$\mathcal{G}rimm$: a Tool for Model Generation

Adel Ferdjoukh, Lirmm

Contents

1 What is $\mathcal{G}rimm$?

$\mathcal{G}rimm$ is a tool whose the purpose it to generate models conforming to meta-models in *ecore* format. It is encoded in *EMF/Java*.

This tool takes as inputs a meta-model conform to *ecore* and configuration data on the models to generate.

It brings two kinds of outputs. The first one is a *xmi* format model which could be manipulated in eclipse and the second one is tu visual representation of this same model generated by the tool *dot* of software *GraphViz*.

2 Functioning Conditions

To make $\mathcal{G}rimm$ working, the computer of a user must gather the following conditions:

1. Install the JVM (*The Java Virtual Machine*).
2. Download the $\mathcal{G}rimm$ archive which contains the tool and the CSP solver Abscon.
3. Install *GraphViz* to be able to visualize the generated models.

3 Execution with simple parameters

After gathering the functioning conditions of $\mathcal{G}rimm$, you are able to spread to the execution step. To do this, you should first extract the files form the $\mathcal{G}rimm$ archive then follow the following steps:

1. Choose one of the meta-models attached with the archive.
2. Recover from the table figure 1 the root class of chosen meta-model.
3. Lunch a shell terminal and execute this following command:

   ```
   java -jar grimm.jar -mm=mm.ecore -rootClass=root -lb=n -ub=m -rb=l
   ```

   where:

   mm.ecore: File path of chosen meta-model.

   root: Root class of chosen meta-model.

   n: Instances per class lower bound.

   m: Instances per class upper bound.

   l: References upper bound (for unbounded ones).

4. Then follow instructions on the terminal. Generated models are saved in a folder named *root* (different for each meta-model).

Meta-model	File path	Root class
Test	test.ecore	Compo
PetriNet	PetriNet.ecore	PetriNet
Entities & Relashionships	ER.ecore	Schema
Royal and Loyal	RoyalAndLoyal.ecore	LoyaltyProgram
BibTex	BIBTEXML.ecore	BibtexFile
BMethod	BMethod.ecore	BSpec
Sad	Sad3.ecore	DocumentRoot
Diagraph	Diagraph.ecore	GraphModel
Business Process	BusinessProcessModel.ecore	CoumpoundTask
MyUML	MyUML.ecore	Model
Ecore	Ecore.ecore	EClass
Jess	jess.ecore	JessModel
Maps	maps.ecore	map

Figure 1: Meta-models benchmark with their root classes.

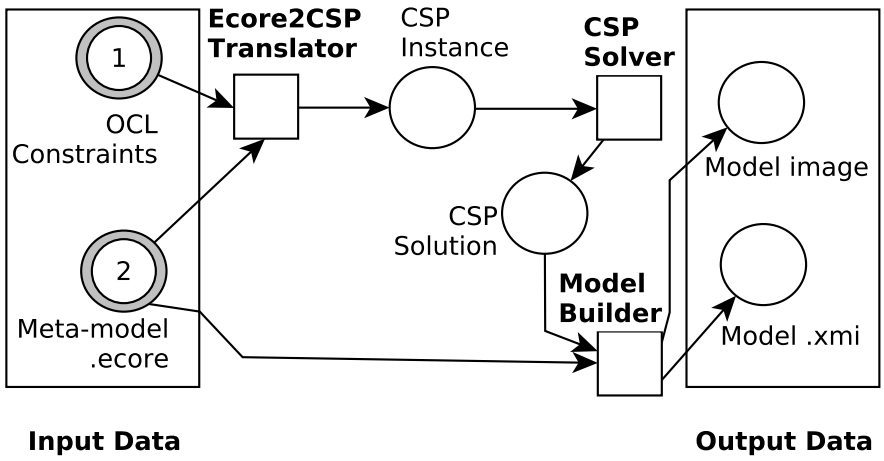

Figure 2: Generation of models with $\mathcal{G}rimm$: Process steps (simple parameters).

The steps of the process of model generation using $\mathcal{G}rimm$ are:

1. From a meta-model, generates a csp instance according to the CSP modelling of a meta-model in [FBC+13].
2. Use *Abscon* CSP solver to solve the csp instance obtained in step 1 ([MLB01]).
3. Build a valid model according to the values of the solution given by the solver.

These three steps are also shown in the Petri net in figure 2.

4 Execution using a configuration file

If you need more control and more options (for example choose a number of instances for each class separately), you can use to generate using a configuration file.

Grimm can generate a pre-filled configuration file using the following command:

```
java -jar grimm.jar -mm=file.ecore -rootClass=root
```

After that, you have to complete the filling of that generated file (root.grimm) and to lunch the following command:

```
java -jar grimm.jar -mm=file.ecore -rootClass=root -configFile=root.grimm
```

The figure 3 gives an example of a configuration file.

```
% -- -- -- -- -- -- -- -- -- -- -- -- --
%Number of instances for Classes
Street=1
Boulevard=2
Pedestrian=1
Garden=2
Square=1
% -- -- -- -- -- -- -- -- -- -- -- -- --
%Domains of the features
map/name=1 2
Street/name=1 2 3 4 5 6
Street/length=-10 200 400 500 150
Boulevard/name=10 12 14 15 18
Boulevard/length=1000 2000 3000 4000
Pedestrian/name=1 2 3 4 5 6
Pedestrian/length=600 700 900
Garden/name=4 6 7 8 9
Square/name=21 22 23 24 25
% -- -- -- -- -- -- -- -- -- -- -- -- --
%Some others
RefsBound=3
FeaturesBound=2
```

Figure 3: An example of a *Grimm* configuration file.

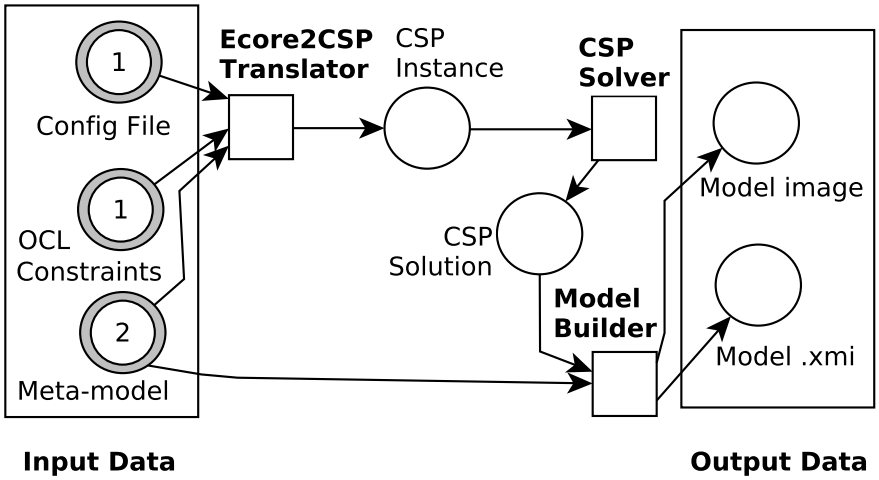

Figure 4: Generation of models with $\mathcal{G}rimm$: Process steps (simple parameters).

5 Other options

To see the other options that $\mathcal{G}rimm$ offers, you can execute this command:

```
java -jar grimm.jar
```

References

[FBC+13] Adel Ferdjoukh, Anne-Elisabeth Baert, Annie Chateau, Rémi Coletta, and Clémentine Nebut. A csp approach for metamodel instantiation. In *ICTAI 2013, Internationnal Conference on Tools with Artificial Intelligence, November 4-6, Whashington D.C., USA.*, 2013.

[MLB01] Sylvain Merchez, Christophe Lecoutre, and Frédéric Boussemart. Abscon: A prototype to solve csps with abstraction. In *CP*, pages 730–744, 2001.