Logic Model Processing

Pierre Dissaux, Ellidiss Technologies

Model Driven Engineering (MDE) refers to the use of software models to represent System or Software conceptual, logical, or physical abstractions, aiming at standardizing and easing industrial engineering processes. MDE is more and more applied in various domains of application and brings foundations for the digitalization of the industry. Avionics and Space software industry started the move several decades ago, which stimulated the emergence of many innovating technologies to improve the elaboration of these models and their exploitation. Logic Model Processing (LMP in short) is one of them.

Logic Model Processing is an adaptation of Logic Programming to Model Driven Engineering using standard Prolog language. The LMP framework consists of a methodology, a set of tools and Prolog libraries. This technology has been progressively and continuously developed during the last thirty years and produced many outcomes, including publications and tools that have been deployed worldwide for industrial usages.

1. General principle of LMP

Assuming that a modeling language to be processed is defined by a meta-model of any kind, the LMP methodology can be summarized as follows:

- Each class of the meta-model defines a Prolog fact specification (fact template) whose parameters correspond to the attributes of this class.
- An instantiated model (instance of the meta-model) is represented by a populated Prolog facts base (fact instances), where facts parameters values correspond to meta-classes attributes values.
- The model processing program is expressed by a set of Prolog rules bases, referring other rules or the facts base representing the model.
- To execute a LMP program, it is necessary to instantiate the facts base associated with the model to be processed, to merge it with the rules base associated with the processing to be performed and to run a query with a standard Prolog engine.

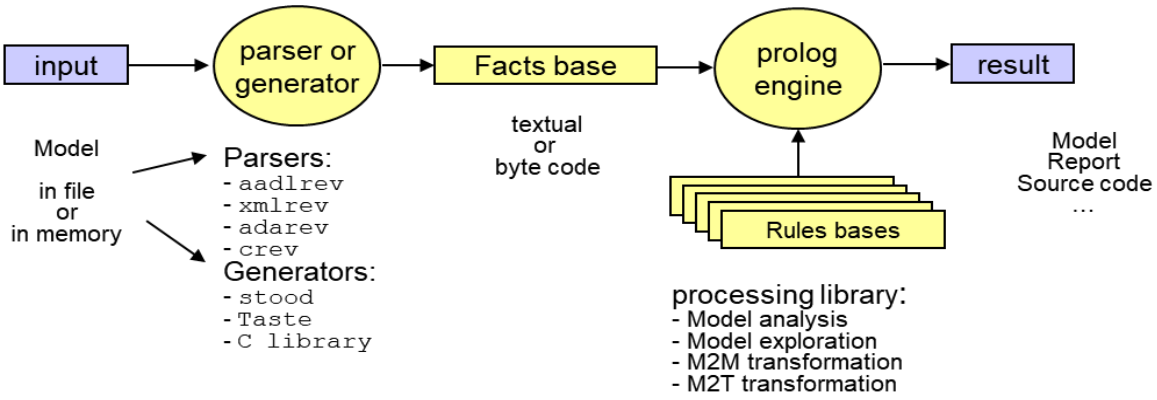

Figure 1: Logic Model Processing

2. Example of LMP applications

LMP has been applied to many kinds of model and for many kinds of processing. Input modeling languages that have been addressed are either token-based languages (i.e., defined by a textual grammar), such as software programming languages (Ada, C,…) or software architecture description languages (HOOD, AADL,…), or tag-based languages (i.e., defined by an XML structure), such as UML, SysML and Domain Specific Languages (DSL).

A parser tool is required in all cases to perform a syntactic translation between the original syntax into a list of Prolog facts. Once the model information is available within a Prolog environment, the following kinds of processing become possible, as far they are properly expressed by Prolog rules.

2.1 Model queries and exploration

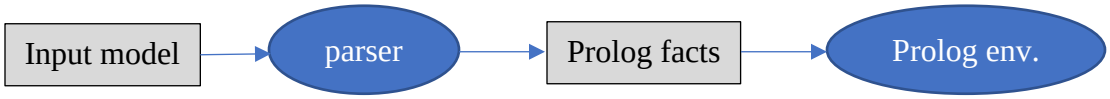

Figure 2: Model queries

This is the simplest processing. The Prolog facts generated by the parser are simply loaded into the Prolog environment and any query becomes available to perform explorations about the model contents.

2.2 Model verification

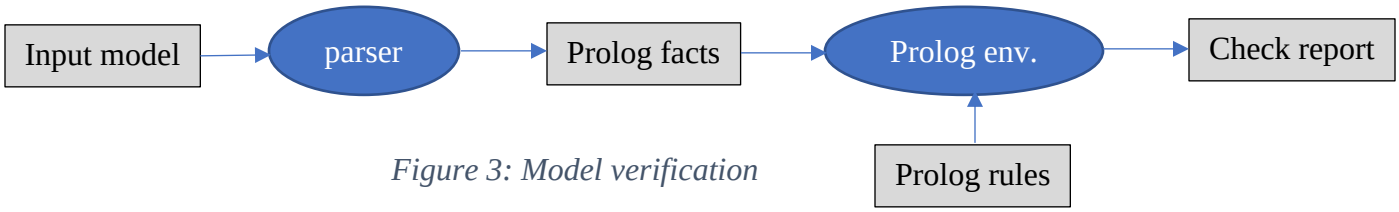

Figure 3: Model verification

With this kind of model processing, the goal consists in doing static analysis of the input model and produce a textual check report summarizing the compliancy with regards to a set of verification rules that must be expressed in Prolog. An example of model verification is the Requirement coverage analysis of a Design model.

2.3 Model transformation

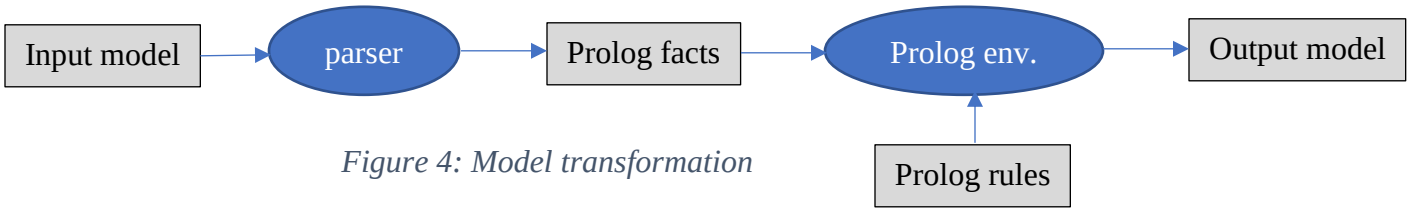

Figure 4: Model transformation

This processing chain looks very similar to the preceding one. However, in this case the rules must implement a transformation of the input model into an output model, that may be different (heterogeneous transformation) or of the same nature (homogeneous transformation). Such rules must implement a conceptual mapping between two languages and may be quite complex. An example of model transformation is the link between a System engineering model and its Software subsystems to ensure a digital continuity within a development environment.

2.4 Inline model processing : LAMP

The three kinds of model processing presented above require that a set of Prolog rules are prepared at tool design time (offline) and embedded into a proper tool chain in advance. In these cases, the end user cannot modify the processing rules at model design time (inline).

In the context of a dedicated tool (AADL Inspector), an inline Prolog processing feature has been implemented. The Prolog rules can thus be embedded inside the model to be processed and potentially modified by the designer at any time. This variant of LMP is called LAMP (Logical AADL Model Processing) and allows for Prolog code to be inserted inside AADL as annexes and processing goals to be executed as Prolog queries. LAMP comes with a library of reusable Prolog model exploration, verification, and transformation rules. Note that AADL (Architecture Analysis and Design Language) is used for the design of real-time critical systems in the aerospace industry.

3. Conclusion

The benefits of the LMP approach are multiple:

- It uses an ISO standard language (Prolog): There is no language specification and maintenance cost. Moreover, its semantics is formally defined and many tool implementations and learning material are available.
- The declarative style of the Prolog language is very appropriate to specify queries and processing rules; implicit loops make Prolog programs more readable.
- The clean separation between the facts bases (input data) and the rules bases (program) brings robust and secure model processing implementations.
- The approach can be applied to any kind of data source, in memory or in a file, given that there is a way to convert it into a facts-base in its textual or binary form. Usual parsing technologies can be used for this purpose.
- Facts bases generated from different data sources can be merged to perform cross-models processing.

After almost 30 years of development, and a deployment within the industry where it is often not directly visible to the end users, several future enhancements of LMP are still under study.

References:

[1] "Model Verification: Return of Experience", P. Dissaux and P. Farail, ERTS 2014.
[2] "Merging and Processing Heterogeneous Models", P. Dissaux and B. Hall, ERTS 2016.
[3] "LAMP: A new model processing language for AADL", P. Dissaux, ERTS 2020.