Model Transformations for the Verification of Model Transformations

Levi Lúcio[†], Hans Vangheluwe[†‡]

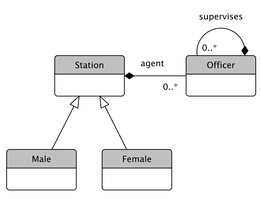
[†] McGill University, Canada
[‡]University of Antwerp, Belgium

Introduction

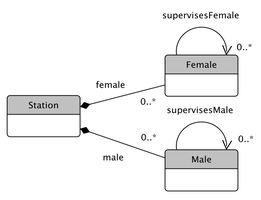
In this work we apply the *program verification* concept of *symbolic execution* to the verification of model transformations written in the Turing-incomplete DSLTrans language. Current state of the art model transformation verification techniques rely on SMT solvers or theorem proving to deal with the complexity of model transformations [1, 2, 3]. Our symbolic execution construction algorithms leverage on DSLTrans' reduced expressiveness in order to cope with the classical state space explosion problem typical to symbolic execution. As a result our verification technique [4, 5] is simple, relies on an in-house model transformation symbolic execution engine, is fully axiomatized, can be mathematically proved and has the potential to scale to real size GM transformations.

DSLTrans Model Transformations and Properties

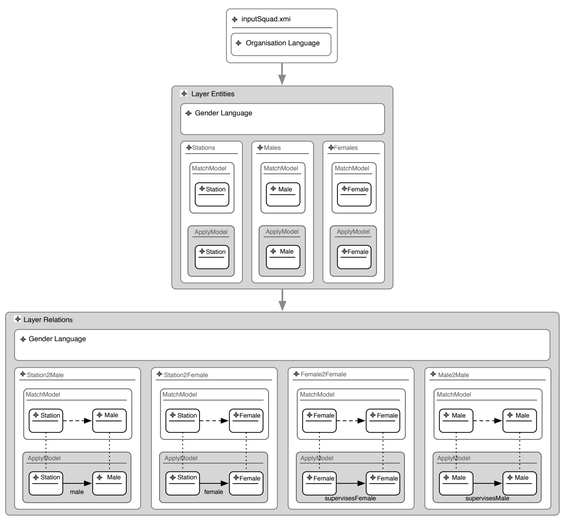
Police Station
Organisation Language

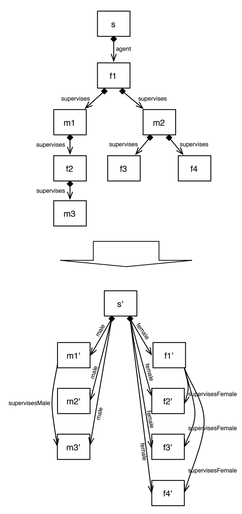


Police Station
Gender Language

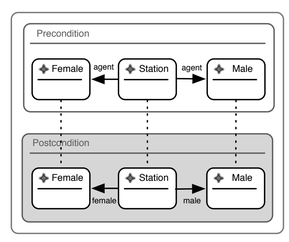


Organisation Language to Gender Language DSLTrans Transformation

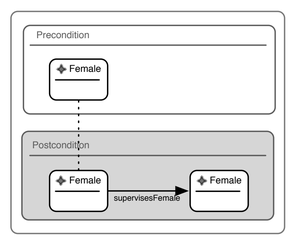




Properties to be Proved



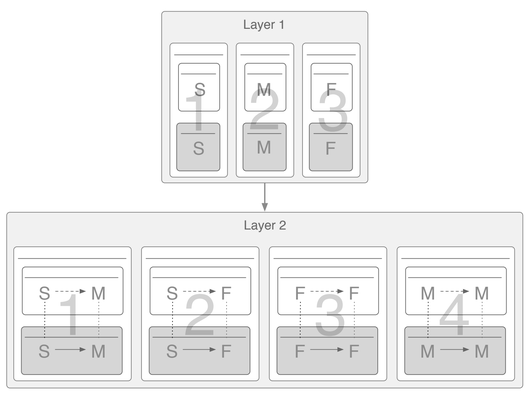
"a *model which includes a police station that has both a male and female chief officers will be transformed into a model where the male chief officer will exist in the male set and the female chief officer will exist in the female set* "



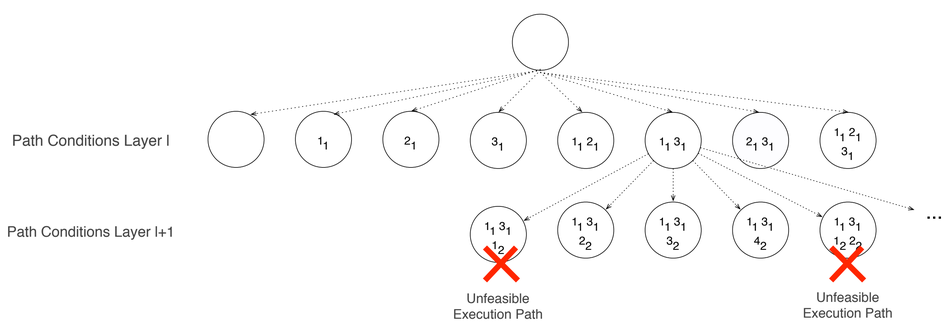
"any *model which includes female officer will be transformed into a model where that female officer will always supervise another female officer* "

Symbolic Execution Construction

DSLTrans Transformation

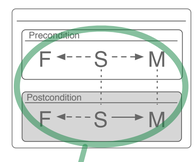


Symbolic Execution Construction



Property Proof

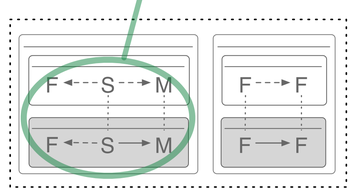
Property



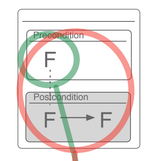
Precondition found
Postcondition found
Property holds!

Property Holds!

Path Condition



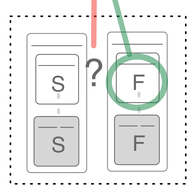
Property



Precondition found
Postcondition NOT found
Property does not hold!

Property does not Hold!

Path Condition



Experimental Results

# of rules	3	5	7	10	12	14
# of path conditions	8	14	31	269	337	1051
symbolic execution construction time (sec)	2.3×10^{-5}	0.12	0.25	0.27	0.40	0.93
used memory (Kb)	0.07	0.09	0.17	1.08	1.41	4.40
Satisfied Property (sec)	-	0.11	0.68	1.80	2.21	6.97
Unsatisfied Property (sec)	-	1.8×10^{-3}	1.8×10^{-3}	1.5×10^{-3}	1.7×10^{-3}	1.6×10^{-3}

# of rules	17	19	21	24	26	28
# of path conditions	9122	11428	35641	309341	387541	1208641
symbolic execution construction time (sec)	4.56	9.88	53.27	1222.11	3144.98	30513.64
used memory (Kb)	38.01	48.16	139.35	1307.66	1655.05	4777.00
Satisfied Property (sec)	66.42	88.57	320.00	-	-	-
Unsatisfied Property (sec)	1.5×10^{-3}	1.6×10^{-3}	1.6×10^{-3}	-	-	-

All experiments ran on a 2.2 GHz Intel Core i7 machine with 8GB of DDR3 memory running Ubuntu 11.10 and Python 2.7.

Implementation

- T-Core is used to to handle all the model manipulation primitives in the Symbolic Execution construction and property proof;
- Implemented as a mix of Python and T-Core;
- The whole prototype was built using the MDD principles, i.e. *Model Transformations are used to verify Model Transformations*.
- Optimizations:
  - *memoization* was used whenever possible to avoid isomorphic graph matching and rewrite operations (space + time complexity);
  - pointers to rules instead of copies of rules to build each path condition (space complexity);
  - For property proof we avoid checking path conditions where the property is sure to hold (time complexity).

Conclusion and Contributions

- We have applied the concept of symbolic executions of (DSLTrans) Model Transformation and provided the necessary algorithms;
- We show our symbolic execution technique scales well in our experimental setting and has the potential to scale for real world problems;
- We demonstrate that expressiveness reduction of a model transformation language can be very beneficial to the design and construction of a model transformation verification tool;
- We demonstrate that model transformations can verify model transformations.

Bibliography

[1] M. Asztalos, L. Lengyel, and T. Levendovszky. Towards Automated, Formal Verification of Model Transformations. In ICST, pages 1524. IEEE, 2010.

[2] F. Büttner, M. Egea, J. Cabot, and M. Gogolla. Verification of ATL Transformations Using Transformation Models and Model Finders. In ICFEM, pages 198213. Springer, 2012.

[3] F. Büttner, M. Egea, and J. Cabot. On Verifying ATL Transformations Using 'off-the-shelf' SMT Solvers. In MoDELS, pages 432448. Springer, 2012.

[4] Symbolic Execution for the Verification of Model Transformations, Levi Lucio and Hans Vangheluwe. Technical report, SOCS-TR-2013.2, McGill University, 2013. http://msdl.cs.mcgill.ca/people/levi/30_publications/files/MTSymbExec.pdf

[5] Model Transformations to Verify Model Transformations, Levi Lúcio and Hans Vangheluwe. Proceedings of Verification of Model Transformations (VOLT) 2013, Budapest, Hungary.