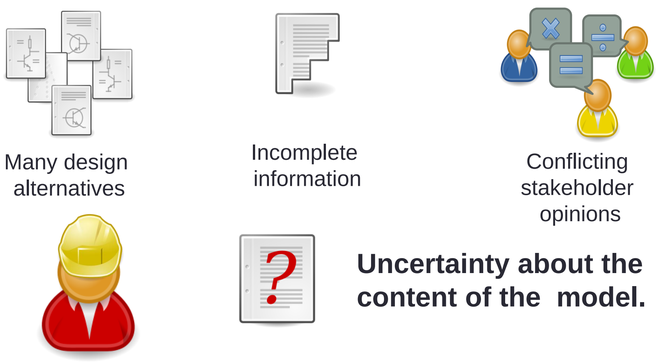
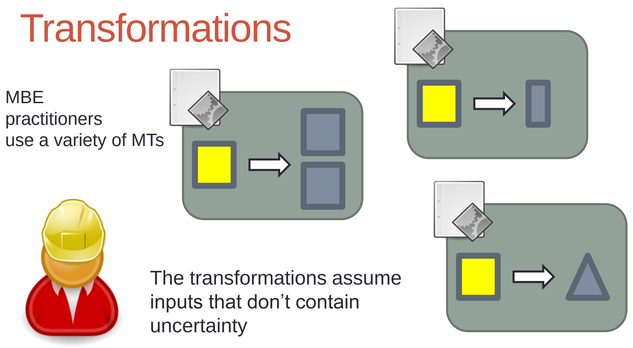
Transformation of Models Containing Uncertainty

Michalis Famelis, Rick Salay, Alessio Di Sandro, Marsha Chechik
{famelis,rsalay,adisandro,chechik}@cs.toronto.edu
University of Toronto

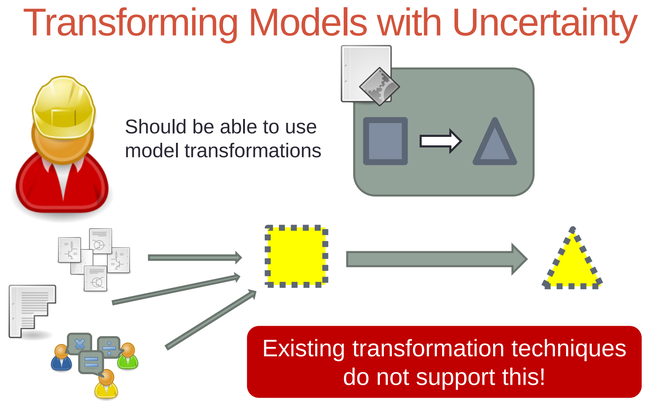
Uncertainty in software development



Transformations



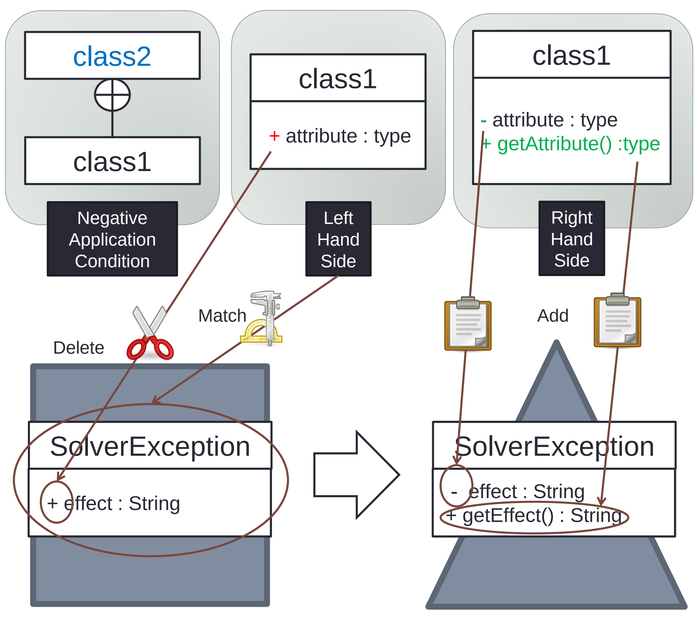
Transforming Models with Uncertainty



To apply MTs, we are forced to **artificially** remove uncertainty. Instead: We need to **lift** transformations so that they can apply to models with uncertainty

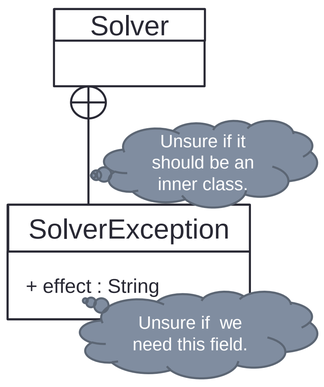
Model Transformations With Graph Rewriting

Example rule: ***EncapsulateVariable*** refactoring: Make fields private and add getter methods *unless* they belong to some inner class



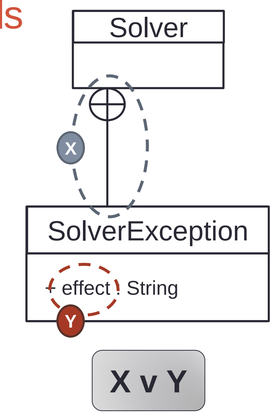
How can you match, add and delete if some elements are *Maybe*?

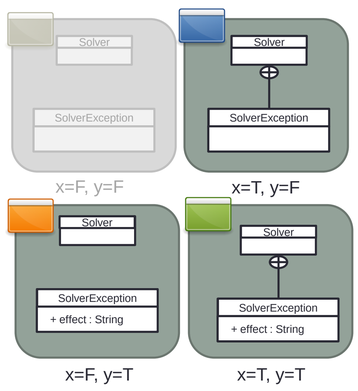
Representing Uncertainty with Partial Models



- Points of uncertainty (*"May elements"*) explicated using **syntactic annotations** Propositional variables: "the element exists"
- Restrictions to the set of concretizations can be captured in the "*May formula*"

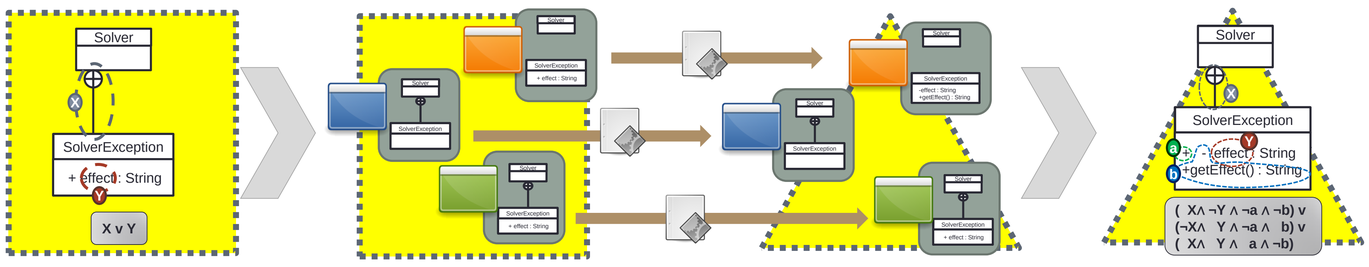
[ICSE'12]





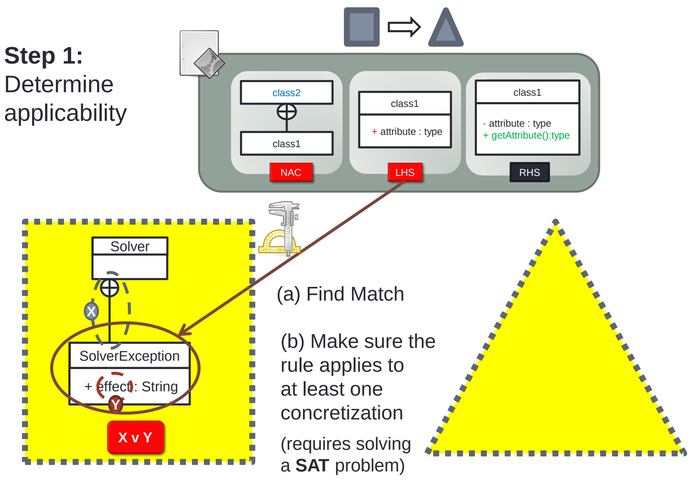
Concretizations: ways to resolve uncertainty.

Intuition (And definition of correctness)



Technique

Step 1: Determine applicability

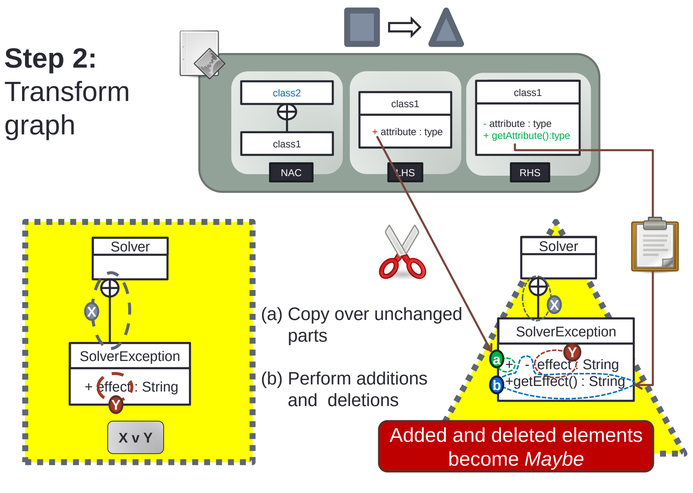


(a) Find Match

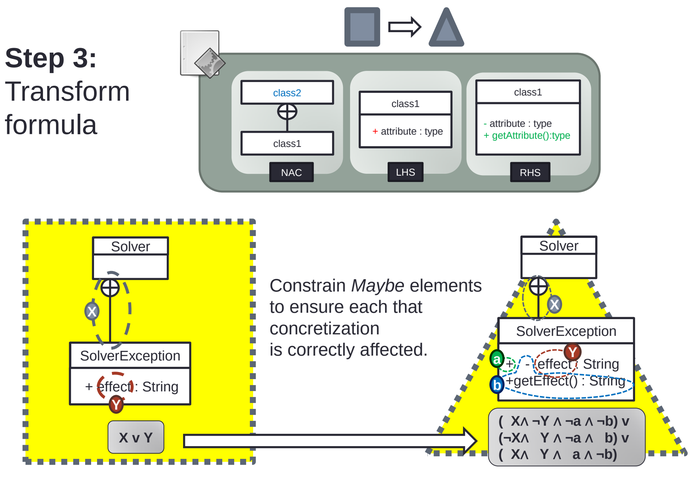
(b) Make sure the rule applies to at least one concretization

(requires solving a **SAT** problem)

Step 2: Transform graph



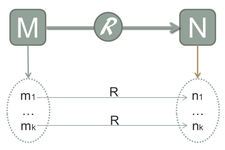
Step 3: Transform formula



Analysis

In the paper:

- Proof of correctness
  - The lifting algorithm implements our intuition



- Proofs of preservation of properties:
  1. Confluence
     The result of applying a set of rules to a model is the same regardless of the order of application or the order of matching sites.
  2. Termination
     Repeated applications will reach a point where the rule will no longer be applicable.

Tool Support

- Reuse partial model implementation in MMTF (Eclipse / EMF)
- Algorithm implementation
  1. *Determine rule applicability*
     - Henshin and the Z3 SMT solver
  2. *Transform the graph*
     - Henshin
  3. *Transform the formula*
     - Java (Z3 input strings)

Henshin emf

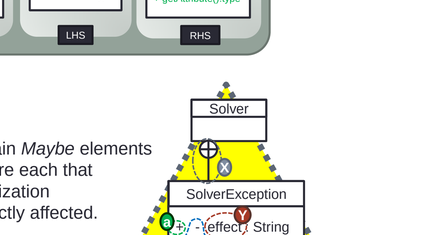
Z3

MMTF

Case Study

- Object–relational mapping (ORM)
  - *"Translate a class diagram to a relational database schema."*
- Triple graph grammar with 5 layered graph rules [Varro06]
- Input model: the Ecore metamodel
  - ORM for Ecore is important: cf. CDO and Teneo
- Manually injected points of uncertainty to create partia models with increasing numbers of concretizations
- **RQ**: *How does lifting scale with increasing uncertainty?*
- **Varied:** number of concretizations of input
- **Measured:** time to complete the ORM transformation
- Ran on Intel Core i7-2600 3.40GHz×4core, 8GB RAM, Ubuntu-64 12.10.

(Image from [Varro06])



# concretizations	1	24	48	108	144	192	256
Time (seconds)	32.6	32.8	32.7	32.9	32.6	33.0	48.4

- Runtime does not increase dramatically. Approach scales.

Next Steps

- Implement lifted semantics as a higher-order transformation (HOT)
  - Given a graph rewrite rule, produce a grammar that implements the lifted semantics
- Expand lifting for other types of model uncertainty, based on the rich MAVO framework [FASE12]

References

[ICSE12] M. Famelis, M. Chechik, and R. Salay. "Partial Models: Towards Modeling and Reasoning with Uncertainty". In Proc. of ICSE'12, 2012.

[Varro06] D. Varro, S. Varro-Gyapay, H. Ehrig, U. Prange, and G. Taentzer. "Termination Analysis of Model Transformations by Petri Nets". In Proc. of ICGT'06, pages 260–274, 2006.

[FASE12] R. Salay, M. Famelis, and M. Chechik. "Language Independent Refinement using Partial Modeling". In Proc. of FASE'12, 2012.