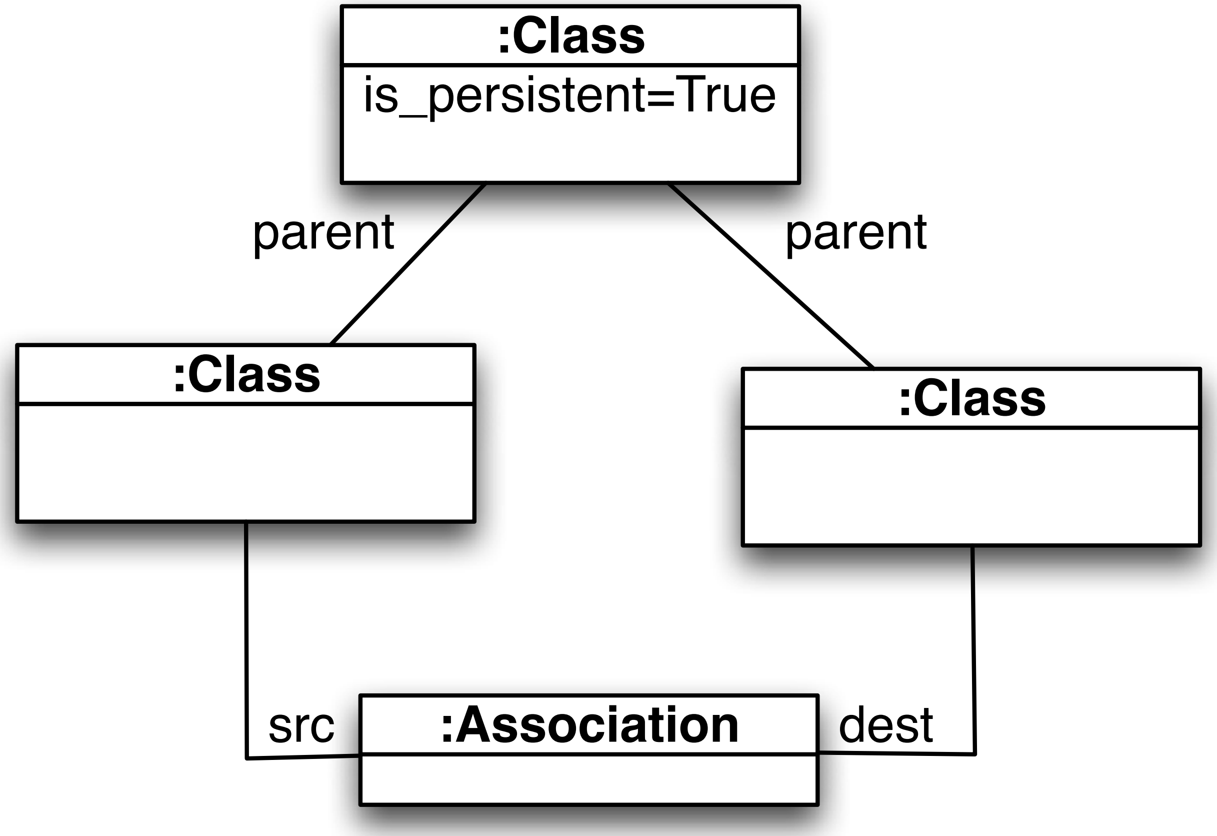
Expressing pure testing ideas as partial models!

Story so far! (3)

- **2011:** How use “partial knowledge” by introducing a human-in-the-loop for test model generation?

- Published in: Sen S.,et. al. Using Models of Partial Knowledge to Test Model Transformations. ICMT 2012

Partial Model



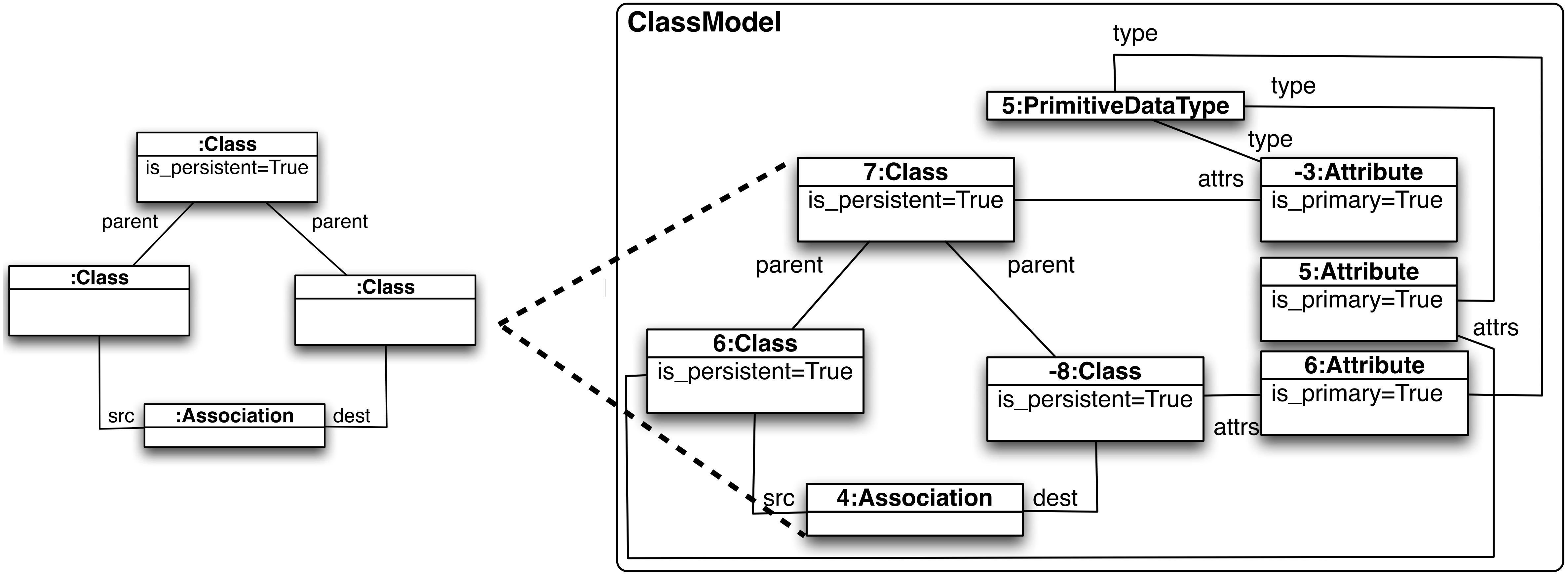
Rewritten Alloy Model

```
pred PartialModel
{
       some c1 : Class, c2: Class, c3:Class |
      c1!=c2 and c2!=c3 and
      c1.is_persistent=True and
      c2.parent = c1 and c3.parent=c1
      and c2!=c3 and c2!=c1 and
      some a1: Association |
      a1.src =c2 and a1.dest=c3
}
```

Story so far! (3)

- **2011:** How use “partial knowledge” by introducing a human-in-the-loop for test model generation?

- Published in: Sen S.,et. al. Using Models of Partial Knowledge to Test Model Transformations. ICMT 2012



Partial models when completed **give 100% mutation** score

just like human-made complete models with the same knowledge.

Question for this talk!

Premise

- Partial testing knowledge is effective if the source is a human expert
- Model transformations themselves are **human-made**. Do they contain testing knowledge? Why cannot we use them as a source?

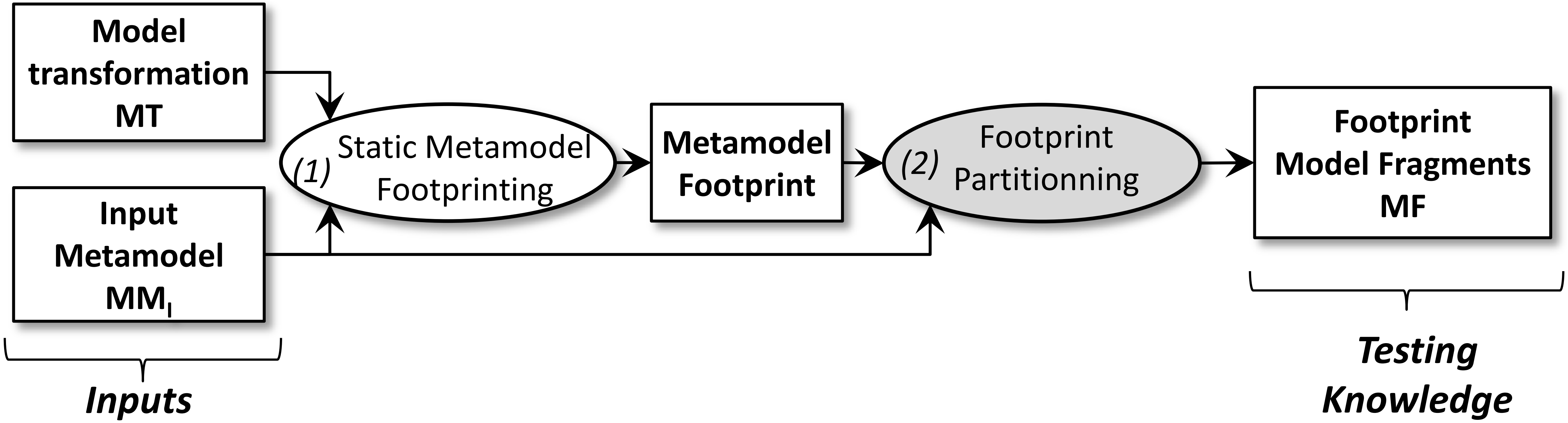
Question

Can we **extract effective testing knowledge** via **static analysis** of a model transformation?

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis
- Conclusion

Part 1: Extracting Footprints


Model transformation MT
Input Metamodel MM_I
Inputs
(1) Static Metamodel Footprinting
Metamodel Footprint
(2) Footprint Partitionning
Footprint Model Fragments MF
Testing Knowledge

Static Metamodel Footprinting (1)

C. Jeanneret, M. Glinz, and B. Baudry. Estimating footprints of model operations. *ICSE'11,* Honolulu, USA, May 2011. IEEE.

<Operation, Feature, Type>

Operation	**Metamodel Feature**	**Types**
getAllClasses	Classifier	Classifier
getAllClasses	ClassModel	ClassModel
getAllClasses	ClassModel::classifier	Classifier
getPersistentClass	Class	Class
getPersistentClass	Class::is_persistent	Boolean
getPersistentClass	Class::parent	Class

Features are unbounded!

Unbounded to Bounded: Partitioning

Metamodel feature	Partitions
Attribute::is_primary	true, false
Attribute::name	””, x | x!=””
Attribute::#type	1
Classifier::name	””, x | x!=””
Class::is_persistent	true, false
Class::#parent	0, 1
Class::#attrs	1, x | x>1
Association::name	””, x | x!=””
Association::#dest	1
Association::#src	1
ClassModel::#association	0, 1, x | x>1
ClassModel::#classifier	0, 1, x | x>1

Model Fragments

Footprint for getAllClasses:

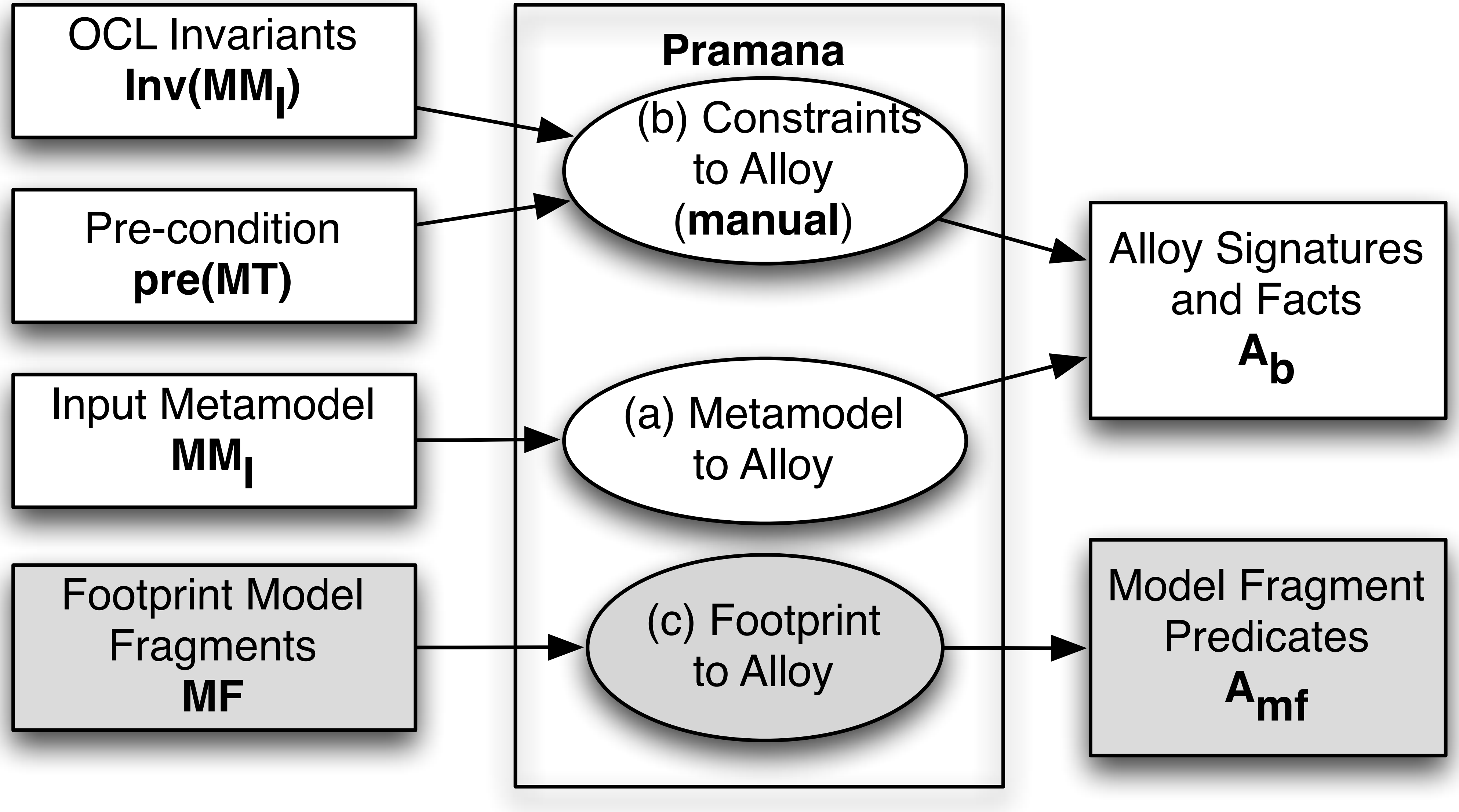
getAllClasses	Classifier	Classifier
getAllClasses	ClassModel	ClassModel
getAllClasses	ClassModel::classifier	Classifier

Model-Fragment	Description
MFgetAllClasses1	a Classifier & a ClassModel cm $\mid$ #cm.classifier = 0
MFgetAllClasses2	a Classifier & a ClassModel cm $\mid$ #cm.classifier = 1
MFgetAllClasses3	a Classifier & a ClassModel cm $\mid$ #cm.classifier $>$ 1
MFgetPersistentClass1	a Class c$\mid$c.is_persistent=True & a Class c2$\mid$#c2.parent=0
MFgetPersistentClass2	a Class c$\mid$c.is_persistent=True & a Class c2$\mid$#c2.parent=1
MFgetPersistentClass3	a Class c$\mid$c.is_persistent=False & a Class c2$\mid$#c2.parent=0
MFgetPersistentClass4	a Class c$\mid$c.is_persistent=False & a Class c2$\mid$#c2.parent=1

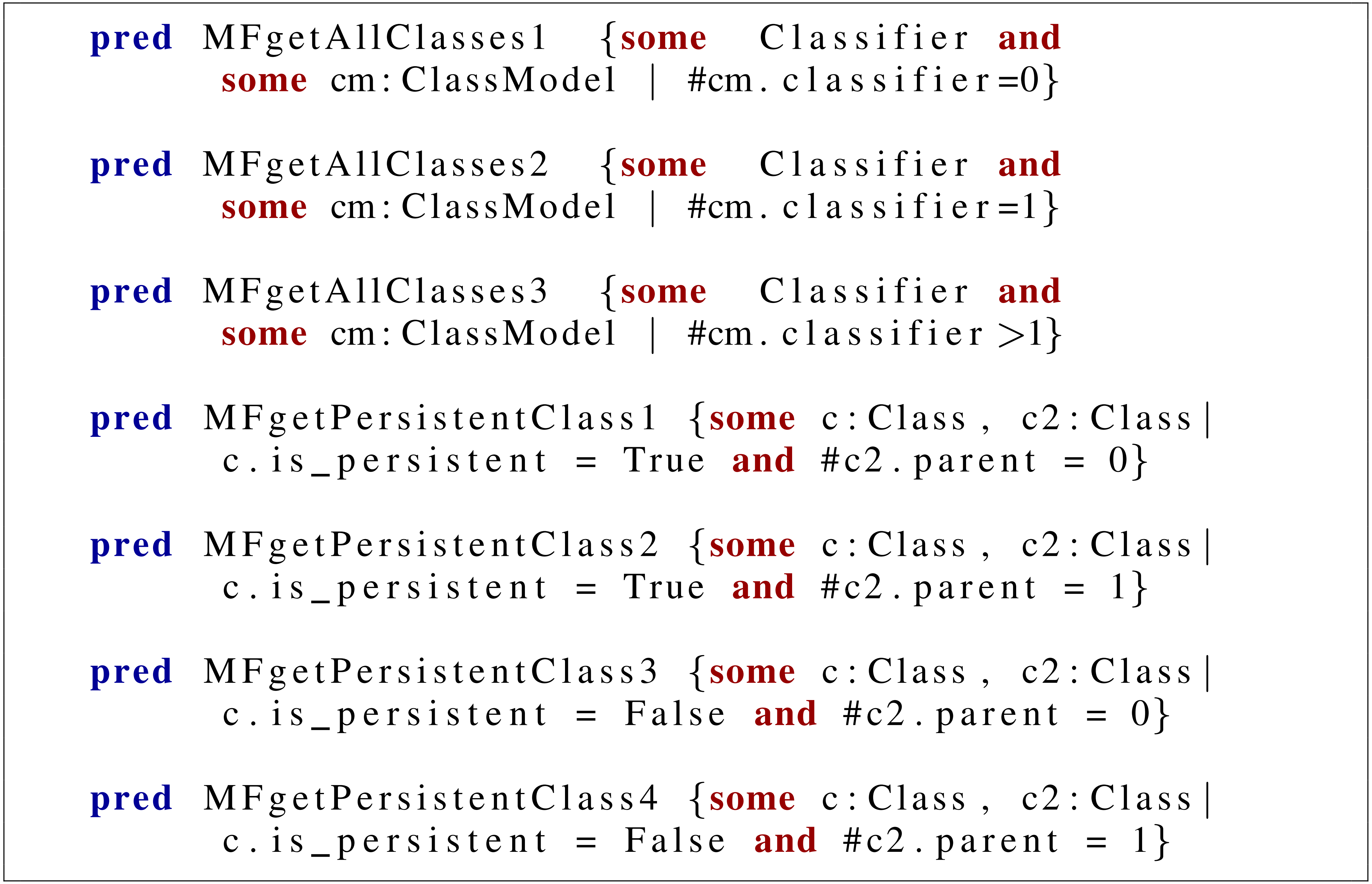
Model fragments of are **combinations of partitions** on **footprints**

Eg. 3 model fragments for partitions on types used in getAllClasses operation

Part 2: Transformation to Alloy

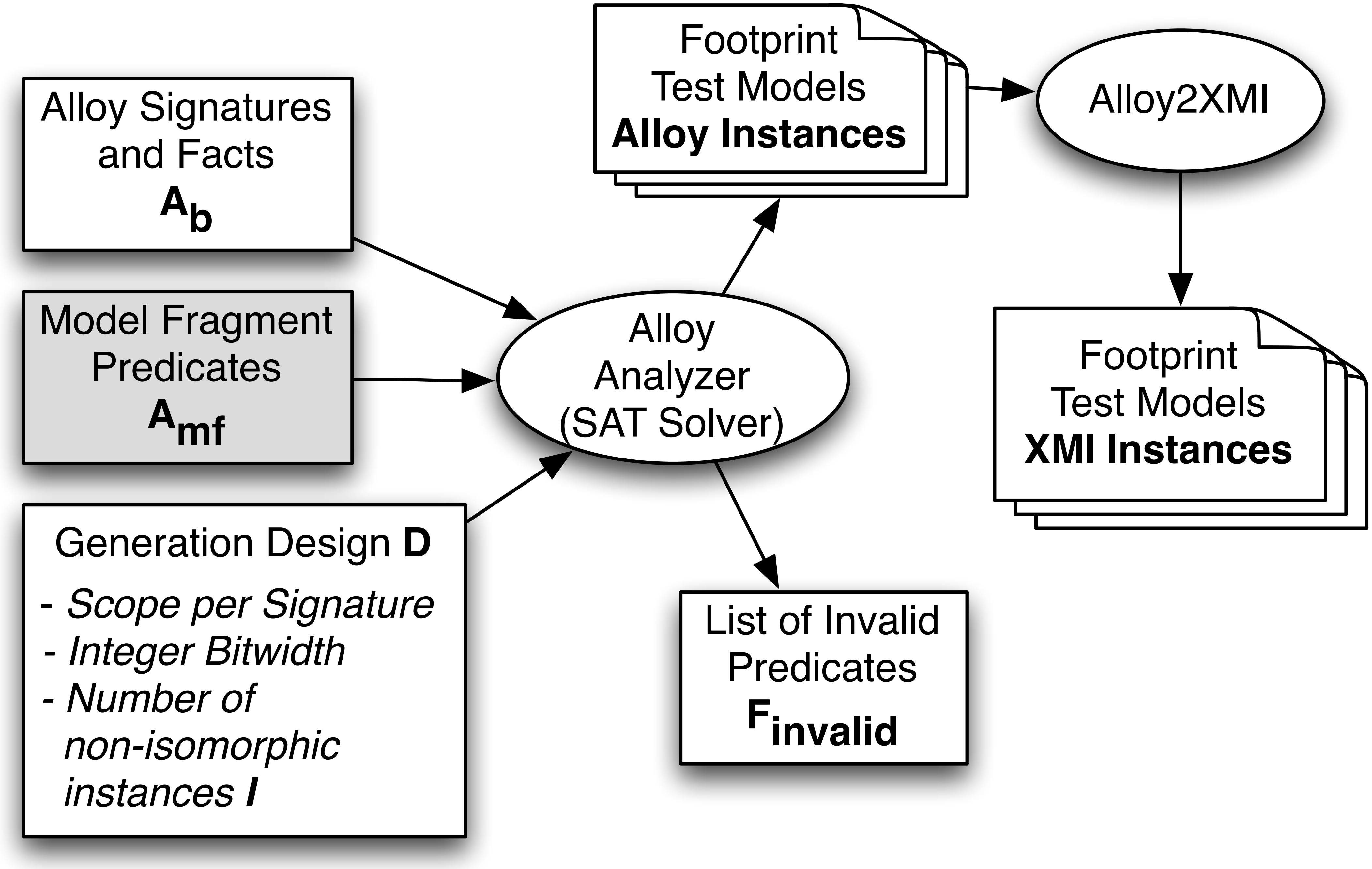


Model Fragment to Alloy



```
pred MFgetAllClasses1  {some  Classifier and
      some cm:ClassModel | #cm.classifier=0}

pred MFgetAllClasses2  {some  Classifier and
      some cm:ClassModel | #cm.classifier=1}

pred MFgetAllClasses3  {some  Classifier and
      some cm:ClassModel | #cm.classifier >1}

pred MFgetPersistentClass1 {some c:Class, c2:Class|
      c.is_persistent = True and #c2.parent = 0}

pred MFgetPersistentClass2 {some c:Class, c2:Class|
      c.is_persistent = True and #c2.parent = 1}

pred MFgetPersistentClass3 {some c:Class, c2:Class|
      c.is_persistent = False and #c2.parent = 0}

pred MFgetPersistentClass4 {some c:Class, c2:Class|
      c.is_persistent = False and #c2.parent = 1}
```

Part 3: Generating Test Models



Alloy Signatures and Facts **A_b**

Model Fragment Predicates **A_{mf}**

Generation Design **D**

- *Scope per Signature*
- *Integer Bitwidth*
- *Number of non-isomorphic instances **I***

Alloy Analyzer (SAT Solver)

Footprint Test Models **Alloy Instances**

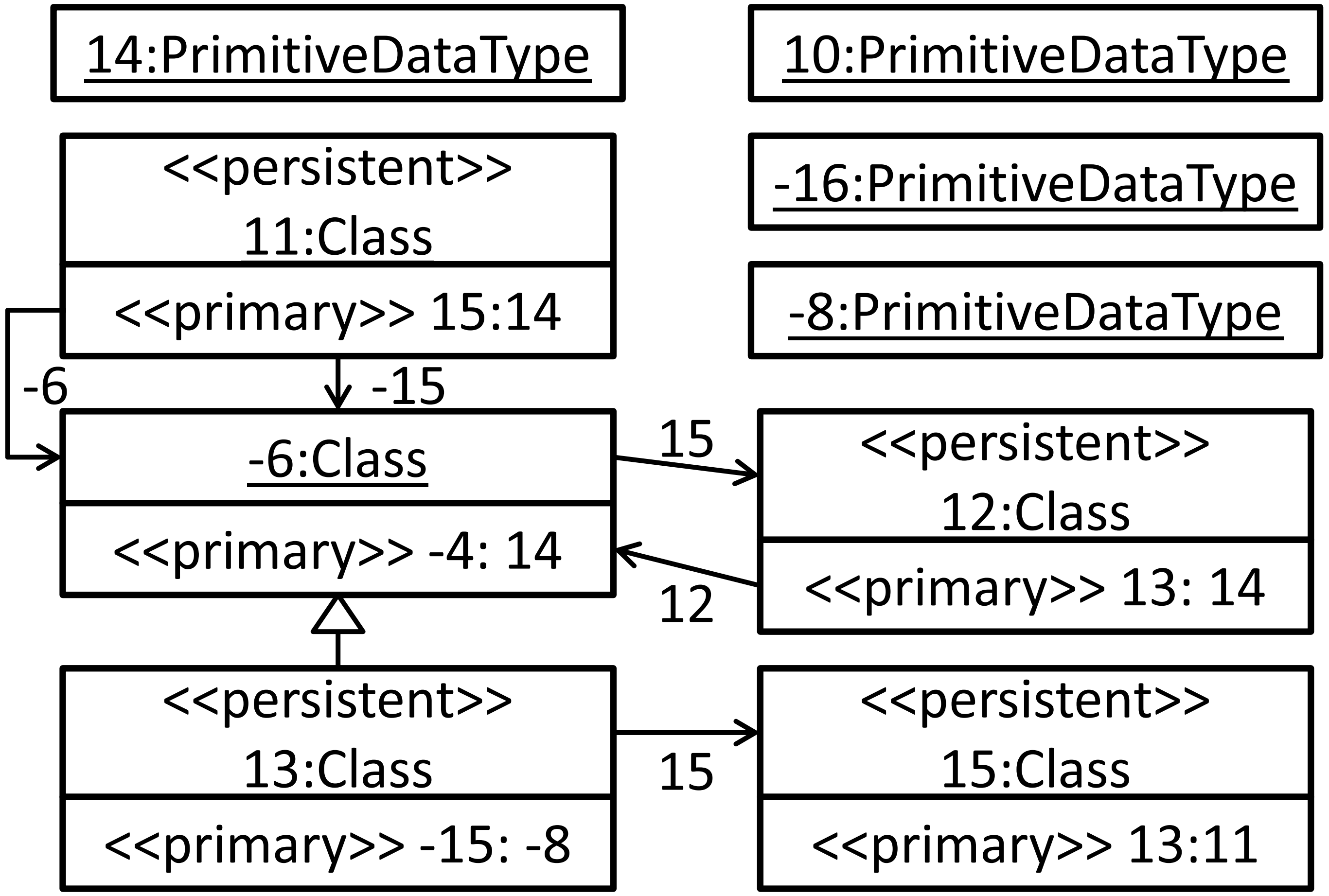
Alloy2XMI

Footprint Test Models **XMI Instances**

List of Invalid Predicates **$F_{invalid}$**

23 Consistent Fragments out of 72 Fragments

Example Test Model



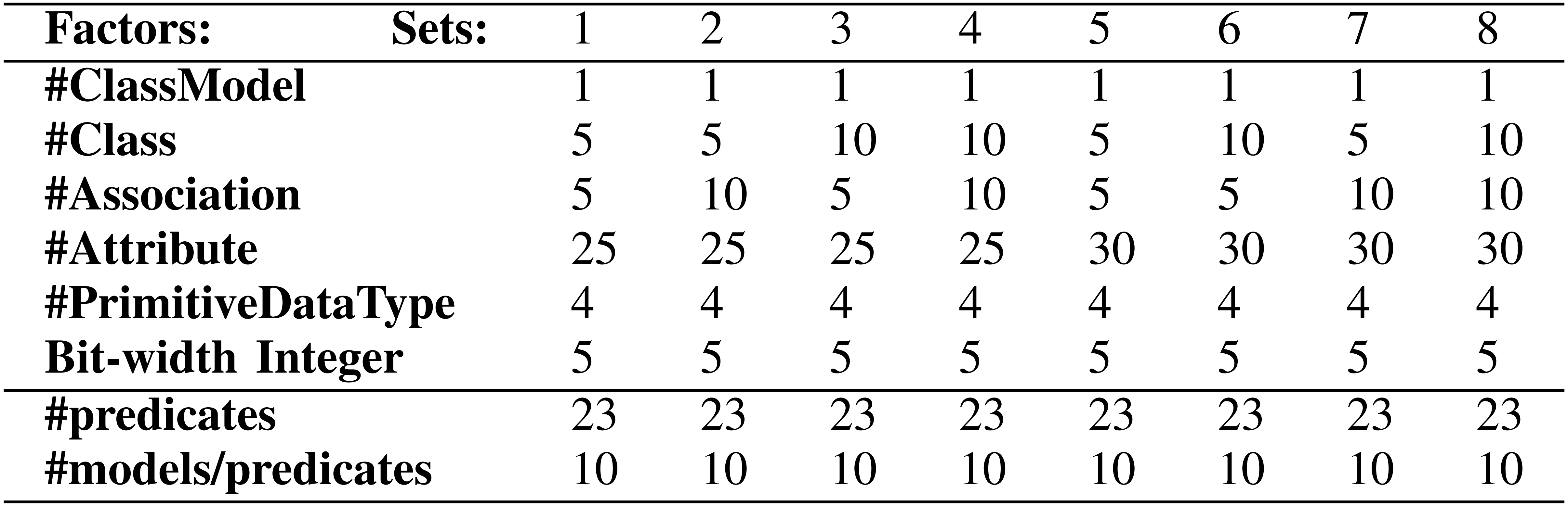
```
run MFgetPersisentClass2 for 1 ClassModel,5 int,exactly 10
  Class,exactly 5 Attribute,exactly 4 PrimitiveDataType,
  exactly 10 Association
```

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis
- Conclusion

Experimental Setup (1)

Input Test Models



Factors:	Sets:	1	2	3	4	5	6	7	8
#ClassModel		1	1	1	1	1	1	1	1
#Class		5	5	10	10	5	10	5	10
#Association		5	10	5	10	5	5	10	10
#Attribute		25	25	25	25	30	30	30	30
#PrimitiveDataType		4	4	4	4	4	4	4	4
Bit-width Integer		5	5	5	5	5	5	5	5
#predicates		23	23	23	23	23	23	23	23
#models/predicates		10	10	10	10	10	10	10	10

8 x 23 (consistent fragments) x 10 (non-isomorphic models) = 1840 test models

Experimental Setup (2)

Mutation Analysis to Qualify Test Models

1. We inject faults into **class2rdbms** using mutation operators

2. We create **200 mutant versions** (6 equivalent mutants) of class2rdbms with one fault each

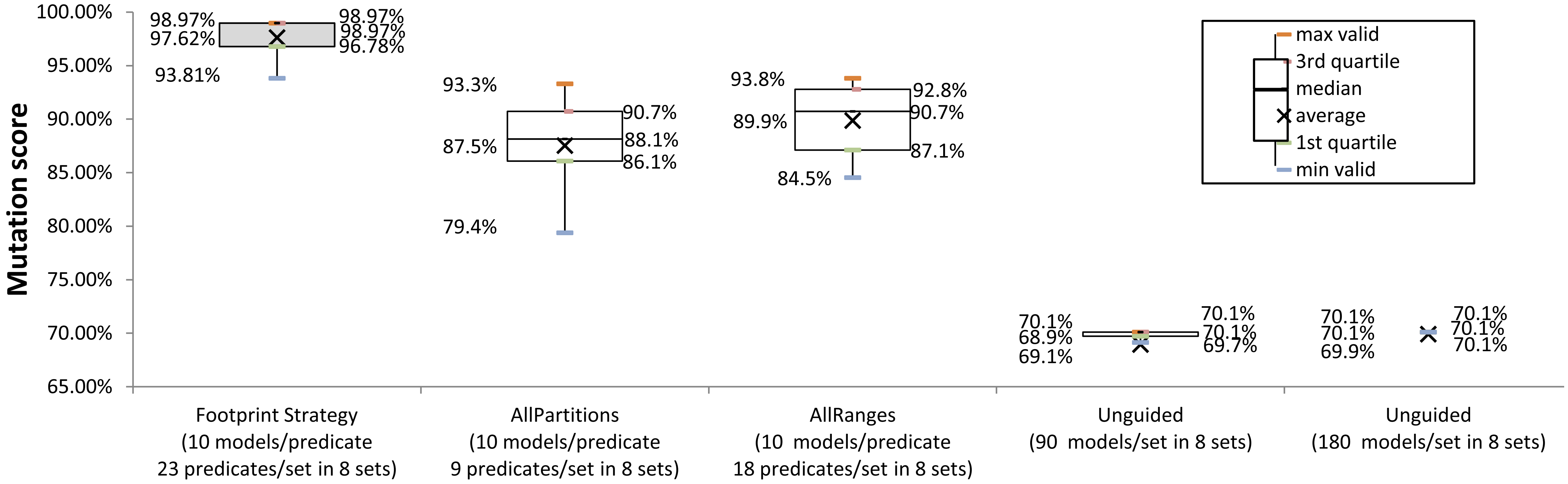
3. Mutant operators are expressed on **filtering, navigation, and creation** operations (Mottu et. al. ECMDA'06)

4. Each test model (1840 of them) is executed for each of the 200 mutant versions of class2rdbms

5. An oracle compares the output of the mutant vs. the original **class2rdbms** transformation

6. **Mutation score** is the **percentage of faults** detected in 200 mutants

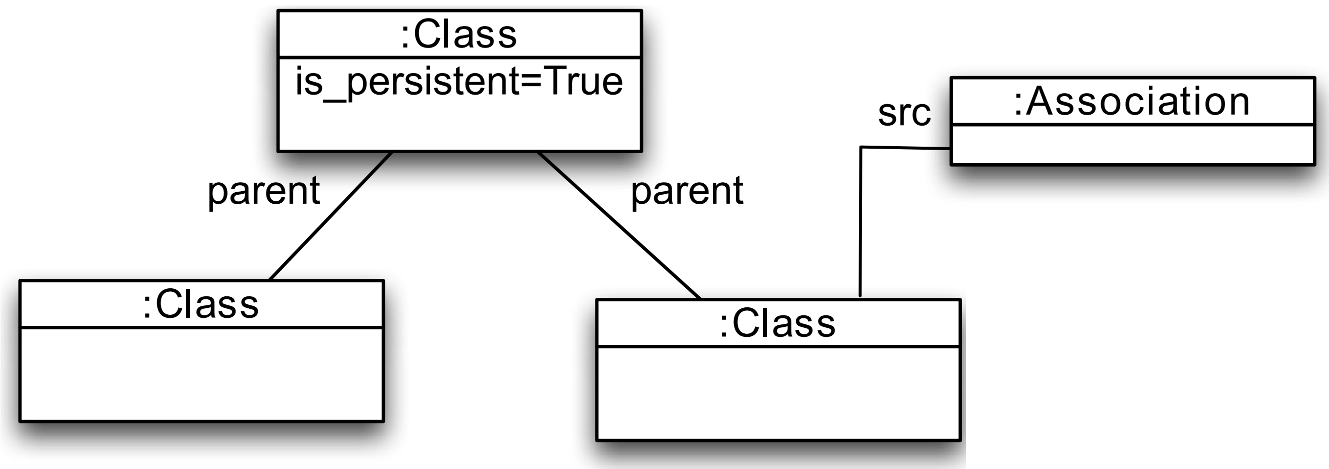
Mutation Analysis Results



Live Mutants?

1. Yes. **Two major** live mutants/faults still remained
2. Present in operations for **collecting classes and associations**
3. Example below: Fault selects only the first child of Class **cls**
4. Why **not killed** by static analysis? No bi-directional parent-child relationship between classes

```
getAllClasses(model).select{ c | c.parent == cls }
          .subSequence(0,0) //Injected fault
```



Human-made partial model to kill the live mutant

Outline

- Introduction: Model Transformation Testing
- Case Study: Class2RDBMS
- Problem: Tediousness of Creating Test Models
- The Story So far!
- Approach: Static Analysis for Transformation Testing
- Effective? Experiments based on Mutation Analysis
- Conclusion

Conclusion

- We present a **semi-automatic methodology** based on **static analysis** of a model transformation for automatic test model generation
- Static analysis **out performs** input domain partitioning (98.3% vs. 93%)
- **Small-model hypothesis** verified! They can uncover most of the faults

Future of automated model transformation testing?

- **Automation for transforming OCL invariants?** Specifying a new **Testable OCL** that ensures a bi-directional transformation to/from Alloy
- Improving **scalability of model loading/saving and operations** of them is important to the future of MDE and hence testing transformations.
- Maturity of **model synthesis and static analysis** for testing transformation languages will be consequence of the above.

Thank you. Pleased to Address Your Questions.