How to Verify a Model Transformation does its Job?

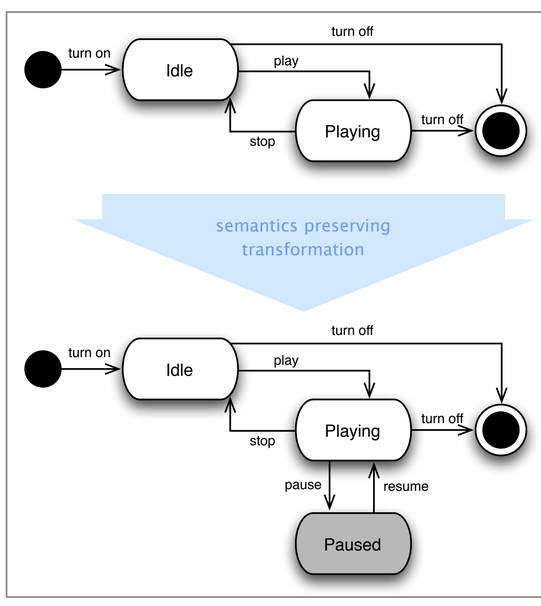
Levi Lúcio[†], Hans Vangheluwe[†] Eugene Syriani[†‡] and Maris Jukss[†]

[†]McGill University, Montréal, Canada [‡]University of Alabama, USA

Problem Statement

The NECSIS (Network on Engineering Complex Software Intensive Systems for Automotive Systems) initiative is a collaboration between several Canadian universities, General Motors of Canada, IBM and Malina software. It is aimed at researching Model Driven Engineering methodologies and tools for increasing productivity when developing automotive software. Our work concentrates on the study of the role of model transformations in this scenario, in particular on their correction. Model transformations are ubiquitous in software development, some examples are data format interchange tools, language translators, small compilers or small language interpreters. Most of the times in industrial settings these tools are defined in an implicit fashion. By treating model transformations as first-class citizens of the development toolchain and verifying they are correct, we aim at increasing the reusability of quality software and thus increasing productivity.

Verifying tranformations for language translators, small compilers (terminating transformations) [1, 2]

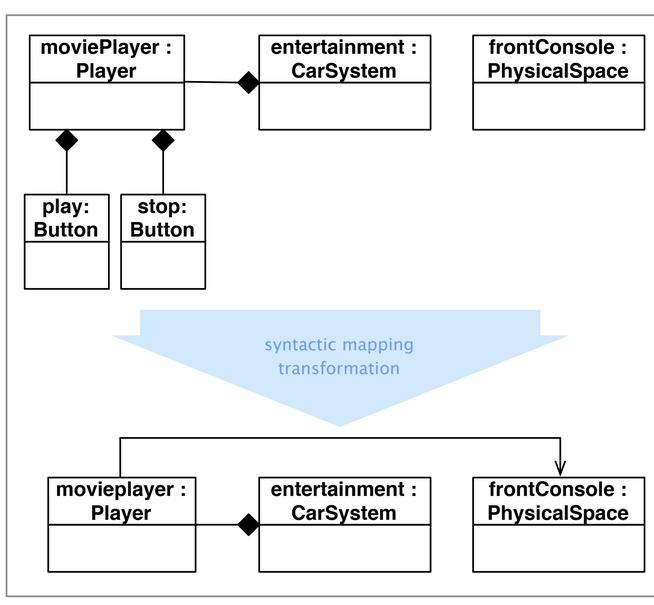


Property example:
" *the enhanced player will still reach the* ***play*** *state* "

Applicable when:
partial model structure can be preserved;

Languages:
same source and target languages;

Implementation:
by restricting transformations by construction, for one particular formalism;



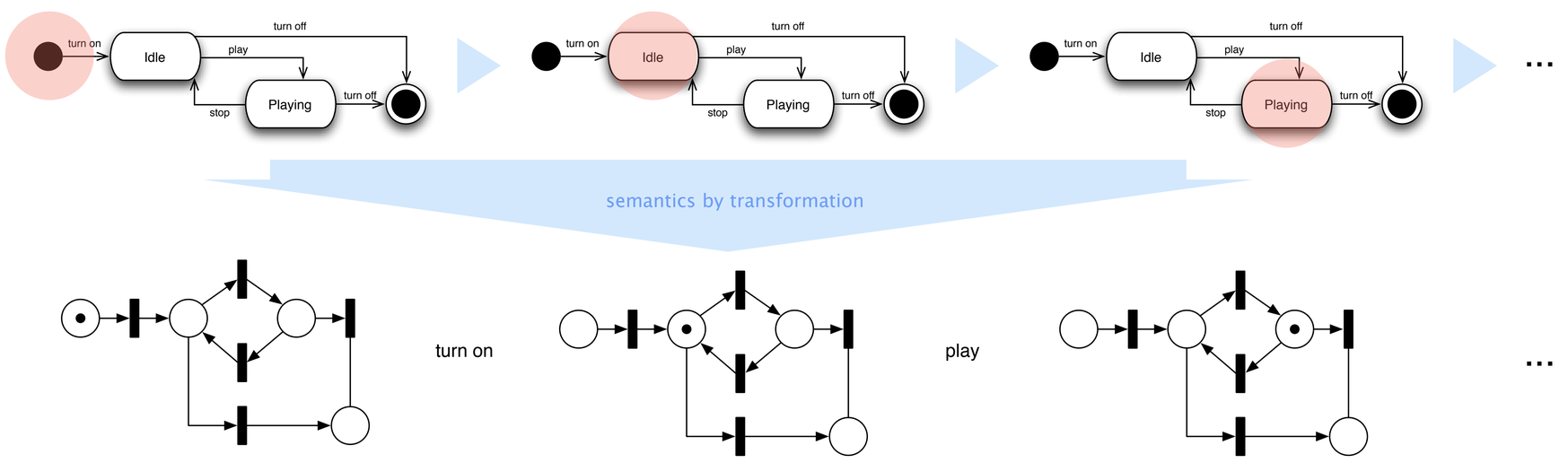
Property example:
" all movie players in the car will become attached to a console "

Applicable when:
transformations are too large or fragmented to be directly understood;

Languages:
same or different source and target languages;

Implementation:
by proving the transformation implements the required mapping between source and target language patterns.

Verifying tranformations for language interpreters (general case non-terminating transformations) [3]



Property example:
" the player will reach the play state "

Applicable when:
the semantics of the model are rule based and those rules are implemented by transformation;

Languages:
same source and target languages;

Implementation:
by transforming the transformation itself into an analyzable formalism.

Model transformations in automotive software

The power window case study

Identified domain specific languages

- **Environment Description Language**: for describing interactions between the power window and humans;
- **Plant Description Language**: for describing the hardware configuration of a power window;
- **Controller Description Language**: for describing the logical operation of the power window hardware components;
- **Deployment Platform Language**: for describing the whole infrastructure.

Identified transformations

- for **verification**:
  e.g. Control DL
  → Petri Nets
- for **simulation**:
  e.g. Plant DL → Causal Block Diagrams
- for **composition**:
  e.g. Env. DL + Ctrl. DL + Plant DL
  → Petri Nets
- for **deployment**:
  e.g. Control DL + Plant DL
  → Deployment DL

Conclusions & Future Work

The problem of verifying model transformations is not yet sufficiently well studied, or even well defined. We are studying the problem from two – ideally converging – directions: (1) what are the usable properties of model transformations and what techniques can be used to verify them; (2) what transformations and respective interesting properties emerge from industrial practice. We have started working on these two directions. Having identified transformations in the power window case study, we will now work on their properties' verification.

Bibliography

[1] J. Padberg, M. Gajewsky, and C. Ermel. *Refinement versus verification*, Technical report, Technische Universität Berlin, 1997.

[2] L. Lúcio, B. Barroca, V. Amaral. *A Technique for Automatic Validation of Model Transformations*, Proceedings of the MoDELS 2010 Conference, Springer, pp. 136-150.

[3] Juan de Lara, Hans Vangheluwe. *Automating the transformation-based analysis of visual languages*, Formal Asp. Comput. vol. 22(3-4) 2010, pp. 297-326.

http://msdl.cs.mcgill.ca/ levi@cs.mcgill.ca, hv@cs.mcgill.ca, esyriani@cs.ua.edu