TTC'16 Live Contest Case Study

Execution of dataflow-based model transformations

Antonio Garcia-Dominguez

Aston University, Birmingham, UK
a.garcia-dominguez@aston.ac.uk

Abstract.

1 Introduction

Model-to-model transformation frameworks and tools have evolved in the recent years to process larger models more efficiently. Traditional *batch* transformation engines took the entire source model(s) and produced the entire target model(s) exactly once. A small change would require going again through the entire model. Alternatively, an *incremental* transformation engine processes only the change itself, updating the target model(s) as needed.

Various approaches for incremental transformations are present in the literature: for instance, ReactiveATL [5] is an alternative execution engine of ATL which only computes and updates results on-demand as the model changes. VIATRA3 is a general purpose framework for reactive transformations [1], in which users write rules that trigger as the model changes. Beyond model transformations, IncQuery [2] is an incremental query language that builds a RETE network and feeds EMF change notifications to keep query matches up to date. Streaming transformations have been studied by Cuadrado and Lara [3] with the Eclectic tool and its ATL-like language, and by David, Rath, and Varró [4] through complex event processing.

Most of these systems can be seen as complex transformations of rule-based systems into event-driven systems, or frameworks that allow for writing these event-driven systems. This case suggests studying a type of notation that may be more directly amenable to incremental execution: a graph of model-oriented primitives which generate and process streams of tuples. The notation is inspired on popular Extract-Transform-Load (ETL) tools such as Pentaho Data Integration[1] or Talend Data Integration[2]. As data integration can be considered a form of model transformation, perhaps there might be

[1] `http://community.pentaho.com/projects/data-integration/`
[2] `https://www.talend.com/products/data-integration`

lessons to learn from these languages. In addition to their potentially simpler incrementality, breaking rules into smaller primitives might increase the level of detail of the execution traces of the transformation.

This case study presents a simplistic version of such a notation, with primitives subdivided into "minimal" and "extended" sets. Participants are tasked with writing an execution engine using their favorite approach (e.g. code generation or model interpretation) and tool, which may support batch and/or incremental transformations.

All the resources are included in the official Github repository[3]. These resources are mentioned in Section 2. The evaluation criteria for the provided solutions are described in Section 3. Contestants are invited to raise questions through GitHub issues should there be any unclear points in the description.

2 Case description

2.1 Abstract syntax

Explain the abstract syntax of the language in rough terms, and also the primitives. Should mention the expected properties of a model element (eContainer and eClass so far).

2.2 Concrete syntax

Explain the textual (Xtext-based) and graphical (Sirius-based) notations. Doesn't need to be too detailed, as contestants won't actually write transformations.

2.3 Example: Families to Persons

Explain the Families2Persons example as described through the notations.

2.4 Task 1: base primitives

families2persons could be implemented, then tree2graph could be the "extra" transformation for soundness.

2.5 Task 2: extended primitives

flowchart2html could be implemented, then class2rdb could be the "shadow" transformation for soundness.

3 Evaluation

Correctness (w/shadow transformation with base primitives revealed during conference), performance, extensibility.

[3] `https://github.com/TransformationToolContest/ttc16-live-contest`

References

[1] Gábor Bergmann, István Dávid, Ábel Hegedüs, Ákos Horváth, István Ráth, Zoltán Ujhelyi, and Dániel Varró. Viatra 3: A Reactive Model Transformation Platform. In Dimitris Kolovos and Manuel Wimmer, editors, *Theory and Practice of Model Transformations*, number 9152 in Lecture Notes in Computer Science, pages 101–110. Springer International Publishing, July 2015.

[2] Gábor Bergmann, Ákos Horváth, István Ráth, Dániel Varró, András Balogh, Zoltán Balogh, and András Ökrös. Incremental Evaluation of Model Queries over EMF Models. In Dorina C. Petriu, Nicolas Rouquette, and Øystein Haugen, editors, *Model Driven Engineering Languages and Systems*, number 6394 in Lecture Notes in Computer Science, pages 76–90. Springer Berlin Heidelberg, 2010.

[3] Jesús Sánchez Cuadrado and Juan de Lara. Streaming Model Transformations: Scenarios, Challenges and Initial Solutions. In Keith Duddy and Gerti Kappel, editors, *Theory and Practice of Model Transformations*, number 7909 in Lecture Notes in Computer Science, pages 1–16. Springer Berlin Heidelberg, June 2013. DOI: 10.1007/978-3-642-38883-5_1.

[4] István Dávid, István Ráth, and Dániel Varró. Streaming Model Transformations By Complex Event Processing. In Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfran, editors, *Model-Driven Engineering Languages and Systems*, number 8767 in Lecture Notes in Computer Science, pages 68–83. Springer International Publishing, September 2014.

[5] Massimo Tisi, Salvador Martínez, Frédéric Jouault, and Jordi Cabot. Lazy execution of model-to-model transformations. In *Model Driven Engineering Languages and Systems*, pages 32–46. Springer, 2011.