Algebraic MOF Framework

Contributed by Artur Boronat
Monday, 11 June 2007
Last Updated Friday, 27 July 2007

The Meta-Object Facility (MOF) standard describes a generic framework where the abstract syntax of modeling languages can be defined. This standard aims at offering a good basis for Model-Driven Development processes, providing some of the building blocks that are needed to support a Model-Driven Development approach: what is a model, what is a metamodel, what is reflection in a MOF framework, etc. However, most of these concepts lack at present a formal semantics in the current MOF standard. The MOF standard also provides the so-called MOF-Reflection facilities, by means of a generic API, to manipulate software artifacts that are made up out of objects. Broadly speaking, reflection is the capacity to represent entities that have a formal semantics at a base level, such as types, as data at a metalevel. Metalevel entities, once metarepresented, can be computationally manipulated and transformed. This notion of reflection is still not supported in the MOF standard.

In the Algebraic MOF Framework, we define a reflective, algebraic, executable framework for precise metamodeling that provides support for the MOF and the OCL standards. On the one hand, our formal framework provides a formal semantics for the building blocks that are usually involved in a Model-Driven Development process. On the other hand, our framework provides an executable environment that is plugged into the Eclipse Modeling Framework (EMF) and that constitutes the kernel of a model management framework, supporting model transformations and formal verification techniques. Once installed, the plugin adds a menu, called AlgebraicMOF. When we choose a XMI file that contains an EMF odel definition, this menu provides several functionalities:
- ToMetaObjectConfiguration, (3) in Fig. 1: generates the algebraic term that represents a model definition as a collection of metarepresented objects. This functionality is enabled for files with extension .xmi.
- FromMetaObjectConfiguration, (2) in Fig. 1: obtains an EMF model from a collection of metarepresented objects. This functionality is enabled for files with extension .maude. A file of this kind must contain a first line with the URI that identifies the corresponding metamodel definition, and the term representing the collection of objects. The uri has to be given using a specific format: ***$ nsPrefix - "nsUri"; where nsPrefix represents the value of the nsPrefix attribute of the root Package instance of the metamodel definition and nsUri represents the value of the nsUri attribute of the of the root Package instance. For example to parse a collection of objects that represents an Ecore model, we have to add the following line: ***$ ecore - "http://www.eclipse.org/emf/2002/Ecore". To use this functionality (serialization of EMF models to terms), the corresponding EMF metamodel (an Ecore model) has to be registered previously. This can be achieved in several ways: by generating the plugin of the metamodel by means of EMF, or by using the EMF dynamic capabilities, see this article for an introduction.
- ToTheory, (1) in Fig. 1: generates a set of Maude modules that represents the MEL theory that provides the algebraic semantics of an Ecore metamodel.

Fig. 1. The algebraic MOF framework into the Eclipse platform.

Requirements for the installation:
- Maude 2.3
- Eclipse 3.2.1
- EMF 2.2.1
- Maude Development Tools
- MOMENT2

All the binaries that are needed can be found here.

Installation guidelines:
- Install Maude 2.3: file MaudeFW_2.3.exe
- Install Eclipse 3.2.1: file eclipse-SDK-3.2.1-win32.zip
- Download the file 20070727_MOMENT2-update-site.zip from here. Once Eclipse has been installed, in the menu Help | Software Updates | Find and Install... | New Archived Site...,select the downloaded file and follow the standard intallation procedure. You can use the instructions given for the Maude Development Tools plugin as an example.Configure the Maude preferences in the menu Windows | Preferences | Maude Preferences, indicating where you have installed Maude 2.3

Note: this version is not compatible with MOMENT-QVT so that you cannot have both plugins in the same Eclipse instance at the same time by now.

Instructions of use:

1. Steps to obtain the MEL theory that corresponds to an EMF metamodel:

1.1 Select the corresponding Ecore model, i.e., the EMF metamodel.

1.2. Right button of the mouse and select AlgebraicMOF | ToTheory

1.3. To load the theory in Maude, you have to have loaded the kernel of MOMENT2 before. The kernel can be found here.

2. Steps to obtain the term that represents a model:2.0. If the metamodel of the model to be serialized is not Ecore.ecore then, the metamodel MUST be registered before:
2.0.a) Click the right button of the mouse on the Ecore model.
2.0.b) Select MOMENT Extensions | register model.
2.1. Right button of the mouse on the model file and click on AlgebraicMOF | ToMetaObjectConfiguration.

3. Steps to obtain the term that represents a model:3.0. If the metamodel of the model to be serialized is not Ecore.ecore then, the metamodel MUST be registered before:
3.0.a) Click the right button of the mouse on the Ecore model.3.0.b) Select MOMENT Extensions | register model.3.1. Be sure that a .maude file contains the term that represents the model and that the first line contains the metainformation of the metamodel as indicated before. For example to parse an ecore model, the first line of the file should be: ***$ ecore -"http://www.eclipse.org/emf/2002/Ecore".3.2. Right button of the mouse on the .maude file that contains the term representing the model, and click on AlgebraicMOF | FromMetaObjectConfiguration.

Acknowledgements

This work has been performed in a collaboration with Prof. José Meseguer, at the University of Illinois at Urbana-Champaign.