Truth Tables to Binary Decision Diagrams in Modern ATL

Dennis Wagelaar
Corilus
Vilvoorde, Belgium
dennis.wagelaar@corilus.be

Théo Le Calvar
LERIA, Université d'Angers
Angers, France
theo.lecalvar@univ-angers.fr

Frédéric Jouault
ERIS, ESEO-TECH
Angers, France
frederic.jouault@eseo.fr

Abstract

Model transformation technology has evolved over the last 15 years, notably regarding scalability and performance. The Truth Tables to Binary Decision Diagrams transformation was written for an early version of ATL roughly 13 years ago. At that time, performance was not as much of a concern as it is today. Not only were execution engines slower then than now, but they also did not provide as many optimization opportunities. Consequently, in its original form, this transformation does not scale well to large models. It remains slow, even when using EMFTVM, the state of the art ATL virtual machine. In this work, we show that by leveraging the profiler, and carefully optimizing the transformation code, significantly improved performance can be achieved. Our updated solution scales up to the largest generated model ($\sim$ 40 MB), which is transformed in about 23 seconds on a modern desktop (Ryzen 5 1600X with 16 GB RAM running Fedora 29): several hundred times faster than the original code.

1 Introduction

This paper presents the ATL/EMFTVM solution to the offline case of the Transformation Tool Contest[1] 2019: Truth Tables to Binary Decision Diagrams (TT2BDD)[2] [GD19]. This case was developed from an ATL transformation published in the ATL Transformation Zoo [Sav06] roughly 13 years ago in 2006: TT2BDD, a model transformation initially written for the first ATL virtual machine ever released. Since that time, two generations of ATL virtual machines have been developed: EMFVM around 2007, and EMFTVM[3] [WTCJ11] from 2011 on.

The resources for the case include the original ATL code from 2006, but run it with EMFTVM. This results in better performance than using the virtual machine of the time, but does not solve the issues that are in the transformation code. Our updated solution[4] is called ATLEMFTVMImproved, and mostly consists in optimizations to the transformation code. These optimizations are described in the following section.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

[1] `https://www.transformation-tool-contest.eu/`

[2] GitHub repository with accompanying resources: `https://github.com/TransformationToolContest/ttc2019-tt2bdd`

[3] Documented at `https://wiki.eclipse.org/ATL/EMFTVM`.

[4] GitHub repository with our solution: `https://github.com/dwagelaar/ttc2019-tt2bdd`

2 Optimizations

In order to optimize the TT2BDD transformation, we used the EMFTVM built-in profiler. This lead us to the three following optimizations:

- **Leveraging attribute helpers caching.** ATL provides two kinds of helpers: operation helpers, and attribute helpers. Attribute helpers are basically similar to parameter-less operation helpers with a significant performance-related difference: their result is cached. Therefore, multiple accesses do not result in multiple computations. The `getTree` operation helper was thus changed into a `tree` attribute helper. With a lower performance impact, the `getNode` operation helper was also changed into a `node` attribute helper.
- **Applying the object indexing pattern**[5]**.** This pattern uses a Map in order to avoid expensive lookups that can be precomputed. A typical example is the navigation of missing opposite references.
- **Leveraging Maps.** In the `getPartition` operation helpers, the original row-accessing code is quadratic: a use of the `exists` iterator in the body of a `select` iterator. We instead compute two Maps: one for each of the possible `true` and `false` values. These Maps are indexed by ports. Moreover, these Maps are computed in attribute helpers, and are therefore cached and computed only once per context. Because EMFTVM uses a HashMap to implement Maps, the resulting code is linear.

Another optimization was performed in the transformation launcher: module loading code was moved into the initialization phase.

3 Conclusion

The state of the art EMFTVM was able to run an old ATL transformation. Although, it cannot automatically optimize it, its built-in profiler makes it possible to quickly spot performance bottlenecks and fix them by making relatively simple changes such as turning parameter-less operation helpers into attribute helpers, or applying well-documented patterns such as the object indexing pattern.

References

[GD19] Antonio García-Domínguez. The TTC 2019 TT2BDD Case (v1.1). In *Proceedings of the 12th Transformation Tool Contest*. CEUR-WS.org, September 2019. To appear.

[Sav06] 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`.

[WTCJ11] Dennis Wagelaar, Massimo Tisi, Jordi Cabot, and Frédéric Jouault. Towards a General Composition Semantics for Rule-based Model Transformation. In *Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems*, MODELS'11, pages 623–637, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] `https://wiki.eclipse.org/ATL/Design_Patterns#Object_indexing`