

Model Typing

Prof. Jean-Marc Jézéquel

jezequel@irisa.fr

<http://www.irisa.fr/triskell>

Intro & Motivation

- MDE: models, metamodels, transformations, code generators, ...
- Coming issues:
 - How do we assemble these pieces into systems/architectures?
 - How do we ensure insure these systems against changes to their parts?
- Reuse and maintainability

Intro & Motivation (2)

- Comparison
 - For each individual model/transformation, we talk and write about model elements (objects)
 - For the system, we talk about models - graphs of objects
- Reuse and maintainability:
 - Understood problems for objects
 - What do we do for models?

Intro & Motivation (3)

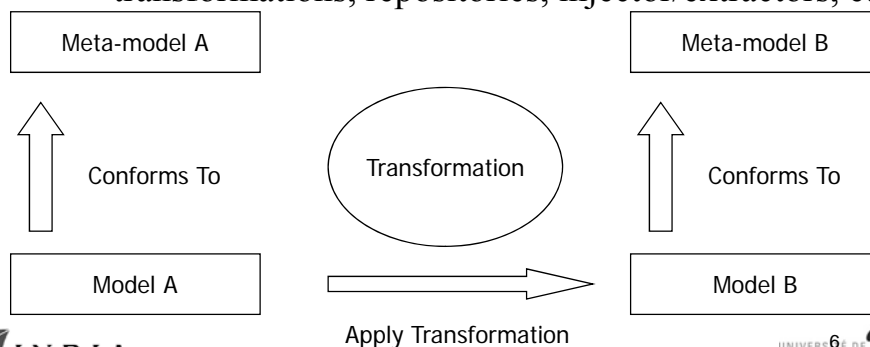
- What are the current techniques when using objects?
- Object-orientation: encapsulation, “inheritance”, *polymorphism*
- Which of these do we have for models?

Types, type systems

- Type: “Suitability for a purpose”
 - Set of values upon which operation/s can be performed successfully
 - E.g. in Java, a class or datatype
- Type System: rules to guarantee type safety
- Polymorphism: when can one type be substituted for another?

Types in MDE?

- Suitability at a coarse-grained level:
 - Types of Input and Output models ?
 - transformations, repositories, injector/extractors, etc



Structures in MDE

- Objects (model elements)
 - With a link to a metaclass (type)
- Models
 - MOF 1.x: “instances” of metamodel
 - MOF 2.0: heterogeneous extents
 - More generally: graphs of objects connected by typed links

Typing objects

- So how do we type our objects?
- Many type systems for objects
 - Cardelli: A Theory of Objects
- Extensions for:
 - 1st-class relationships/associations:
Bierman & Wren, FOOL 2005
 - MOF peculiarities
- Basically, classes

Class \neq type?

- However, class \neq type (LaLonde & Pugh, JOOP '91)
- Actually, in MOF, it kinda does
 - MOF is structural, not behavioural
- Classes also have more detail, but they're close enough

Typing models

- So, what is the type of a graph of interconnected objects?
- Answer: The set of types for all the objects in the graph
- + the set of types for the links, i.e. associations
 - Depends how you model associations

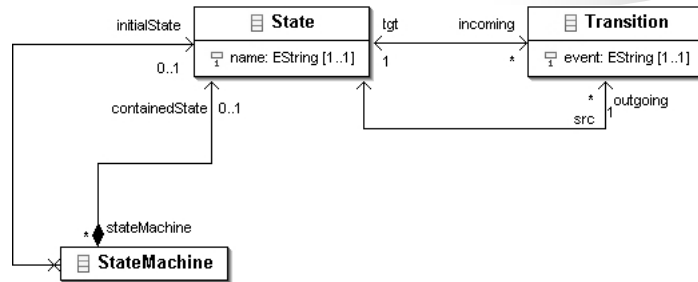
Typing models

- Hang on, a set of classes? Sounds like a package...
- Packages contain their classes
- Won't work with heterogeneous models
- We want to be able to specify the minimal types needed

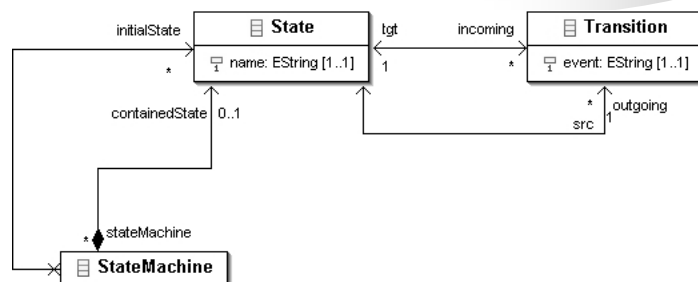
Problem

- If a program (i.e. model transformation) is written to work with one Meta-Model, what rule determines whether it would also work for a variant of this meta-model?
 - Meta-model evolutions (e.g. UML1.3, 1.4, 2.0...)
 - Compatibility among tools (XMI: just syntax)
- Model polymorphism/
model type substitutability
- Example with state machine variants

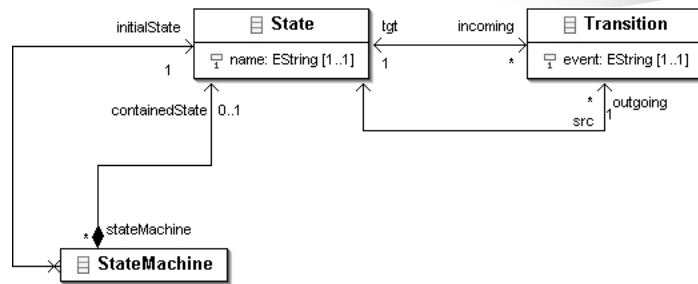
Simple State Machines



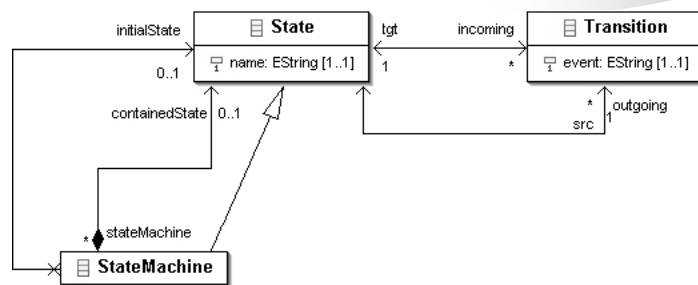
Multiple Start States



Mandatory Start State



Composite States



Object Type Conformance

- Many varieties
- Java-like
 - Explicit, subclass-based subtyping
 - As specified by OCL/UML/MOF
- Structural conformance
 - Does the type have all the required features?
 - Covariance, contravariance, etc
 - Efficiency considerations

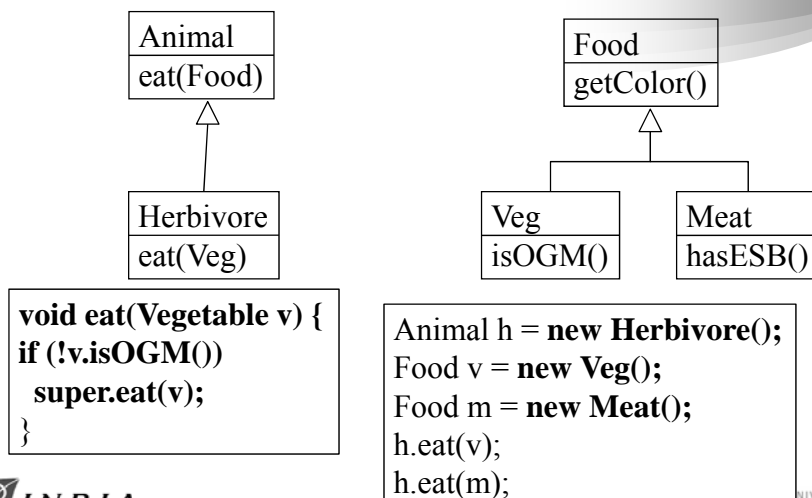
Object Type Conformance

- Some differences for objects in MOF:
 - Multiplicities on properties
 - Properties can be combined to form associations:
makes checking cyclical
 - Need to check whether properties are reflexive or not

Model Type Conformance

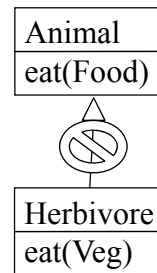
- For each *required* object type, there must be a conformant *provided* object type
 - Additional *provided* types are OK
 - As a general principle, providing too much is OK
- The problem is in relations among object types
 - Related to the covariance problem in OO languages
 - Animals eat food, herbivores eat vegetables (*pair wise subtypes*)
 - Need concepts such as *type groups/virtual types* (as in *Scala / Java5*)
- We have developed a type system for KerMeta to statically decide model substitutability
 - Formalization based on Bruce's works (Jim Steel PhD thesis)

Covariance problem in OO

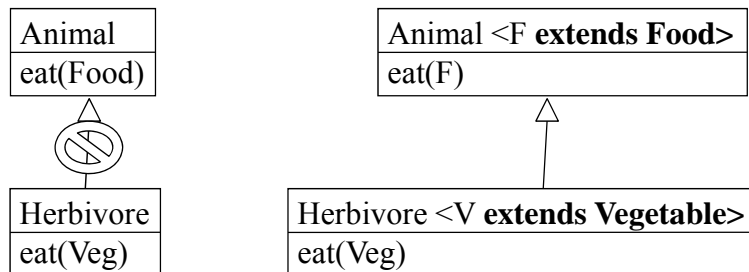


Result in Java

- Exception in thread "main"
java.lang.ClassCastException: Meat cannot be cast to Vegetable
- at Herbivore.eat(Herbivore.java:1)
- at Main.main(Main.java:8)



Pair wise subtyping



Results on the examples

^Conforms to ->	Simple	n-St	Mand-St	Comp	Final
Simple	✓	✗	✗	✗	✗
Multi-Start	✗	✓	✗	✗	✗
Mand-Start	✓	✗	✓	✗	✗
Composite	✓	✗	✗	✓	✗
Final States	✓	✗	✗	✗	✓

In Practice: KerMeta

- First-class support for model-types
 - Parameters & return types for operations
 - Variables
 - Other places?
- Fairly lightweight extension

In Practice: “megamodels”

- Describing architectures of models, metamodels, transformations and model programs, injectors/extractors
- Combination of model types and “representations”
- Sophisticated “Makefile” for MDE

Conclusion

- Reuse is (about to) become a problem at the MDE system level
- Model Types
 - Build on existing work in object type systems
 - Simple definition of a model type
 - Simple rule for substitutability