Truth Tables to Binary Decision Diagrams

Antonio García-Domínguez
Aston University
B4 7ET, Birmingham, United Kingdom
a.garcia-dominguez@aston.ac.uk

Georg Hinkel
Tecan Software Competence Center
55252, Mainz-Kastel, Germany
georg.hinkel@tecan.com

Abstract

Model transformation tools have reached a considerable level of maturity in the core features, and are currently developing in many directions. Some tools are focusing on providing higher performance for large models or complex transformations. Others focus on bidirectionality, visualisation, traceability, or verifiability, among other research directions. Whereas past cases in TTC have focused on specific research directions, this case study presents a well-known simple transformation and welcomes researchers to apply their research to it. The aim of this case is to serve as a showcase of the various directions that model transformation research is going towards at the moment.

1 Introduction

Past editions of the Transformation Tool Contest have focused on a variety of topics:

- In 2018, the Quality-based Software Selection and Hardware-Mapping case [4] discussed optimisation-oriented model transformations (with a combination of performance and solution quality). The Social Media Live Case [7] considered performance in updating model views as models changed (with a strong preference for approaches supporting incrementality).
- In 2017, the Smart Grid case focused on incrementality [6], the Families to Persons case discussed bidirectional transformations [1], State Elimination focused on performance and the live case on Transformation Reuse [5] discussed mechanisms to share complex logic across multiple versions of a transformation.
- In 2016, optimisation-oriented model transformations were discussed in considerable breadth through the Class Responsibility Assignment case [2], and an alternative dataflow-based notation for model transformation was evaluated in the live case study [3].

While these were notable examples of realistic transformations, they were narrowly focused on a specific topic, and their complexity discouraged some attendees from applying their own research agenda to the transformation.

This year, we proposed a broader contest that welcomed all active lines of work on model transformation. It was based on a simpler, well-known transformation from the ATL Zoo [8]: TT2BDD (Truth Tables to Binary Decision Diagrams). Striving for raw performance was an option, but the case welcomed approaches that focused on other attributes of interest of a high-quality model transformation: for example, verifiability, traceability, bidirectionality, or understandability. In general, this case was proposed as a showcase of the current variety of model transformation tools. All resources for this case are available on Github[1].

In: A. Garcia-Dominguez, G. Hinkel and F. Krikava (eds.): Proceedings of the 12th Transformation Tool Contest, Eindhoven, The Netherlands, 19-07-2019, published at http://ceur-ws.org.

[1] https://github.com/TransformationToolContest/ttc2019-tt2bdd

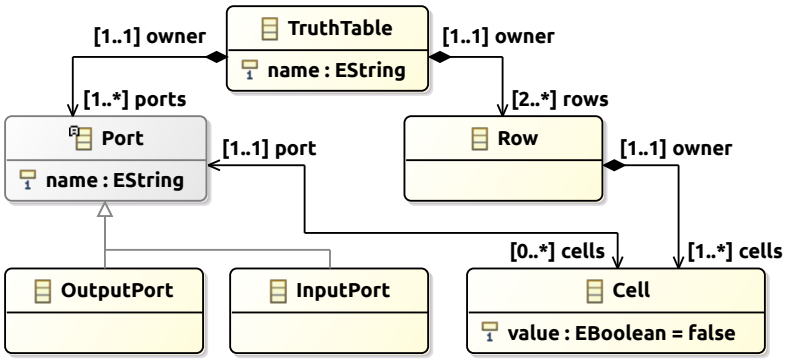

Figure 1: Class diagram for the input Truth Tables metamodel

A	B	C	D	S
0	0	-	-	0
0	1	0	0	1
0	1	0	0	1
0	1	0	1	0
0	1	1	-	0

A	B	C	D	S
1	0	0	0	0
1	0	1	0	1
1	-	-	1	0
1	1	0	0	1
1	1	1	0	0

Table 1: Example truth table: A to D are input ports and S is an output port. "-" means "ignored".

The rest of the document is structured as follows: Section 2 describes the TT2BDD transformation. Section 3 several tasks of interest that should be tackled in a solution (authors are free to propose their own tasks of interest). Section 4 mentions the benchmark framework for those solutions that focus on raw performance. Finally, Section 5 mentions an outline of the initial audience-based evaluation across all solutions, and the approach that will be followed to derive additional prizes depending on the attributes targeted by the solutions.

2 Transformation Description

This section introduces the Truth Tables to Binary Decision Diagram transformation, with a description of the input and output metamodels, and an outline of an implementation.

2.1 Input Metamodel: Truth Tables

The input metamodel is shown on Figure 1. The TruthTable class is as the root of the model, and contains a collection of Ports and Rows. Ports come in two types: InputPorts and OutputPorts. Rows contain sequences of Cells, which assign values to the InputPorts and OutputPorts of the table.

Automated EMF opposite references are used liberally in the metamodel to simplify the specification of the transformation. *owner* is used to access the container from several child objects (e.g. the TruthTable of a Row), and it is possible to see which *cells* referenced a specific Port.

A sample model is shown on Table 1. The truth table has four InputPorts named A to D, and one OutputPort named S. The first Row contains only 3 Cells, specifying that if A and B are 0, then S should be 0.

2.2 Output Metamodel: Binary Decision Trees

The original output metamodel from the ATL Zoo version is shown on Figure 2. The BDD class is the root of the model, and contains the root of the Tree and a collection of Ports. Similarly to the Truth Tables metamodel, there are InputPorts and OutputPorts.

Figure 3 shows the changes introduced into a revised version of the original metamodel, which allow the BDD to be a rooted acyclic graph and not just a tree. This would allow the BDD to reuse common subtrees, which would produce more compact circuits in some situations. The case provides both versions as separate metamodels: solution implementers were free to choose either version.

Tree is the common superclass for any node in the tree. Inner nodes check the value of an InputPort: if it is a false value, evaluation will proceed through the *treeForZero* Tree; otherwise, evaluation will go through

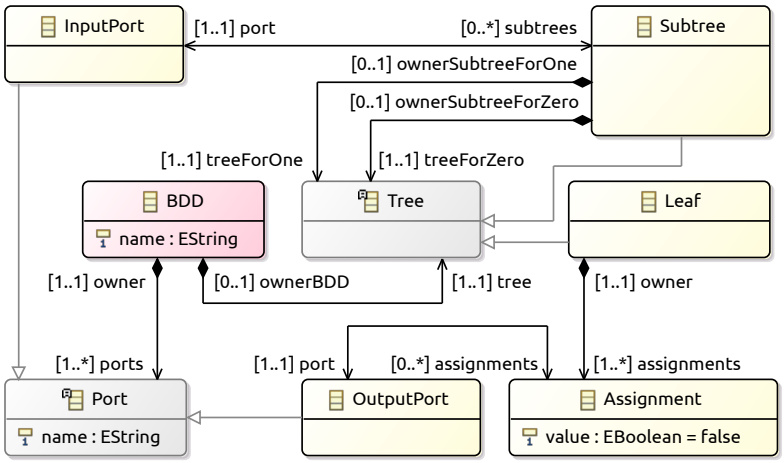

Figure 2: Class diagram for the output Binary Decision Diagram metamodel, in the original tree-based version.

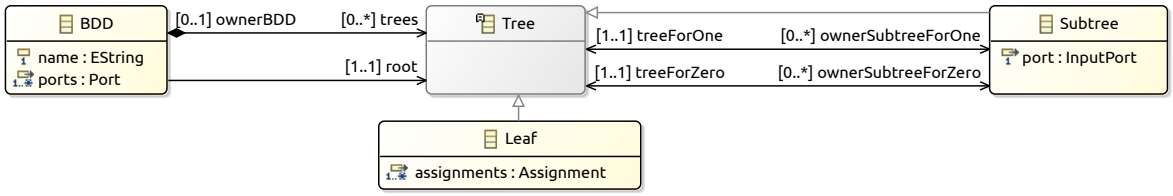

Figure 3: Changed classes in the graph-based variant of the Binary Decision Diagram metamodel (2019-05-26). Omitted clases remain the same.

the *treeForOne* TREE. Leaf nodes are LEAF objects, which provide an ASSIGNMENT of a boolean value to each of the available OUTPUTPORTS.

The equivalent BDD for the truth table on Table 1 is shown on Figure 4. SUBTREEs are represented by the circle referencing an INPUTPORT and their subtrees for when the port takes a 0 or 1 value. ASSIGNMENTS are represented by the highlighted nodes that provide values to the OUTPUTPORTS.

2.3 Process Outline

The transformation essentially needs to construct a (preferably minimal) binary decision diagram that produces the same values for the output ports, given the same values in the input ports. Given the high interest about this problem in circuit design, many algorithms have been proposed in the literature.

Solution authors are welcome to implement a more optimal algorithm in their transformation tool. This section will only outline a simple approach that can be readily implemented without too much complication.

There are some basic mappings which are immediately obvious:

- Each TRUTHTABLE object should correspond to a BDD object, with the same name and equivalent PORTS.
- Each INPUTPORT and OUTPUTPORT should be mapped to an object of the BDD type with the same name.
- Each ROW should become a LEAF node: the CELLS for the OUTPUTPORTS will become ASSIGNMENTS.

The complexity is in deriving the inner nodes: the SUBTREE objects. One simple approach is to find a TT INPUTPORT which is (ideally) defined in all the ROWS, and turn it into an inner node (a SUBTREE) which points to the equivalent BDD INPUTPORT and has two TREES:

- The zero subtree, produced from the ROW(s) where the port was 0. This will be a SUBTREE if there are at least two rows in that situation: the transformation should proceed recursively in this case with those rows, excluding the input ports that have already been considered. If there is only one such row, this would simply point to the LEAF created above.
- The one subtree, produced in a parallel way to the zero subtree.

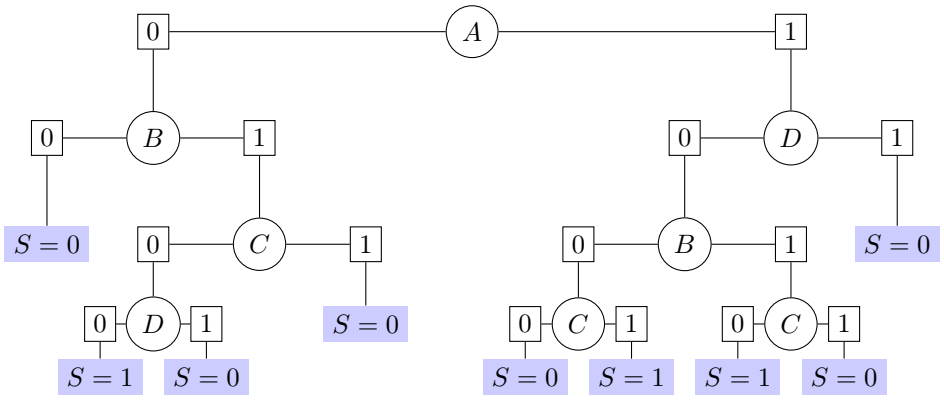

Figure 4: Equivalent BDD for the truth table on Table 1.

This simple approach does not necessarily ensure a minimal subtree, as in some points there may be multiple ports to choose from. It may require improvements for cases where there are no input ports which are defined in all available rows.

3 Main Task

The main task was implementing the transformation, ensuring that the BDDs were equivalent to the original truth table. To simplify the work involved, the case included an Eclipse Modelling Framework implementation, and a set of sample XMI input models conforming to the metamodels.

The `models` folder includes two additional tools:

- A generator that produced truth tables of an arbitrary number of input and output ports, given a seed.
- A validator that checked if a BDD model was equivalent to a source truth table model, by evaluating all possible input combinations through the BDD and comparing the values of the OutputPorts.

Solutions could focus on any specific quality attribute of the transformation beyond optimality and performance. For instance, it may be useful to be able to prove that the transformation does indeed produce a BDD which is equivalent to the TT (even if suboptimal). One of the solutions did focus on verifiability.

4 Benchmark Framework

If focusing on performance, the solution authors had to integrate their solution with the provided benchmark framework. It is based on that of the TTC 2017 Smart Grid case [6], and supports the automated build and execution of solutions. The framework consists of a Python 3.3 script which directs the process, and a set of R scripts which perform basic data analysis and reporting.

The benchmark consisted of three phases: **Initialisation** (setting up the basic infrastructure), **Load** (for loading the models), and **Run** (for executing the transformation and saving the results).

Solutions had to be forks of the main Github project[2], for their later integration into the main Github repository. The solutions could be implemented in any language: they only had to print to the standard output lines in a specific format, mentioning the memory usage and wall clock time spent in transforming each model.

The solutions had to include a `solution.ini` file with instructions on how to automatically build, test, and run them. Solutions were run multiple times, as indicated in the main configuration of the benchmark. To ensure reproducibility, the solutions were integrated into a Docker image, which is automatically built by Docker Hub on each push[3].

5 Evaluation

Given the call for a broader set of research interests in this transformation, the evaluation operated on two dimensions:

[2] https://github.com/TransformationToolContest/ttc2019-tt2bdd
[3] https://hub.docker.com/r/bluezio/ttc2019-tt2bdd-git

- Was it a high-quality model transformation? The transformation should be complete, correct, easy to understand, efficient, and produce optimal results.

  The validator was used to check the produced BDDs against the source truth tables, and the authors had to provide a convincing argument about the correctness of the solution.

  The understandability of the solution was evaluated through an audience survey, and the Docker image was used by the contest organizers to provide independent measurements of memory and time usage in identical conditions. Particularly, Google Cloud Compute `c2-standard-4` images were used.

  Tree sizes for the BDDs were considered for the optimality of the transformations: the smaller, the better.

- Did it highlight a promising research direction? Although the transformation may not be entirely complete or may be harder to understand, it may serve as an example of an active research area within model transformations that the community may wish to showcase.

  This may include aspects such as incrementality, bidirectionality, traceability, verifiability, or the ability to visualize interactively the transformation, among other areas of interest in the field.

  As mentioned before, there was one solution which focused on the verifiability of the solution. This solution received an award for the promising results towards integrating verifiability in a model transformation language.

References

[1] Anthony Anjorin, Thomas Buchmann, and Bernhard Westfechtel. The Families to Persons Case. In *Proceedings of the 10th Transformation Tool Contest*, volume 2026, pages 27–34, Marburg, Germany, July 2017. CEUR-WS.org.

[2] Martin Fleck and Javier Troya. The Class Responsibility Assignment Case. In *Proceedings of the 9th Transformation Tool Contest*, volume 1758, pages 1–8, Vienna, Austria, July 2016. CEUR-WS.org.

[3] Antonio García-Domínguez. TTC'16 live contest case study: execution of dataflow-based model transformations. https://www.transformation-tool-contest.eu/2016/livecontest.html, July 2016. Last accessed on 2019-05-10. Archived on http://archive.is/gHEys.

[4] Sebastian Götz, Johannes Mey, René Schöne, and Uwe Aßmann. Quality-based Software-Selection and Hardware-Mapping as Model Transformation Problem. In *Proceedings of the 11th Transformation Tool Contest*, volume 2310, pages 3–11, Toulouse, France, June 2018. CEUR-WS.org.

[5] Georg Hinkel. The TTC 2017 live contest on transformation reuse in the presence of multiple inheritance and redefinitions. https://www.transformation-tool-contest.eu/2017/solutions_livecontest.html, July 2017. Last accessed on 2019-05-10. Archived on http://archive.is/gHEys.

[6] Georg Hinkel. The TTC 2017 Outage System Case for Incremental Model Views. In *Proceedings of the 10th Transformation Tool Contest*, volume 2026, pages 3–12, Marburg, Germany, July 2017. CEUR-WS.org.

[7] Georg Hinkel. The TTC 2018 Social Media Case. In *Proceedings of the 11th Transformation Tool Contest*, volume 2310, pages 39–43, Toulouse, France, June 2018. CEUR-WS.org.

[8] Guillaume Savaton. Truth Tables to Binary Decision Diagrams, ATL Transformations. https://www.eclipse.org/atl/atlTransformations/#TT2BDD, February 2006. Last accessed on 2019-05-14. Archived on http://archive.is/HdoHM.