Framework for automated analyses of feature models

User Guide

Content

1. Introduction

FAMA-FW is a Framework for automated analysis of feature models integrating some of the most commonly used logic representations and solvers proposed in the literature (BDD, SAT and CSP solvers are implemented). FAMA is the first tool integrating different solvers for the automated analyses of feature models.

Welcome to the FAMA Tool Suite user's guide. With this guide, you will learn how to use our framework, that it is intended to be a reference in Features Model Analysis.

2. FAMA Tool Suite files

FAMA.jar: This is the core application. It must be included in the build path.

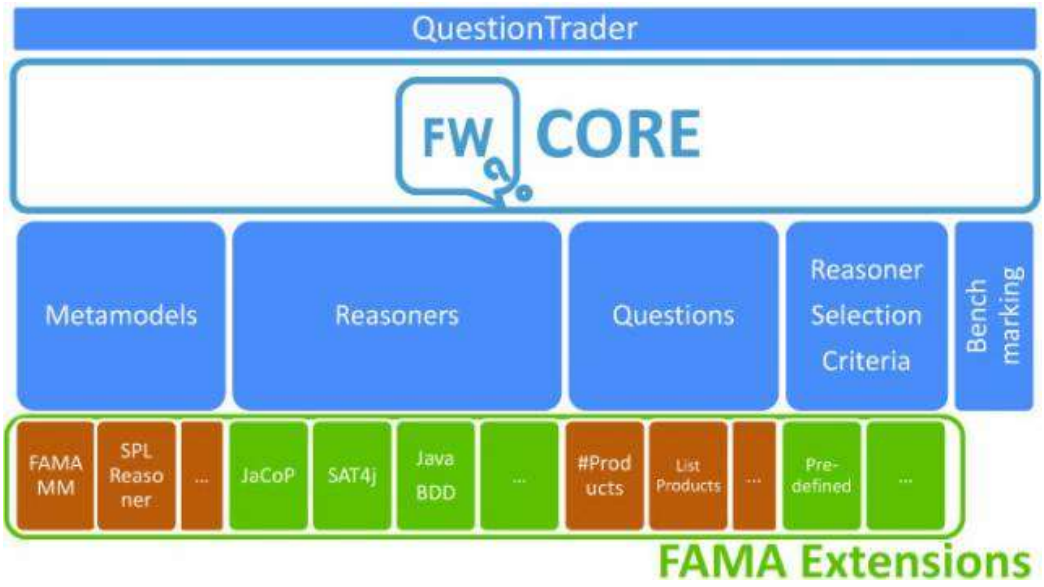
FAMAquestions.jar: This library includes the questions interfaces that are implemented in the reasoners

FAMAmodel.jar: This library includes the specification of the FAMA Feature Model. This is the variability model provided with FAMA Tool Suite by default.

JavaBDDReasoner.jar/ Sat4jReasoner.jar: The reasoners that FAMA Tool Suite uses to answer the questions. You can also use another one.

FAMAconfig.xml: This is the main configuration file where we could define the questions, reasoners and criteria we want FAMA Tool Suite to use.

3. FAMA architecture



We can structure FAMA files in three types:

- **FAMA core**: FAMA.jar & FAMAquestions.jar
- **FAMA metamodel**: FAMAmodel.jar
- **FAMA reasoners**: JavaBDDReasoner.jar & Sat4jReasoner.jar

4. Quick Start

At first, you will need all pieces of software that were needed, all them are included in a zip. You can find it at www.isa.us.es/fama.

Once you have downloaded it, the first thing you must do is unzip the file. Inside it, you will find three folders (lib, with the jars; docs, with the user guide, and fm-samples, with feature models samples on xml format), and four files (the version file, the license file, a configuration file called FAMAconfig.xml, and the FaMaTS logo).

4.1 Creating and configuring a new empty project.

Now, you have uncompressed FaMaTS. The next step is to create a new empty java project in your IDE, and take a new class with a main method. Also, you should include in the project the FaMaTS libraries. For this, create a folder on the project called `lib`, copy inside all the jars that are on `lib` folder on FaMaTS.zip, and copy also FAMAconfig.xml and one fm-sample on the project's root folder. Finally, add FAMA.jar to the build path of the project,

Copy the code of the `Figure1` on the main method, and run your first FaMaTS test (with fama.test)

```
QuestionTrader qt = new QuestionTrader();
GenericFeatureModel fm = null;

fm = (GenericFeatureModel) qt.openFile("test.fama");
qt.setVariabilityModel(fm);

ProductsQuestion q = (ProductsQuestion)qt.createQuestion("Products");
qt.ask(q);

System.out.println(q.getNumberOfProduct());
```

Figure 1

Do not forget the imports:

```
import es.us.isa.FAMA.Reasoner.QuestionTrader;
import es.us.isa.FAMA.Reasoner.questions.ProductsQuestion;
import es.us.isa.FAMA.models.featureModel.GenericFeatureModel;
```

Figure 2

4.2 Main classes

Now, we are going to detail the classes that appear on the example.

-**QuestionTrader**: This class is the main interface of the tool.

-**GenericFeatureModel**: This class represents an abstraction of a feature model, hiding our implementation of the VM. We can load it from a file (common)[1].

-**Question** (and subclasses): Question class (and its subclasses, detailed more ahead) represents the abstraction of the different questions that can be answered by FAMA.

4.3 Getting a model

The first step is to create a QuestionTrader object.

```
qt = new QuestionTrader();
```

Now, we need a model to work. There are two ways to give a model to FAMA Tool Suite. We can load the model from a file, or build it from scratch and save into a file. The common way is load the model from a file. Here there's a example (with a file called test.fama).

```
GenericFeatureModel fm = (GenericFeatureModel)qt.openFile("test.fama");
qt.setVariabilityModel(fm);
```

4.4 Creating and asking questions

At this time, we can create questions, and ask these questions to FAMA. Calling the `createQuestion` method of the class QuestionTrader we get a `Question` object.

```
Question q = qt.createQuestion("Products");
```

When we have the question yet, the next step is to ask to FAMA.

```
PerformanceResult pr = qt.ask(q);
```

Finally, we can retrieve the answer of the question, as follows, and extract the information about the question.

```
ProductsQuestion pq = (ProductsQuestion) q;
Integer np = pq.getNumberOfProduct();
```

1: It is also supported to build a feature model from scratch.

Here, we offer the different questions that are implemented now.

Question	Id	Class
Number of products	#Products	NumberOfProductsQuestion
Products	Products	ProductsQuestion
Valid model	Valid	ValidQuestion
Commonality	Commonality	CommonalityQuestion
Filter	Filter	FilterQuestion

Figure 3

Note: these identifiers are on FAMAconfig.xml

And these are the methods for each question, to retrieve information about the model.

Question	Methods
Number of products	`getNumberOfProducts(): long`
Products	`getNumberOfProduct(): int` `getProduct(int index): Product`
Valid model	`isValid(): boolean`
Commonality	`setFeature(GenericFeature f): void` `getCommonality(): int`
Filter	`addFeature(GenericFeature f,int cardinality): void` `removeFeature(GenericFeature f): void`

Figure 4

4.5 Questions

In this section, we will explain all the questions supported now. To create a question, you should ask to QuestionTrader. Above you can see the identifiers for that.

4.5.1 Products

This is the most common question for a feature model. What products can I have with a specified feature model? There are not methods to call before ask to FAMA for this question. Once we have asked to FAMA, we have two possible methods to use:

-`getNumberOfProduct()` returns the total number of products of the model.

-`getProduct(int index)` returns the product (class Product) associated with that index. A product have a list of features that we can consult (getFeature(int index) method, or getNumberOfFeatures()).

4.5.2 Number of products

NumberOfProducts can be considered a subquestion of Products. Only have one method:

-`getNumberOfProduct()` returns the total number of products of the model.

4.5.3 Valid

This question determines if the model is valid, and also has one method only.

`-isValid()` returns a boolean that indicates if the model is valid, or not.

4.5.4 Commonality

The goal of this question is to determine the commonality of a feature. Hence, before we call to the ask method, it is necessary to specify a feature. The methods are:

`-setFeature(Feature f)`: with this method, we specify the feature that we want to know his commonality

`-getCommonality()`: once we have specify the feature and we have ask to FAMA, it's the moment to use getCommonality, to obtain the result.

4.5.6 Filter

With FilterQuestion, we can specify a filter on our model, adding or removing features (before)

```
-addFeature(GenericFeature f, int cardinality)
```

```
-removeFeature(GenericFeature f)
```

Then, we can ask other questions over this filtered model. For this, FilterQuestion needs an additional question called SetQuestion. The goal of SetQuestion is to have a set of questions, and throw them one after one.

```
-addQuestion(Question q)
```

First, we create the FilterQuestion, and we add o remove features. Later, creates others questions that we consider. Finally, we create the SetQuestion, we add the FilterQuestion first, after we add the others questions, and ask to the QuestionTrader for the SetQuestion.

4.6 FAMAconfig.xml

On the FAMAconfig.xml, we tell FAMA what reasoners, criteria selectors, questions and feature models are available to use. If you are a common user, you don't need edit this file. But if you are interested on add a new reasoner, or use your own criteria selector, for example, you must write some lines on FAMAconfig.xml.

Also maybe you can be interested on use a concrete reasoner to answer the questions. For this, you have to remove other lines about reasoners on FAMAconfig.xml, and stay alone the line about your reasoned. Remember to put the lines again if you want to use all the reasoners!

4.7 Examples

Finally, we show two examples (for copy and paste). Figure 5 shows a simple NumberOfProducts question, and Figure 6 shows a FilterQuestion (together NumberOfProductsQuestion), because it can be the most confused question to ask.

```
import es.us.isa.FAMA.Reasoner.QuestionTrader;
import es.us.isa.FAMA.Reasoner.questions.NumberOfProductsQuestion;
import es.us.isa.FAMA.models.featureModel.GenericFeatureModel;


public class NumberOfProductsFAMATest {

      public static void main(String[] args) {

              QuestionTrader qt = new QuestionTrader();
              GenericFeatureModel fm = null;

              fm = (GenericFeatureModel) qt.openFile("test.fama");
              qt.setVariabilityModel(fm);

              NumberOfProductsQuestion q =
 (NumberOfProductsQuestion) qt.createQuestion("#Products");
              qt.ask(q);

              System.out.println(q.getNumberOfProducts());

      }

}
```

Figure 5

```
import es.us.isa.FAMA.Reasoner.QuestionTrader;
import es.us.isa.FAMA.Reasoner.questions.FilterQuestion;
import es.us.isa.FAMA.Reasoner.questions.NumberOfProductsQuestion;
import es.us.isa.FAMA.Reasoner.questions.SetQuestion;
import es.us.isa.FAMA.models.featureModel.GenericFeatureModel;

public class FilterFAMATest {

      public static void main(String[] args) {
              QuestionTrader qt = new QuestionTrader();
             GenericFeatureModel fm = null;

             fm = (GenericFeatureModel) qt.openFile("test.fama");
             qt.setVariabilityModel(fm);




             FilterQuestion fq =
                  (FilterQuestion) qt.createQuestion("Filter");
             NumberOfProductsQuestion nq =
      (NumberOfProductsQuestion) qt.createQuestion("#Products");
             SetQuestion sq = (SetQuestion) qt.createQuestion("Set");

             fq.removeFeature(fm.searchFeatureByName("root"));
             sq.addQuestion(fq);
             sq.addQuestion(nq);

             qt.ask(sq);

             System.out.println(nq.getNumberOfProducts());
      }

}
```

Figure 6