Language-Driven Engineering Tutorial 2

2 November 2018

1 Evolving the Modelling Language and its mIDE

You have seen how an mIDE for Binary Decision Diagrams can be evolved to one for decision diagrams that operate on min-max fuzzy logic. The evolution requires only simple changes, which, however, are spread across the entire project. This is cumbersome especially if you plan to adapt the logic semantics more than once.

In this session, you will learn to describe any logic with a domain-specific language (DSL). The complete mIDE can be generated from this description fully automatically. We will first see, how the mIDE for Binary Decision Diagrams can be expressed and generated in this fashion. We will then evolve the mIDE for decision diagrams to utilise min-max fuzzy logic and probabilistic fuzzy logic.

1.1 Generate the Mindset-Supporting Integrated Development Environment (mIDE) for Binary Decision Diagrams

We have prepared a DSL description of Boolean algebra for you. We will use this to generate a tool similar to that used in the previous session.

1. Start `cinco` and choose a **new workspace** for this tutorial (must be in the same folder as the old workspace) and `Launch`.
2. Install the Eclipse plugin prepared to generate mIDEs for decision diagrams.
   (a) In the main toolbar go to `Help` > `Install new Software...` > `Add...` > `Local...`
   (b) Choose the update site `add-lib-meta-mide-update-site`. Double-check that this is a normal folder (unzip if provided as `*.zip` file).
   (c) `Add` the site, select `ADD-Lib Meta mIDE` > `Next` > `Next` `I accept the terms of the license agreement` > `Finish` > `Install anyway` > `Restart Now`

3. Import the prepared meta-mIDE project `info.scce.addlib.mide.somelogic.zip` that we will evolve from one logic to another:
   (a) In the main toolbar choose `File` > `Import...` > `General` > `Existing Projects into Workspace` > `Next`.
   (b) Choose the option `Select archive file` and `Browse...` your file system to import the prepared project archive `info.scce.addlib.mide.somelogic.zip`. Click `Finish` to finalise the import.
4. Find and open the file `SomeLogic.as`. The file defines the algebraic structure to underly decision diagrams in the generated mIDE. In this case, the standard Boolean logic is defined with the carrier set $\mathbb{B}$, binary operations `and`, `or`, and a single unary operation `not`. The semantics are implemented in the referenced Java class `SomeLogic`.
5. Find and open the Java class `SomeLogic` located in the package `info.scce.addlib.mide.somelogic`. The class implements the logic's semantics in one a simple Java method per operator.
6. From the DSL description of the logic the mIDE is generated in two steps. First, a CINCO product model is generated automatically upon saving the file in `as-model-gen`. Find this folder and select the `DDTool.cpd` file to manually generate the CINCO product: right-click on the file > `Generate Cinco Product`. The process may take some seconds to minutes. In case an error happens during generation, retry once and then ask for help, if the problem remains.
7. You can now start the new mIDE: right-click the main project `info.scce.addlib.mide.somelogic` and choose `Run As` > `Eclipse Application`. This will start a new instance of Eclipse: the generated mIDE for Binary Decision Diagrams.
8. Start the simulation again and verify that the generated mIDE behaves exactly as the previously hand-coded one. Close this mIDE before you proceed.

1.2 From Boolean Logic to Min-Max Fuzzy Logic

You successfully generated and used your first mIDE. Now evolve the language for decision diagrams to deal with uncertainty using a fuzzy logic. A particularly simple fuzzy logic is min-max logic. Its carrier set is the real interval $[0, 1]$ and it defines logic operations as follows:

$$\begin{aligned} MinMaxLogic := (\ [0,1],\ \{\ \wedge_m,\ \vee_m\ \},\ \{\ \neg_m\ \}\ ) \text{ with} \\ a \wedge_m b := min(a, b) \\ a \vee_m b := max(a, b) \\ \neg_m a := 1 - a. \end{aligned}$$

1. Find and open `SomeLogic.as` and ensure that the carrier set can represent degrees of certainty [0, 1]. Use `Real` instead of `Bool`.
2. Save the file. This triggers the generation of CINCO model files (1st generation step). You can find the resulting `DDModel.mgl` in `as-model-gen`.
3. Adapt the semantics implementation in the Java class `SomeLogic`: methods implementing fuzzy operators (`and`, `or`, `not`) should take and return `double` values.
4. Generate the CINCO product: select and right-click the file `DDTool.cpd` > `Generate Cinco Product`.
5. Start the new mIDE for fuzzy decision diagrams: select and right-click the project `info.scce.addlib.mide.somelogic` > `Run As` > `Eclipse Application`.

1.3 Modelling Min-Max Fuzzy Logic Decision Diagrams

You can now try to open the models you created in the previous session. You will see that this will not work without making some slight changes. The terminals of the new fuzzy decision diagrams hold real values instead of Boolean values. You can fix this easily by replacing the Boolean values with 1.0 or 0.0:

1. Open your decision diagram model `MyEmailClassificationRules.dd` from the previous session if it is not already open.
2. You will see an error indicating the expected problem and subsequently a textual representation of the model.
3. Change the values of the terminals from `true`/`false` to real values in the range [0, 1]. Affected lines are highlighted in red.
4. Switch from `Source` to `Diagram` view at the bottom left after resolving all errors.
5. Open your composition model `MyEmailClassificationService.comp`. This model should still be valid in the new mIDE.
6. Take your time and model your fuzzy email classification service. Make sure to use the full range of terminal values [0, 1].
7. When you finished modelling your decision service, generate its Java implementation fully automatically by clicking the `G`-button in the Eclipse toolbar.
8. We will again start the email simulation and use your fuzzy email classification service. Find the main method in the class `Application` and ensure the simulation includes your service as a fuzzy service in the simulation (use the code from the comments).
9. Run the email simulation application again: right-click the class `Application` and choose `Run As` > `Java Application`.

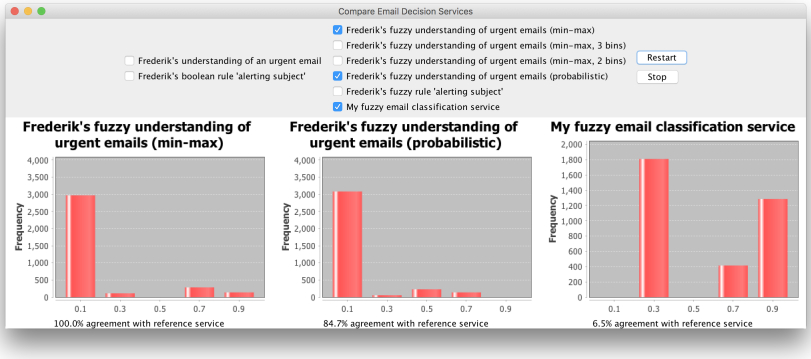

Figure 1: The email simulation using three fuzzy services to classify emails

1.4 From Min-Max Fuzzy Logic to Some Other Fuzzy Logic

There exist various different fuzzy logics. Using any of these in our mIDE requires only small changes in DSL for logics. Choose a fuzzy logic of your choice and evolve the mIDE to incorporate its semantics.

To give another example: probabilistic logic is also a variant of fuzzy logics that resembles probabilities, with the implicit assumption that variables are independent. It is defined as follows:

$$\begin{aligned} \mathit{ProbabilisticLogic} := (\ [0,1],\ \{\ \wedge_p,\ \vee_p\ \},\ \{\ \neg_p\ \}\ ) \text{ with} \\ a \wedge_p b := a * b \\ a \vee_p b := 1 - ((1-a) * (1-b)) \\ \neg_p a := 1 - a. \end{aligned}$$

1. Adopt the Logic definition in `SomeLogic.as` if needed.
2. Implement the semantics of your fuzzy logic in the class `SomeLogic`.
3. Regenerate the mIDE: Find and right-click the file `DDTool.cpd` > `Generate Cinco Product`.
4. Start the new mIDE: right-click the main project `info.scce.addlib.mide.somelogic` and choose `Run As` > `Eclipse Application`.
5. Regenerate your email classification model `MyEmailClassificationService.comp` with the G-button.
6. Run the email simulation application: right-click the class `Application` and choose `Run As` > `Java Application`.