FuzzyLite v1.02
A Fuzzy Inference System written in C++

by Juan Rada-Vilela

Supported by the Foundation for the Advancement of Soft Computing

September, 2010

Contents

List of Figures

Chapter 1

Overview

1.1 Introduction

`FuzzyLite v1.02` is a multiplatform, free, and open-source Fuzzy Inference System (FIS) written in C++ and released under the Apache License 2.0, which makes this software freely available for commercial and non-commercial use. The idea behind this FIS is to have a very simple and lite FIS. Simple as in simple to use, simple to understand, and simple to extend, without sacrificing performance. And lite because it requires no additional libraries more than the Standard Template Library included in the C++ Standard Library. It has an object-oriented approach and a clear separation between the headers and sources, so it is easy to extend. Furthermore, it is GUI-agnostic, meaning that the FIS does not require a GUI to run, encouraging its use as a library. Nevertheless, a Qt-based GUI is provided using `FuzzyLite v1.02` as a shared library.

1.2 Features

- Linguistic terms are continuous and the following ones are available: triangular, trapezoidal, rectangular, shoulder, singleton, custom function, and compound (multiple functions).

- Export any fuzzy system to text using a slightly modified version of the Fuzzy Controller Language (FCL).

- Defuzzification using center of gravity (COG).

- Mamdani rule parsing with grammar checking.

- Takagi Sugeno rules of any order (e.g. f(x) = (sin x) / x, f(x) = 0.5 * input-1).

- Weights for each rule.

- TNorm: minimum, product, bounded difference.

- SNorm: maximum, sum, bounded sum.

- Modulation: clipping, scaling.

- Aggregation: maximum, sum, bounded sum.

- Variable sampling size for membership functions to compute area and centroid.

- Triangulation and Trapezoidal algorithm to compute the area and centroid.

- Hedges: not, somewhat, very, any.

- Very easy to implement and incorporate new linguistic terms, defuzzification methods, fuzzy rules (antecedents and consequents), fuzzy operations (T-Norms, S-Norms, methods for modulation and aggregation), algorithms for computing the area and centroid of linguistic terms, hedges, among other things.

1.3 What's new?

1.3.1 `FuzzyLite v1.02`

- Fixed the Triangulation Algorithm to include the first and last triangle (improved accuracy). Courtesy of rctaylor.

- Implemented the Trapezoidal Algorithm suggested by William Roeder, although its accuracy is not as good as the Triangulation algorithm.

- Created the scripts for building fuzzylite as static and dynamic library, as well as building a demo version of it. Tested on Linux Ubuntu and Mac OS X 10.5. Although the Unix version should work under Windows as well using G++.

- Added LeftShoulderTerm and RightShoulderTerm, just to provide a better understanding when configuring the FuzzyEngine.

- Changed all the `#include <fuzzylite/?.h>` for `'' fuzzylite/?.h''`.

- Included the Trapezoidal Algorithm into the GUI.

1.3.2 `FuzzyLite v.1.01`

- The source can be built on Linux with no need to add several includes to some files. (I work on MacOSX and I did not build fuzzylite on Linux, I just assumed it would build just fine, but some includes were missing in some files. This was FIXED).

- Several scripts for building fuzzylite using a simple make. These scripts are created automatically by NetBeans, however, you do not need NetBeans to build fuzzylite nor the gui. The scripts are available for Linux and Mac OS X.

1.3.3 `FuzzyLite v1.0`

- The GUI is working again.

- A class diagram of `FuzzyLite v1.02`.

- A detailed explanation of `FuzzyLite v1.02`.

- Minor changes.

1.4 What's next?

- Figure out why the accuracy of the Trapezoidal Algorithm is not as good as the Triangulation algorithm.

- Fix the GUI so Takagi Sugeno rules can be tested with their respective graph.

- Improve the `InfixToPostfix` class so infix functions are parsed as one would normally expect.

- Load the fuzzy engine from text using the Fuzzy Controller Language (FCL).

- Include more linguistic terms (e.g. sigmoidal, gaussian, sine, cosine).

- Include more defuzzifiers (e.g. Right Most Maximum, Left Most Maximum, Mean Maximum).

- Make some functions inline to increase performance and check those that are already inline to ensure they do increase performance.

1.5 Known bugs

- Accuracy of Trapezoidal Algorithm is not as good as Triangulation Algorithm. Although I do not know whether it is a bug or a feature ;).

- `InfixToPostfix` conversion might not parse functions as one would normally expect. For example, $\sin(x)/x$ is *not* equivalent to $(\sin x)/x$, the latter yields the result one might expect. Make sure by validating the postfix expression, or by evaluating its results.

- In the GUI, when using a Takagi Sugeno system, the output graphs do not work.

1.6 Building from source

1.6.1 FuzzyLite v1.02

Inside `./fuzzylite` there are 6 folders, on each there is a `makefile`. So all you have to do is execute from the folder the command `make`. The folders are described below (notice that `[OS]` stands for the operating system).

- `[OS]-demo`: Builds a demo version of `fuzzylite` which can be executed later as `./fuzzylite`. It shows the results for four examples of Fuzzy Engines.

- `[OS]-static`: Builds the library as a static library (`fuzzylite.a`).

- `[OS]-shared`: Builds the library as a dynamic library (`fuzzylite.so` (unix) or `fuzzylite.dylib` (mac)).

This is a huge improvement from those really nasty NetBeans scripts. These scripts were automatically created by Eclipse,

1.6.2 FuzzyLite v.1.01

Version 1.01 contains the following files `linux-Makefile` and `macosx-Makefile`, and the following folders `linux-nbproject` and `macosx-nbproject`. Depending on your platform, you must rename them by removing the name of the platform, so the new names are `Makefile` and `nbproject` respectively. This should work, otherwise, follow the steps below.

1. Create a `C++` Project either in Eclipse IDE or Netbeans IDE.

2. Add all the source files and include files to the project.

3. Add `.` to the includes path in the project properties.

4. Add the directive `-DFL_USE_LOG` to enable the use of logging via `FL_LOG(message)`. In `./include/defs.h` there are more symbols that can be defined via `-D` for further customization.

5. Decide whether to build a library or an executable (in project properties).

6. Use the IDE's normal build.

1.6.3 Graphic User Interface

1.6.4 Requirements

- `Qt` which can be freely downloaded from http://qt.nokia.com/.

1.6.5 `FuzzyLite v1.02`

In order to build this version, all you need to do (aside from having installed `Qt` which can be freely downloaded from http://qt.nokia.com/.) is execute from `./fuzzylite/` the command `qmake`. This will create a Makefile. Then, run `make` and it should start building the GUI. Notice that in `gui.pro` it links to the `unix` version of `fuzzylite` static library using relative path, so be sure to build such library first. If it is not `unix` what you need, perform your changes accordingly.

1.6.6 `FuzzyLite v.1.01`

In order to build the graphical user interface of `FuzzyLite v1.02`, it is necessary to first install `Qt` which can be freely downloaded from http://qt.nokia.com/.

The `Makefile` included within the `./gui` is quite easy to read. Basically, the most important thing here is to copy the `libfuzzylite.dylib` (or whatever the extension be according to your platform) into the folder `./gui/dist` which is where the executable will be put. An alternative is to copy the library into `/usr/local/lib` and comment those lines in the `Makefile` that build and copy the library into the `./gui/dist` directory.

After configuring the `Makefile` to fit your system, a `make all` from `./gui` would build the graphical user interface of `FuzzyLite v1.02`. To run, it suffices to execute `./gui` from `./gui/dist`.

1.7 Acknowledgements

Chapter 2

The Model

This chapter is devoted to explain `FuzzyLite v1.02` by means of a class diagram based on UML. For a better comprehension, it is divided in five groups: Fuzzy operations, Linguistic variables and terms, Linguistic terms, Fuzzy rules, Fuzzy engine, and Fuzzy exceptions. It is important to mention that all classes related to `FuzzyLite v1.02` are inside the namespace `fl`.

2.1   Fuzzy operations

Figure 2.1 shows the class diagram for this group. The classes that can be seen in it are briefly explained in the following sections.

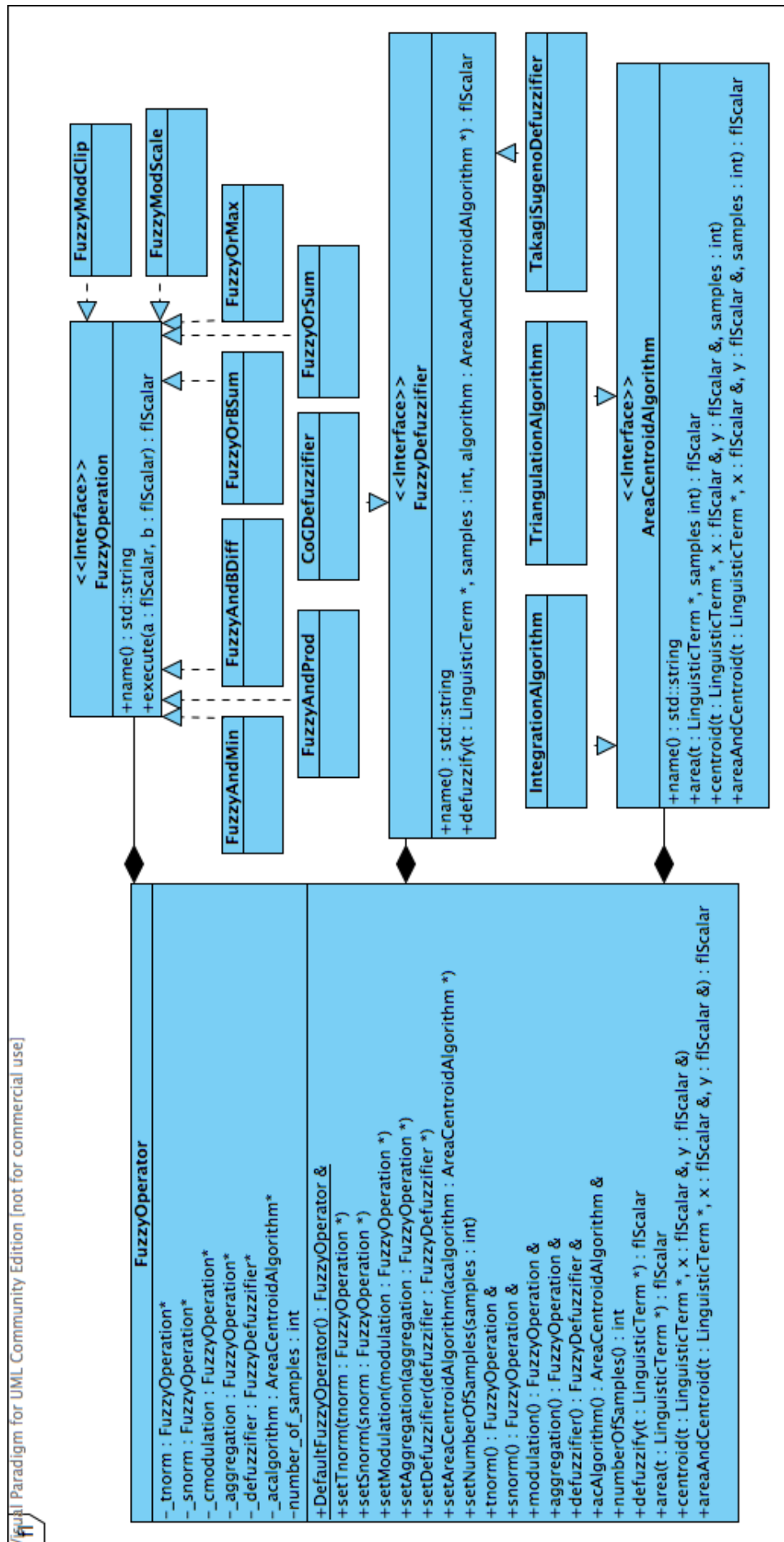Figure 2.1: Class diagram: Fuzzy operations

2.1.1 `FuzzyOperator`

`FuzzyOperator` centralizes all the operations that can be performed in the fuzzy system. It contains the type of T-Norms, S-Norms, modulation, aggregation, defuzzification methods, algorithms for computing the area and centroid of any linguistic term, and the number of samples that are drawn from any linguistic term to be used by the algorithm.

This class has an static default `FuzzyOperator` instance that can be obtained using the method `fl::FuzzyOperator::DefaultFuzzyOperator()` anywhere and anytime. The defaults for this operator are the following

- **T-Norm:** `FuzzyAndMin`.
- **S-Norm:** `FuzzyOrMax`.
- **Modulation:** `FuzzyModClip`.
- **Aggregation:** `FuzzyOrMax`.

- **Defuzzifier:** `CoGDefuzzifier`.
- **Algorithm:** `TriangulationAlgorithm`.
- **Number of samples:** 100.

These defaults may be changed at any moment from anywhere, but consider that this is an instance that is shared among all instances from many classes, so be careful about changing these values. Nevertheless, if you need to, you may use different instances among all those classes composed by `FuzzyOperator`.

2.1.2 `FuzzyOperation`

This is the interface shared by all T-Norms, S-Norms, and methods for modulation and aggregation. If you want to implement your own operations, you may do so by implementing this interface. The operations included in `FuzzyLite v1.02` are:

- T-Norm: minimum (`FuzzyAndMin`), product (`FuzzyAndProd`), and bounded difference (`FuzzyAndBDiff`).

- S-Norm: maximum (`FuzzyOrMax`), sum (`FuzzyOrSum`), bounded sum (`FuzzyOrBSum`).

- Modulation: clipping (`FuzzyModClip`), scaling (`FuzzyModScale`).

- Aggregation: maximum (`FuzzyOrMax`), sum (`FuzzyOrSum`), bounded sum (`FuzzyOrBSum`).

2.1.3 `FuzzyDefuzzifier`

This is the interface shared by all defuzzifiers. If you want to implement a different defuzzifier, you may do so by implementing this interface. The defuzzifiers included in `FuzzyLite v1.02` are: Centre of Gravity (`CoGDefuzzifier`) for Mamdani rules, and `TakagiSugenoDefuzzifier` for Takagi-Sugeno rules.

2.1.4 `AreaAndCentroidAlgorithm`

This is an interface used for computing the area and centroid of linguistic terms. The algorithms included in `FuzzyLite v1.02` are: `TriangulationAlgorithm` which is a triangulation algorithm appropriate for terms that can be easily triangulated, and `IntegrationAlgorithm` which is a regular integration algorithm. The algorithm should be chosen according to the shape of the fuzzy partitions. You may also create your own algorithm by implementing this interface.

2.2 Linguistic variables and terms

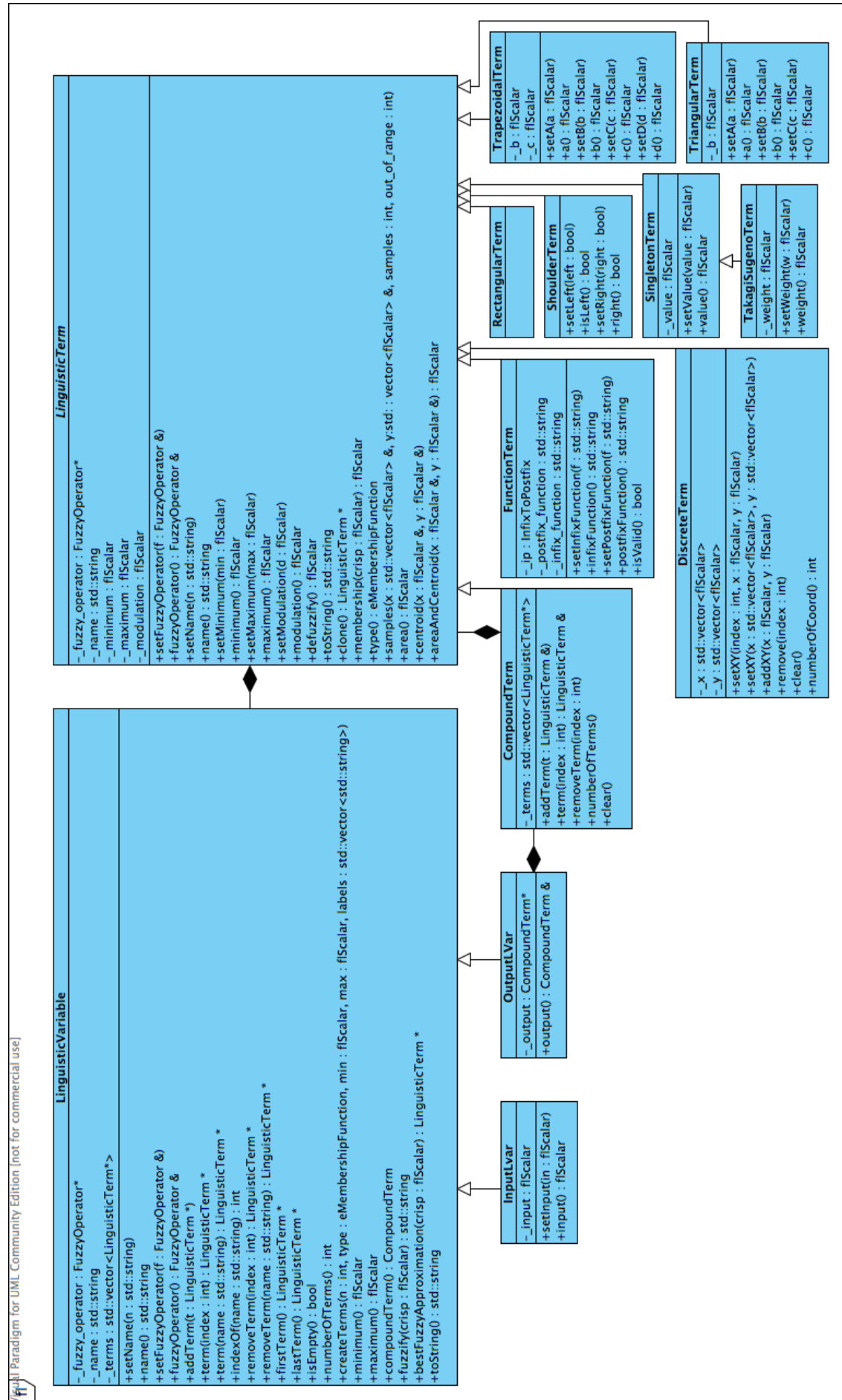Figure 2.2 shows the class diagram for the linguistic variables and terms included in `FuzzyLite v1.02`.

Visual Paradigm for UML Community Edition [not for commercial use]

LinguisticVariable

- _fuzzy_operator : FuzzyOperator*
- _name : std::string
- _terms : std::vector<LinguisticTerm*>

+setName(n : std::string)
+name() : std::string
+setFuzzyOperator(f : FuzzyOperator &)
+fuzzyOperator() : FuzzyOperator &
+addTerm(t : LinguisticTerm *)
+term(index : int) : LinguisticTerm *
+term(name : std::string) : LinguisticTerm *
+indexOf(name : std::string) : int
+removeTerm(index : int) : LinguisticTerm *
+removeTerm(name : std::string) : LinguisticTerm *
+firstTerm() : LinguisticTerm *
+lastTerm() : LinguisticTerm *
+isEmpty() : bool
+numberOfTerms() : int
+createTerms(n : int, type : eMembershipFunction, min : flScalar, max : flScalar, labels : std::vector<std::string>)
+minimum() : flScalar
+maximum() : flScalar
+compoundTerm() : CompoundTerm
+fuzzify(crisp : flScalar) : std::string
+bestFuzzyApproximation(crisp : flScalar) : LinguisticTerm *
+toString() : std::string

LinguisticTerm

- _fuzzy_operator : FuzzyOperator*
- _name : std::string
- _minimum : flScalar
- _maximum : flScalar
- _modulation : flScalar

+setFuzzyOperator(f : FuzzyOperator &)
+fuzzyOperator() : FuzzyOperator &
+setName(n : std::string)
+name() : std::string
+setMinimum(min : flScalar)
+minimum() : flScalar
+setMaximum(max : flScalar)
+maximum() : flScalar
+setModulation(d : flScalar)
+modulation() : flScalar
+defuzzify() : flScalar
+toString() : std::string
+clone() : LinguisticTerm *
+membership(crisp : flScalar) : flScalar
+type() : eMembershipFunction
+samples(x : std::vector<flScalar> &, y:std::: vector<flScalar> &, samples : int, out_of_range : int)
+area() : flScalar
+centroid(x : flScalar &, y : flScalar &)
+areaAndCentroid(x : flScalar &, y : flScalar &) : flScalar

InputLvar

- _input : flScalar

+setInput(in : flScalar)
+input() : flScalar

OutputLVar

- _output : CompoundTerm*

+output() : CompoundTerm &

CompoundTerm

- _terms : std::vector<LinguisticTerm*>

+addTerm(t : LinguisticTerm &)
+term(index : int) : LinguisticTerm &
+removeTerm(index : int)
+numberOfTerms()
+clear()

DiscreteTerm

- _x : std::vector<flScalar>
- _y : std::vector<flScalar>

+setXY(index : int, x : flScalar, y : flScalar)
+setXY(x : std::vector<flScalar>, y : std::vector<flScalar>)
+addXY(x : flScalar, y : flScalar)
+remove(index : int)
+clear()
+numberOfCoord() : int

FunctionTerm

- _ip : InfixToPostfix
- _postfix_function : std::string
- _infix_function : std::string

+setInfixFunction(f : std::string)
+infixFunction() : std::string
+setPostfixFunction(f : std::string)
+postfixFunction() : std::string
+isValid() : bool

RectangularTerm

ShoulderTerm

+setLeft(left : bool)
+isLeft() : bool
+setRight(right : bool)
+right() : bool

SingletonTerm

- _value : flScalar

+setValue(value : flScalar)
+value() : flScalar

TakagiSugenoTerm

- _weight : flScalar

+setWeight(w : flScalar)
+weight() : flScalar

TrapezoidalTerm

- _b : flScalar
- _c : flScalar

+setA(a : flScalar)
+a() : flScalar
+setB(b : flScalar)
+b() : flScalar
+setC(c : flScalar)
+c() : flScalar
+setD(d : flScalar)
+d() : flScalar

TriangularTerm

- _b : flScalar

+setA(a : flScalar)
+a() : flScalar
+setB(b : flScalar)
+b() : flScalar
+setC(c : flScalar)
+c() : flScalar

Figure 2.2: Class diagram: Linguistic variables and terms

2.2.1 `LinguisticVariable`

Linguistic variables are composed of a vector of `LinguisticTerm`s. This is an abstract class that is used as base for input variables (`InputLVar`) and output variables (`OutputLVar`). Each input variable has an input value that should be the input received by the system, and each output variable has a compound linguistic term composed by all the linguistic terms added by the activation of the different rules given the input values of the input variables and their respective processing.

2.2.2 `LinguisticTerm`

Linguistic terms, also known as fuzzy partitions, define the shape of each label. Each linguistic term has a minimum and maximum that are used to define the limits of the area it covers. Several linguistic terms are included in `FuzzyLite v1.02`:

- `TriangularTerm`: is the well known triangular term, it is defined by its three vertices a, b, and c, where a and c are wrappers of `minimum()` and `maximum()`, respectively.

- `RectangularTerm`: is a simple rectangular term delimited by the minimum and maximum of its parent class `LinguisticTerm`.

- `TrapezoidalTerm`: defines a trapezoid by its four vertices a, b, c, and d, where a and d are wrappers of `minimum()` and `maximum()`, respectively.

- `SingletonTerm`: defines a singleton for the value v delimited by $v - \delta_{low}$ and $v + \delta_{hi}$ where δ_{low} and δ_{hi} are very small values in order to be able to create samples of this term.

- `ShoulderTerm`: defines a trapezoid that extends to $\pm\infty$ depending on whether it is left $(-\infty)$ or right $(+\infty)$.

- `DiscreteTerm`: is a term composed of a vector of (x, y) coordinates. It is important to note that when sampling this type of term, the result of `membership(flScalar crisp)` is the value of y for the closest x given the value of `crisp`.

- `FunctionTerm`: is a term that is defined as a function $f(x)$ which accepts several mathematical and trigonometrical operations over a given function, for example: `setInfixFunction( "(sin x) / x" )`. It is *very* important to remark that the parser used to parse infix functions may not function as expected (e.g. $\sin(x)/x$ is different of $(\sin x)/x$, the latter one gives the result one expects). When in doubt, test your expression by converting it to postfix.

- `CompoundTerm`: is a term composed of a list of `LinguisticTerm`s. This is used in `OutputLVar` in order to aggregate all the linguistic terms that were added by the activation of rules, but it may be used as well to create more complex partitions.

- `TakagiSugenoTerm`: is a term used when dealing with `TakagiSugenoRule`s, it extends from `SingletonTerm` so it has a value, and it includes a `weight` as well, both necessary for this kind of system.

Finally, figure 2.3 shows examples of the linguistic terms, obtained by printing the window from the GUI developed for `FuzzyLite v1.02`.

2.3 Fuzzy rules

Figure 2.4 shows the class diagram for fuzzy rules and all its related classes included in `FuzzyLite v1.02`.
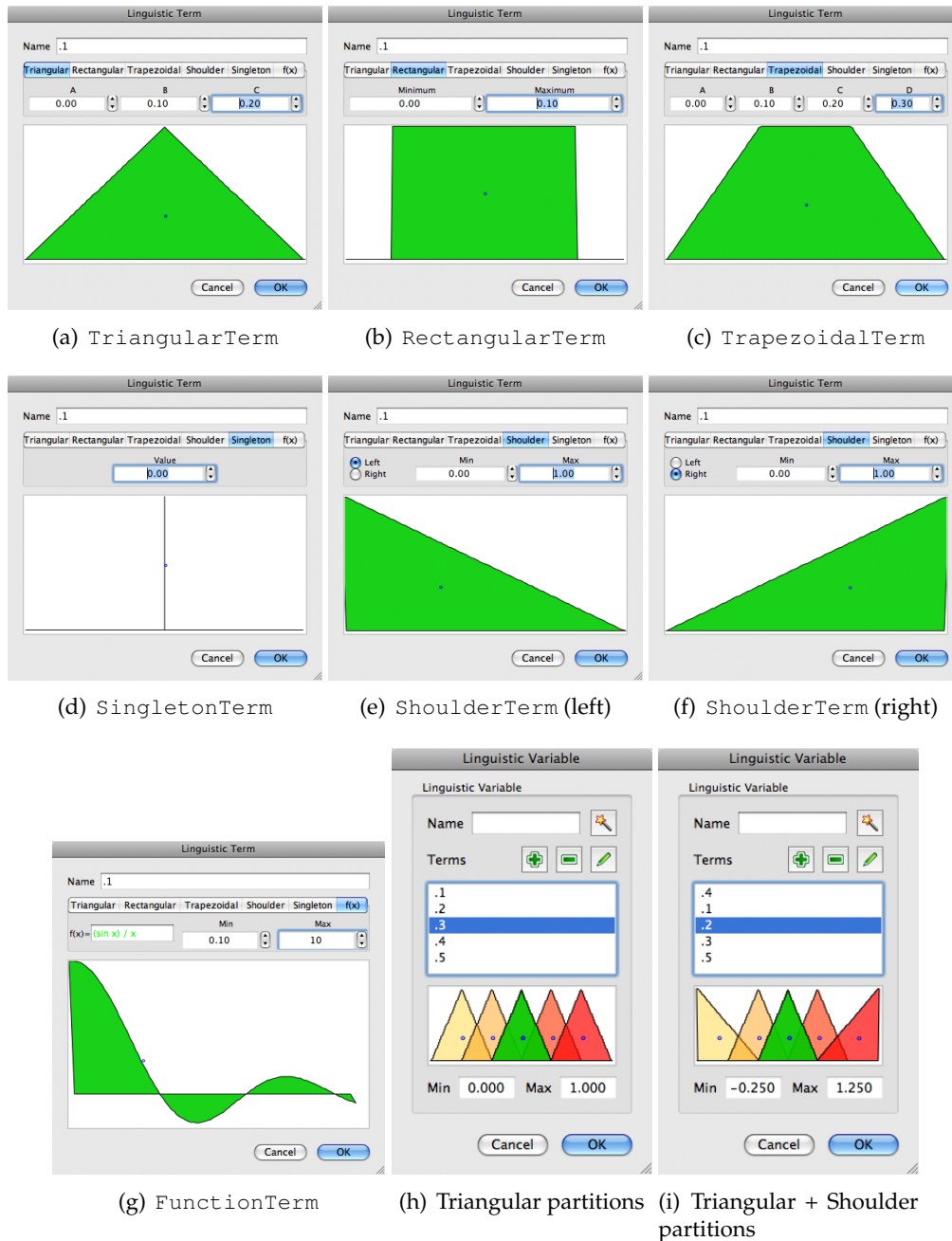
(a) `TriangularTerm`   (b) `RectangularTerm`   (c) `TrapezoidalTerm`

(d) `SingletonTerm`   (e) `ShoulderTerm` (left)   (f) `ShoulderTerm` (right)

(g) `FunctionTerm`   (h) Triangular partitions   (i) Triangular + Shoulder partitions

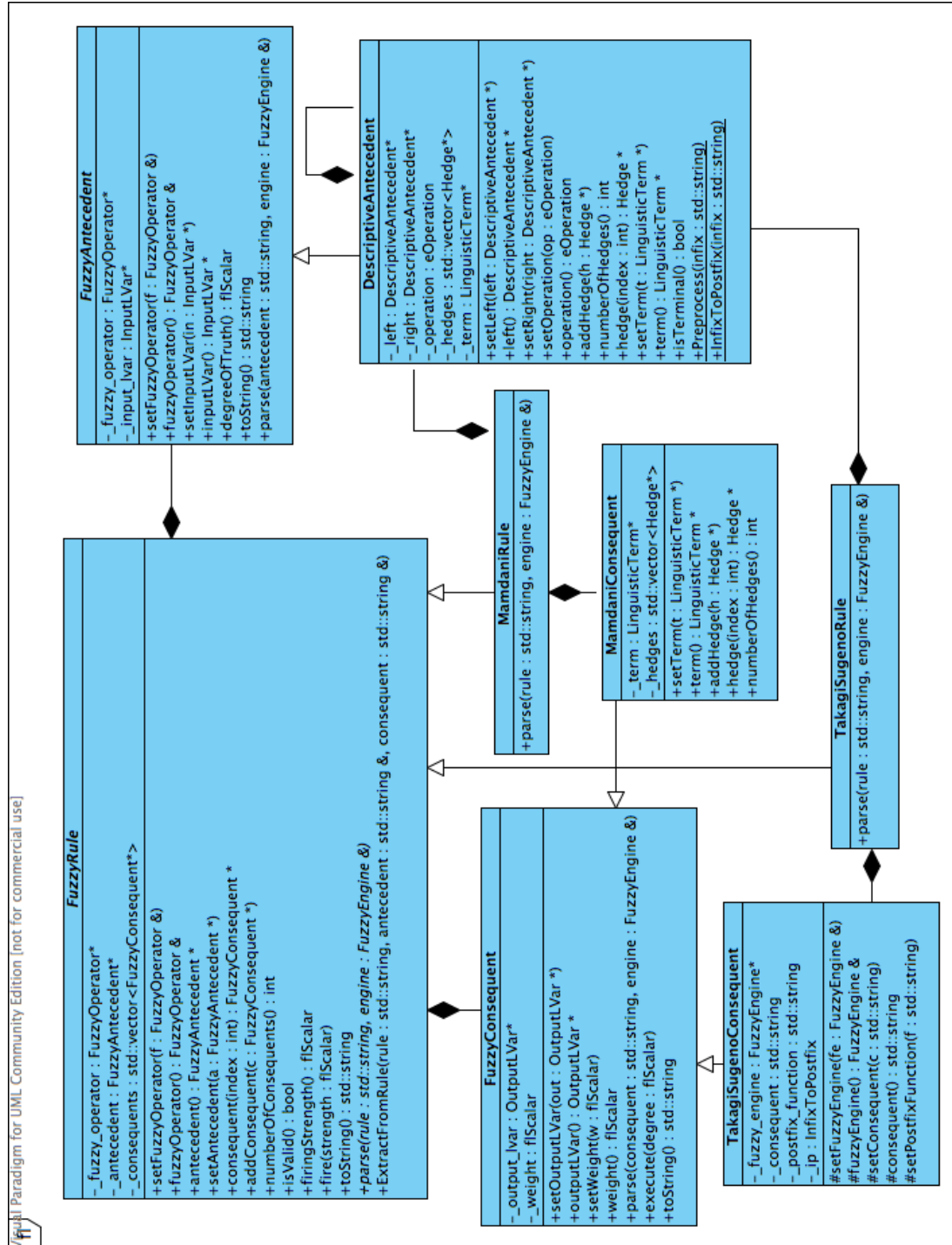Figure 2.3: Linguistic Terms

Figure 2.4: Class diagram: Fuzzy rules

2.3.1 `FuzzyRule`

This abstract class is the base for `MamdaniRule`s and `TakagiSugenoRule`s. It is composed by an antecedent (`FuzzyAntecedent`) and a list of consequents (`FuzzyConsequent`s). There are two methods that are worth mentioning here: `firingStrength()` which determines the activation degree of the rule given the inputs in the `InputLVar`s, and `fire( flScalar d )` which fires the rule with a degree d. When firing the rule, the consequents are modulated according to d and then added to the `OutputLVar`s with the respective weight of the rule. It is also important to remark that parsing the rules is case-sensitive.

`FuzzyAntecedent` and `FuzzyConsequent`

These abstract classes represent the antecedent and consequent of a rule, respectively. `FuzzyAntecedent` has a pointer to the input variable to which it is related in order to access the value of the respective `InputLVar`, similarly, `FuzzyConsequent` has a pointer to the output variable to which it is related in order to aggregate the respective modulated linguistic term to the output.

`DescriptiveAntecedent`

This class is based on `FuzzyAntecedent` and it is a red-black tree that relates two propositions with an operator, for example, `input-1` **is** `LOW` **and** `input-2` **is** `GOOD` is separated into a left antecedent (`input-1` **is** `LOW`), an operator (**and**), and a right antecedent (`input-2` **is** `GOOD`). Given the recursion of this model, it is possible to create rules of any depth, as it is evaluated bottom-up. This is the class used by `MamdaniRule` and `TakagiSugenoRule`.

`Hedge`

Hedges are modifiers of the propositions (antecedent) and actions (consequent). All hedges must implement the interface `Hedge`, and must be included when configuring the `FuzzyEngine`. `FuzzyLite v1.02` includes four hedges: not (`HedgeNot`), somewhat (`HedgeSomewhat`), very (`HedgeVery`), any (`HedgeAny`). In order to use `HedgeAny`, it is necessary to include a *dummy* linguistic term to comply with the general form of the rules (i.e. *input-1* **is** `any LOW`); this hedge will always return 1.0, so it does not matter the linguistic term as long as it is within the linguistic variable used. Figure 2.5 shows the class diagram regarding hedges.

`MamdaniRule` and `TakagiSugenoRule`

These classes represents the type of rules of a system based on Mamdani's rules, and the one based on Takagi-Sugeno rules, respectively. They extend the abstract class `FuzzyRule` by implementing the `parse()` method accordingly. Both classes use `DescriptiveAntecedent` as the antecedent of the rule, but differ in the consequent: `MamdaniRule` uses `MamdaniConsequent`, and `TakagiSugenoRule` uses `TakagiSugenoConsequent`.

`MamdaniConsequent` and `TakagiSugenoConsequent`

The consequent of a Mamdani rule is of the form: `OutputLVar` **is** [`Hedge`*] `LinguisticTerm`, where [`Hedge`*] means that none or many hedges may be included.

   Conversely, the consequent of a Takagi-Sugeno rule is of the form `OutputLVar`={*expression*}, where *expression* is a mathematical expression that may include references to the values of the `InputLVar`s, in which case the name of the input variable is used (e.g. "... f_x=0.5 * input-1 + (sin input-2)"). This implementation also allows to include previously computed outputs, in which case the name of the output should be used instead.
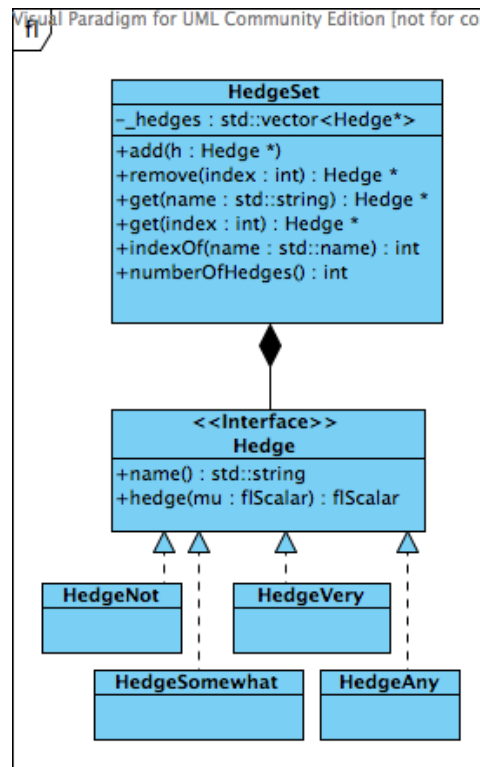
Figure 2.5: Class diagram: Hedges

2.4   Fuzzy engine

The class diagram for this group can be seen in figure 2.6. It contains the class `FuzzyEngine` which is composed by a `FuzzyOperator`, a vector of `RuleBlock`s which contain the `FuzzyRule`s, a `HedgeSet` that contains the hedges registered in the engine, and a vector of `InputLVar`s and `OutputLVar`s for input and output variables (respectively).

The `FuzzyEngine` class contains the whole fuzzy control system. The only methods which might require a bit of explanation are `process( bool )`, and `process(int, bool)`. The former receives a boolean parameter that defaults to `true` and (if true) clears the output of all the output variables. The latter receives an `int` parameter that determines the index of the `RuleBlock` to be fired, and a `bool` that defaults to `true` which defines whether to clear the output of the output variables.
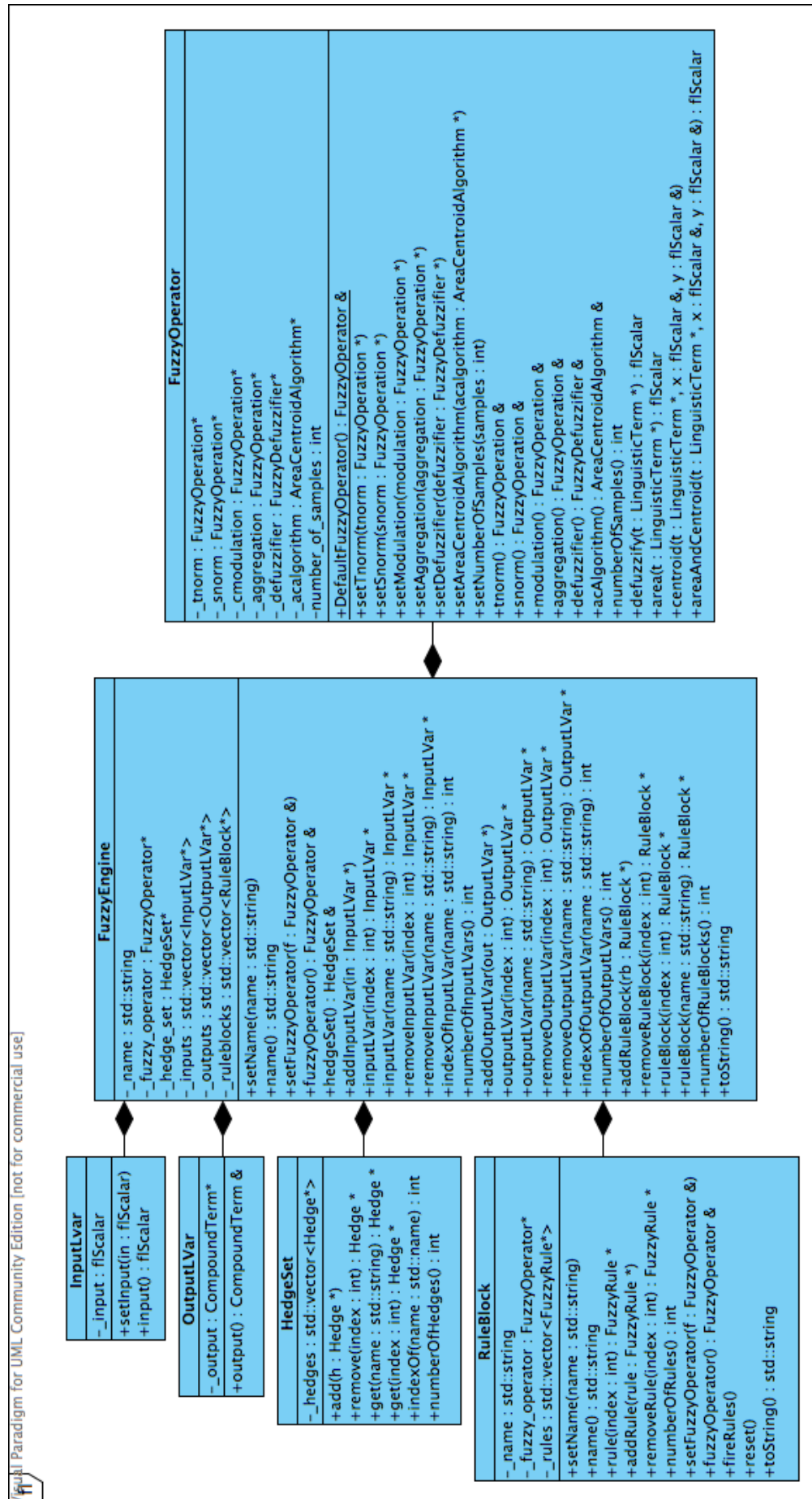
FuzzyOperator

- _tnorm : FuzzyOperation*
- _snorm : FuzzyOperation*
- _cmodulation : FuzzyOperation*
- _aggregation : FuzzyOperation*
- _defuzzifier : FuzzyDefuzzifier*
- _acalgorithm : AreaCentroidAlgorithm*
- number_of_samples : int

+ DefaultFuzzyOperator() : FuzzyOperator &
+ setTnorm(tnorm : FuzzyOperation *)
+ setSnorm(snorm : FuzzyOperation *)
+ setModulation(modulation : FuzzyOperation *)
+ setAggregation(aggregation : FuzzyOperation *)
+ setDefuzzifier(defuzzifier : FuzzyDefuzzifier *)
+ setAreaCentroidAlgorithm(acalgorithm : AreaCentroidAlgorithm *)
+ setNumberOfSamples(samples : int)
+ tnorm() : FuzzyOperation &
+ snorm() : FuzzyOperation &
+ modulation() : FuzzyOperation &
+ aggregation() : FuzzyOperation &
+ defuzzifier() : FuzzyDefuzzifier &
+ acAlgorithm() : AreaCentroidAlgorithm &
+ numberOfSamples() : int
+ defuzzify(t : LinguisticTerm *) : flScalar
+ area(t : LinguisticTerm *) : flScalar
+ centroid(t : LinguisticTerm *, x : flScalar &, y : flScalar &)
+ areaAndCentroid(t : LinguisticTerm *, x : flScalar &, y : flScalar &) : flScalar

FuzzyEngine

- _name : std::string
- _fuzzy_operator : FuzzyOperator*
- _hedge_set : HedgeSet*
- _inputs : std::vector<InputLVar*>
- _outputs : std::vector<OutputLVar*>
- _ruleblocks : std::vector<RuleBlock*>

+ setName(name : std::string)
+ name() : std::string
+ setFuzzyOperator(f : FuzzyOperator &)
+ fuzzyOperator() : FuzzyOperator &
+ hedgeSet() : HedgeSet &
+ addInputLVar(in : InputLVar *)
+ inputLVar(index : int) : InputLVar *
+ inputLVar(name : std::string) : InputLVar *
+ removeInputLVar(index : int) : InputLVar *
+ removeInputLVar(name : std::string) : InputLVar *
+ indexOfInputLVar(name : std::string) : int
+ numberOfInputLVars() : int
+ addOutputLVar(out : OutputLVar *)
+ outputLVar(index : int) : OutputLVar *
+ outputLVar(name : std::string) : OutputLVar *
+ removeOutputLVar(index : int) : OutputLVar *
+ removeOutputLVar(name : std::string) : OutputLVar *
+ indexOfOutputLVar(name : std::string) : int
+ numberOfOutputLVars() : int
+ addRuleBlock(rb : RuleBlock *)
+ removeRuleBlock(index : int) : RuleBlock *
+ ruleBlock(index : int) : RuleBlock *
+ ruleBlock(name : std::string) : RuleBlock *
+ numberOfRuleBlocks() : int
+ toString() : std::string

InputLvar

- _input : flScalar

+ setInput(in : flScalar)
+ input() : flScalar

OutputVar

- _output : CompoundTerm*

+ output() : CompoundTerm &

HedgeSet

- _hedges : std::vector<Hedge*>

+ add(h : Hedge *)
+ remove(index : int) : Hedge *
+ get(name : std::string) : Hedge *
+ get(index : int) : Hedge *
+ indexOf(name : std::name) : int
+ numberOfHedges() : int

RuleBlock

- _name : std::string
- _fuzzy_operator : FuzzyOperator*
- _rules : std::vector<FuzzyRule*>

+ setName(name : std::string)
+ name() : std::string
+ rule(index : int) : FuzzyRule *
+ addRule(rule : FuzzyRule *)
+ removeRule(index : int) : FuzzyRule *
+ numberOfRules() : int
+ setFuzzyOperator(f : FuzzyOperator &)
+ fuzzyOperator() : FuzzyOperator &
+ fireRules()
+ reset()
+ toString() : std::string

Figure 2.6: Class diagram: Fuzzy engine

2.5 Fuzzy exceptions

This group contains some exceptions that are used within several classes of `FuzzyLite v1.02`. The class `FuzzyException` extends the `std::exception` of the Standard Template Library (STL), and adds additional methods. The other classes derived from `FuzzyException` also contain some static methods to help a bit with the programming. There is not much to say about these exceptions, except to take a look at the code when using them to become familiar. Figure 2.7 shows the class diagram for this group.
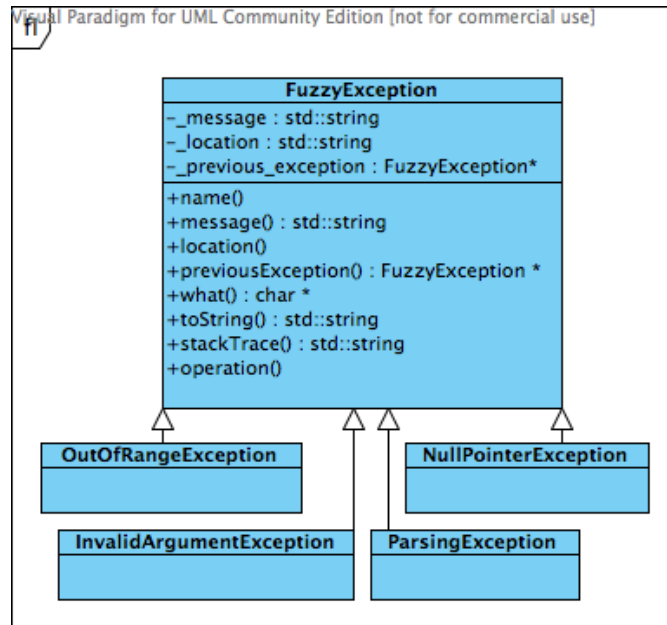
Figure 2.7: Class diagram: Fuzzy exceptions

Chapter 3

Examples

3.1 Example #1: Basic FIS

This is a basic FIS composed of one input variable and one output variable.

```cpp
fl::FuzzyEngine engine;
engine.hedgeSet().add(new fl::HedgeNot);
engine.hedgeSet().add(new fl::HedgeSomewhat);
engine.hedgeSet().add(new fl::HedgeVery);
fl::InputLVar* energy = new fl::InputLVar("Energy");
energy->addTerm(new fl::ShoulderTerm("LOW", 0.25, 0.5, true));
energy->addTerm(new fl::TriangularTerm("MEDIUM", 0.25, 0.75));
energy->addTerm(new fl::ShoulderTerm("HIGH", 0.50, 0.75, false));
engine.addInputLVar(energy);

fl::OutputLVar* health = new fl::OutputLVar("Health");
health->addTerm(new fl::TriangularTerm("BAD", 0.0, 0.50));
health->addTerm(new fl::TriangularTerm("REGULAR", 0.25, 0.75));
health->addTerm(new fl::TriangularTerm("GOOD", 0.50, 1.00));
engine.addOutputLVar(health);
fl::RuleBlock* block = new fl::RuleBlock();
block->addRule(new fl::MamdaniRule("if Energy is LOW then Health is BAD", engine));
block->addRule(new fl::MamdaniRule("if Energy is MEDIUM then Health is REGULAR", engine));
block->addRule(new fl::MamdaniRule("if Energy is HIGH then Health is GOOD", engine));
engine.addRuleBlock(block);
```

Once the FIS is configured, the control process may begin anytime by setting the input value to the input variables and processing. For example,

```cpp
for (fl::flScalar in = 0.0; in < 1.1; in += 0.1) {
    energy->setInput(in);
    engine.process();
    fl::flScalar out = health->output().defuzzify();
    FL_LOG("Energy=" << in);
    FL_LOG("Energy is " << energy->fuzzify(in));
    FL_LOG("Health=" << out);
    FL_LOG("Health is " << health->fuzzify(out));
    FL_LOG("--");
}
```

The previous code would yield the following results in console (assuming that FL_USE_LOG was defined):

```
Energy=0
Energy is 1.000/LOW + 0.000/MEDIUM + 0.000/HIGH
Health=0.249902
Health is 1.000/BAD + 0.000/REGULAR + 0.000/GOOD
--
Energy=0.1
```

```
7 Energy is 1.000/LOW + 0.000/MEDIUM + 0.000/HIGH
8 Health=0.249902
9 Health is 1.000/BAD + 0.000/REGULAR + 0.000/GOOD
10 --
11 Energy=0.2
12 Energy is 1.000/LOW + 0.000/MEDIUM + 0.000/HIGH
13 Health=0.249902
14 Health is 1.000/BAD + 0.000/REGULAR + 0.000/GOOD
15 --
16 Energy=0.3
17 Energy is 0.800/LOW + 0.200/MEDIUM + 0.000/HIGH
18 Health=0.309985
19 Health is 0.760/BAD + 0.240/REGULAR + 0.000/GOOD
20 --
21 Energy=0.4
22 Energy is 0.400/LOW + 0.600/MEDIUM + 0.000/HIGH
23 Health=0.394929
24 Health is 0.420/BAD + 0.580/REGULAR + 0.000/GOOD
25 --
26 Energy=0.5
27 Energy is 0.000/LOW + 1.000/MEDIUM + 0.000/HIGH
28 Health=0.499902
29 Health is 0.000/BAD + 1.000/REGULAR + 0.000/GOOD
30 --
31 Energy=0.6
32 Energy is 0.000/LOW + 0.600/MEDIUM + 0.400/HIGH
33 Health=0.604537
34 Health is 0.000/BAD + 0.582/REGULAR + 0.418/GOOD
35 --
36 Energy=0.7
37 Energy is 0.000/LOW + 0.200/MEDIUM + 0.800/HIGH
38 Health=0.689444
39 Health is 0.000/BAD + 0.242/REGULAR + 0.758/GOOD
40 --
41 Energy=0.8
42 Energy is 0.000/LOW + 0.000/MEDIUM + 1.000/HIGH
43 Health=0.749902
44 Health is 0.000/BAD + 0.000/REGULAR + 1.000/GOOD
45 --
46 Energy=0.9
47 Energy is 0.000/LOW + 0.000/MEDIUM + 1.000/HIGH
48 Health=0.749902
49 Health is 0.000/BAD + 0.000/REGULAR + 1.000/GOOD
50 --
51 Energy=1
52 Energy is 0.000/LOW + 0.000/MEDIUM + 1.000/HIGH
53 Health=0.749902
54 Health is 0.000/BAD + 0.000/REGULAR + 1.000/GOOD
55 --
```

3.2 Example #2: 3D Pole Balancing

A simulation video of this fuzzy system implemented with `FuzzyLite v1.02` as shown below is available at http://www.youtube.com/watch?v=YOKk8G_5aRA.

```
1 FuzzyEngine engine("pole-balancing-3d");
2
3 InputLVar* anglex = new InputLVar("AngleX");
4 std::vector<std::string> labels;
5 labels.push_back("NEAR_0");
6 labels.push_back("NEAR_45");
7 labels.push_back("NEAR_90");
```

```cpp
 8  labels.push_back("NEAR_135");
 9  labels.push_back("NEAR_180");
10  anglex->createTerms(5, LinguisticTerm::MF_SHOULDER, 0, 180, labels);
11  engine.addInputLVar(anglex);
12
13  InputLVar* anglez = new InputLVar("AngleZ");
14  labels.clear();
15  labels.push_back("NEAR_0");
16  labels.push_back("NEAR_45");
17  labels.push_back("NEAR_90");
18  labels.push_back("NEAR_135");
19  labels.push_back("NEAR_180");
20  anglez->createTerms(5, LinguisticTerm::MF_SHOULDER, 0, 180, labels);
21  engine.addInputLVar(anglez);
22
23  OutputLVar* forcex = new OutputLVar("ForceX");
24  labels.clear();
25  labels.push_back("NL");
26  labels.push_back("NS");
27  labels.push_back("ZR");
28  labels.push_back("PS");
29  labels.push_back("PL");
30  forcex->createTerms(5, LinguisticTerm::MF_TRIANGULAR, -1, 1, labels);
31  engine.addOutputLVar(forcex);
32
33  OutputLVar* forcez = new OutputLVar("ForceZ");
34  labels.clear();
35  labels.push_back("NL");
36  labels.push_back("NS");
37  labels.push_back("ZR");
38  labels.push_back("PS");
39  labels.push_back("PL");
40  forcez->createTerms(5, LinguisticTerm::MF_TRIANGULAR, -1, 1, labels);
41  engine.addOutputLVar(forcez);
42
43  RuleBlock* ruleblock = new RuleBlock("Rules");
44  ruleblock->addRule(new MamdaniRule("if AngleX is NEAR_180 then ForceX is NL", engine));
45  ruleblock->addRule(new MamdaniRule("if AngleX is NEAR_135 then ForceX is NS", engine));
46  ruleblock->addRule(new MamdaniRule("if AngleX is NEAR_90 then ForceX is ZR", engine));
47  ruleblock->addRule(new MamdaniRule("if AngleX is NEAR_45 then ForceX is PS", engine));
48  ruleblock->addRule(new MamdaniRule("if AngleX is NEAR_0 then ForceX is PL", engine));
49
50  ruleblock->addRule(new MamdaniRule("if AngleZ is NEAR_180 then ForceZ is NL", engine));
51  ruleblock->addRule(new MamdaniRule("if AngleZ is NEAR_135 then ForceZ is NS", engine));
52  ruleblock->addRule(new MamdaniRule("if AngleZ is NEAR_90 then ForceZ is ZR", engine));
53  ruleblock->addRule(new MamdaniRule("if AngleZ is NEAR_45 then ForceZ is PS", engine));
54  ruleblock->addRule(new MamdaniRule("if AngleZ is NEAR_0 then ForceZ is PL", engine));
55  engine.addRuleBlock(ruleblock);
56
57  FL_LOG(engine.toString());
```

The output from the previous block of code exports the fuzzy system to text using the Fuzzy Controller Language (FCL) as shown below.

```
 1  FUNCTION_BLOCK pole-balancing-3d
 2
 3  VAR_INPUT
 4  AngleX: REAL;
 5  AngleZ: REAL;
 6  END_VAR
 7
 8  FUZZIFY AngleX
 9  TERM NEAR_0 := Shoulder left(0 60);
10  TERM NEAR_45 := Triangular (30 60 90);
```

```
11  TERM NEAR_90 := Triangular (60 90 120);
12  TERM NEAR_135 := Triangular (90 120 150);
13  TERM NEAR_180 := Shoulder right(120 180);
14  END_FUZZIFY
15
16  FUZZIFY AngleZ
17  TERM NEAR_0 := Shoulder left(0 60);
18  TERM NEAR_45 := Triangular (30 60 90);
19  TERM NEAR_90 := Triangular (60 90 120);
20  TERM NEAR_135 := Triangular (90 120 150);
21  TERM NEAR_180 := Shoulder right(120 180);
22  END_FUZZIFY
23
24  VAR_OUTPUT
25  ForceX: REAL
26  ForceZ: REAL
27  END_VAR
28
29  DEFUZZIFY ForceX
30  TERM NL := Triangular (-1 -0.666667 -0.333333);
31  TERM NS := Triangular (-0.666667 -0.333333 5.96046e-08);
32  TERM ZR := Triangular (-0.333333 5.96046e-08 0.333333);
33  TERM PS := Triangular (5.96046e-08 0.333333 0.666667);
34  TERM PL := Triangular (0.333333 0.666667 1);
35  END_DEFUZZIFY
36
37  DEFUZZIFY ForceZ
38  TERM NL := Triangular (-1 -0.666667 -0.333333);
39  TERM NS := Triangular (-0.666667 -0.333333 5.96046e-08);
40  TERM ZR := Triangular (-0.333333 5.96046e-08 0.333333);
41  TERM PS := Triangular (5.96046e-08 0.333333 0.666667);
42  TERM PL := Triangular (0.333333 0.666667 1);
43  END_DEFUZZIFY
44
45  RULEBLOCK Rules
46  RULE 1: if AngleX is NEAR_180 then ForceX is NL;
47  RULE 2: if AngleX is NEAR_135 then ForceX is NS;
48  RULE 3: if AngleX is NEAR_90 then ForceX is ZR;
49  RULE 4: if AngleX is NEAR_45 then ForceX is PS;
50  RULE 5: if AngleX is NEAR_0 then ForceX is PL;
51  RULE 6: if AngleZ is NEAR_180 then ForceZ is NL;
52  RULE 7: if AngleZ is NEAR_135 then ForceZ is NS;
53  RULE 8: if AngleZ is NEAR_90 then ForceZ is ZR;
54  RULE 9: if AngleZ is NEAR_45 then ForceZ is PS;
55  RULE 10: if AngleZ is NEAR_0 then ForceZ is PL;
56  END_RULEBLOCK
57
58  END_FUNCTION_BLOCK
```

3.3   Example 3: Approximating a function

It is also possible to create a fuzzy system to approximate a function. For example, if we were to approximate the function $sin(x)/x$, we could do so by using the with the following code.

```
1  FuzzyOperator& op = FuzzyOperator::DefaultFuzzyOperator();
2  op.setDefuzzifier(new TakagiSugenoDefuzzifier);
3  FuzzyEngine engine("approximation", op);
4
5  fl::InputLVar* x = new fl::InputLVar("x");
6  x->addTerm(new fl::TriangularTerm("NEAR_1", 0, 2));
7  x->addTerm(new fl::TriangularTerm("NEAR_2", 1, 3));
8  x->addTerm(new fl::TriangularTerm("NEAR_3", 2, 4));
```

```
9  x->addTerm(new fl::TriangularTerm("NEAR_4", 3, 5));
10 x->addTerm(new fl::TriangularTerm("NEAR_5", 4, 6));
11 x->addTerm(new fl::TriangularTerm("NEAR_6", 5, 7));
12 x->addTerm(new fl::TriangularTerm("NEAR_7", 6, 8));
13 x->addTerm(new fl::TriangularTerm("NEAR_8", 7, 9));
14 x->addTerm(new fl::TriangularTerm("NEAR_9", 8, 10));
15 engine.addInputLVar(x);
16
17 fl::OutputLVar* f_x = new fl::OutputLVar("f_x");
18 f_x->addTerm(new fl::FunctionTerm("function", "(sin x) / x", 0, 10));
19 engine.addOutputLVar(f_x);
20
21 fl::RuleBlock* block = new fl::RuleBlock();
22 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_1 then f_x=0.84", engine));
23 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_2 then f_x=0.45", engine));
24 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_3 then f_x=0.04", engine));
25 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_4 then f_x=-0.18", engine));
26 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_5 then f_x=-0.19", engine));
27 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_6 then f_x=-0.04", engine));
28 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_7 then f_x=0.09", engine));
29 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_8 then f_x=0.12", engine));
30 block->addRule(new fl::TakagiSugenoRule("if x is NEAR_9 then f_x=0.04", engine));
31
32 engine.addRuleBlock(block);
33
34 int n = 40;
35 flScalar mse = 0;
36 for (fl::flScalar in = x->minimum(); in < x->maximum() ;
37         in += (x->minimum() + x->maximum()) / n) {
38     x->setInput(in);
39     engine.process();
40     flScalar expected = f_x->term(0)->membership(in);
41     flScalar obtained = f_x->output().defuzzify();
42     flScalar se = (expected - obtained) * (expected - obtained);
43     mse += isnan(se) ? 0 : se;
44     FL_LOG("x=" << in << "\texpected_out=" << expected << "\tobtained_out=" << obtained
45             << "\tse=" << se);
46 }
47 FL_LOG("MSE=" << mse / n);
```

The output of this piece of code is the following.

```
x=0.000        expected_out=nan        obtained_out=nan         se=nan
x=0.250        expected_out=0.990       obtained_out=0.840        se=0.022
x=0.500        expected_out=0.959       obtained_out=0.840        se=0.014
x=0.750        expected_out=0.909       obtained_out=0.840        se=0.005
x=1.000        expected_out=0.841       obtained_out=0.840        se=0.000
x=1.250        expected_out=0.759       obtained_out=0.743        se=0.000
x=1.500        expected_out=0.665       obtained_out=0.645        se=0.000
x=1.750        expected_out=0.562       obtained_out=0.547        se=0.000
x=2.000        expected_out=0.455       obtained_out=0.450        se=0.000
x=2.250        expected_out=0.346       obtained_out=0.347        se=0.000
x=2.500        expected_out=0.239       obtained_out=0.245        se=0.000
x=2.750        expected_out=0.139       obtained_out=0.142        se=0.000
x=3.000        expected_out=0.047       obtained_out=0.040        se=0.000
x=3.250        expected_out=-0.033       obtained_out=-0.015        se=0.000
x=3.500        expected_out=-0.100       obtained_out=-0.070        se=0.001
x=3.750        expected_out=-0.152       obtained_out=-0.125        se=0.001
x=4.000        expected_out=-0.189       obtained_out=-0.180        se=0.000
x=4.250        expected_out=-0.211       obtained_out=-0.183        se=0.001
```

```
x=4.500      expected_out=-0.217      obtained_out=-0.185      se=0.001
x=4.750      expected_out=-0.210      obtained_out=-0.188      se=0.001
x=5.000      expected_out=-0.192      obtained_out=-0.190      se=0.000
x=5.250      expected_out=-0.164      obtained_out=-0.153      se=0.000
x=5.500      expected_out=-0.128      obtained_out=-0.115      se=0.000
x=5.750      expected_out=-0.088      obtained_out=-0.078      se=0.000
x=6.000      expected_out=-0.047      obtained_out=-0.040      se=0.000
x=6.250      expected_out=-0.005      obtained_out=-0.007      se=0.000
x=6.500      expected_out=0.033       obtained_out=0.025       se=0.000
x=6.750      expected_out=0.067       obtained_out=0.058       se=0.000
x=7.000      expected_out=0.094       obtained_out=0.090       se=0.000
x=7.250      expected_out=0.114       obtained_out=0.098       se=0.000
x=7.500      expected_out=0.125       obtained_out=0.105       se=0.000
x=7.750      expected_out=0.128       obtained_out=0.112       se=0.000
x=8.000      expected_out=0.124       obtained_out=0.120       se=0.000
x=8.250      expected_out=0.112       obtained_out=0.100       se=0.000
x=8.500      expected_out=0.094       obtained_out=0.080       se=0.000
x=8.750      expected_out=0.071       obtained_out=0.060       se=0.000
x=9.000      expected_out=0.046       obtained_out=0.040       se=0.000
x=9.250      expected_out=0.019       obtained_out=0.040       se=0.000
x=9.500      expected_out=-0.008      obtained_out=0.040       se=0.002
x=9.750      expected_out=-0.033      obtained_out=0.040       se=0.005
MSE=0.001
```
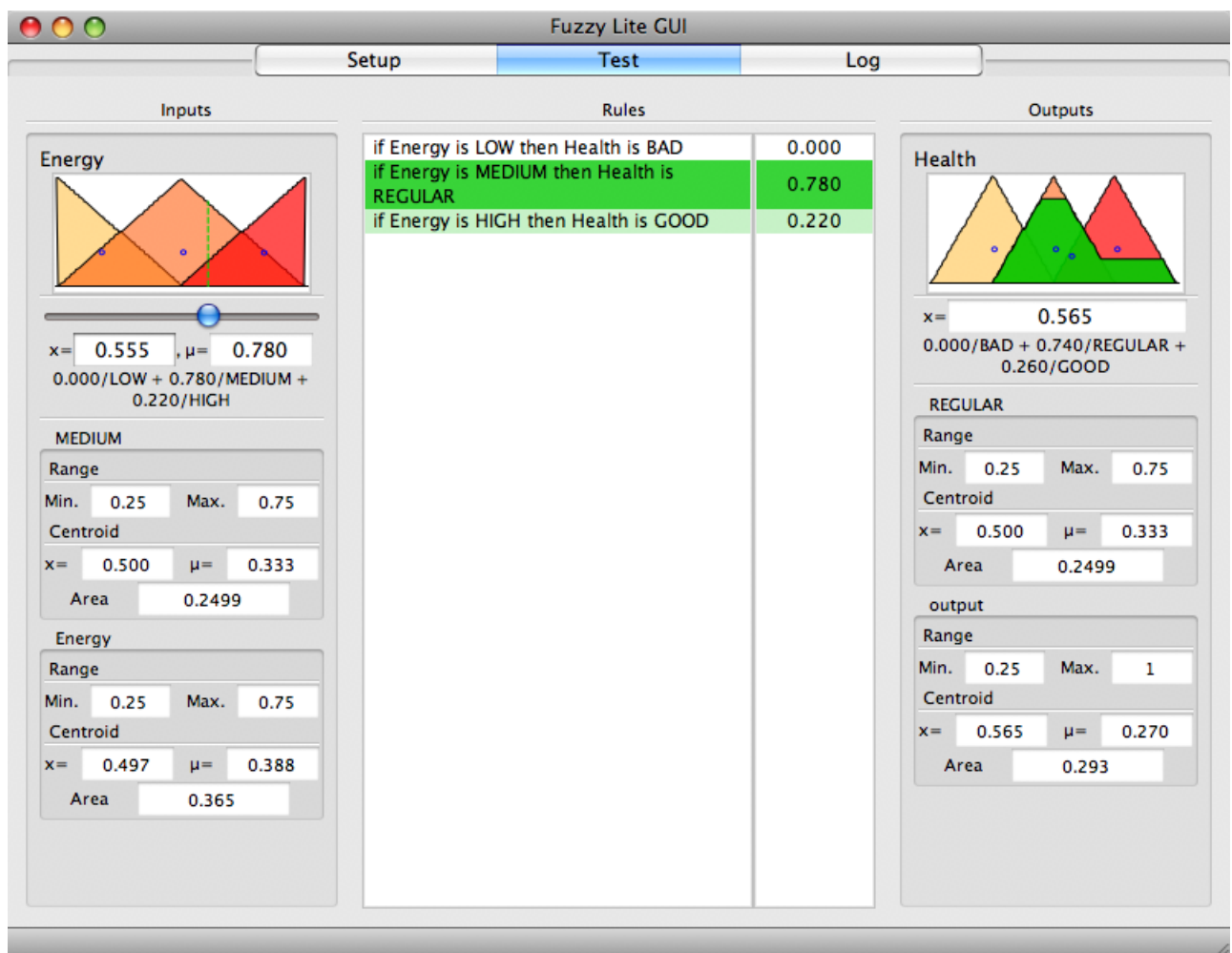
Appendix

GUI Setup

The following figure shows the Setup part of the GUI build for `FuzzyLite v1.02`.
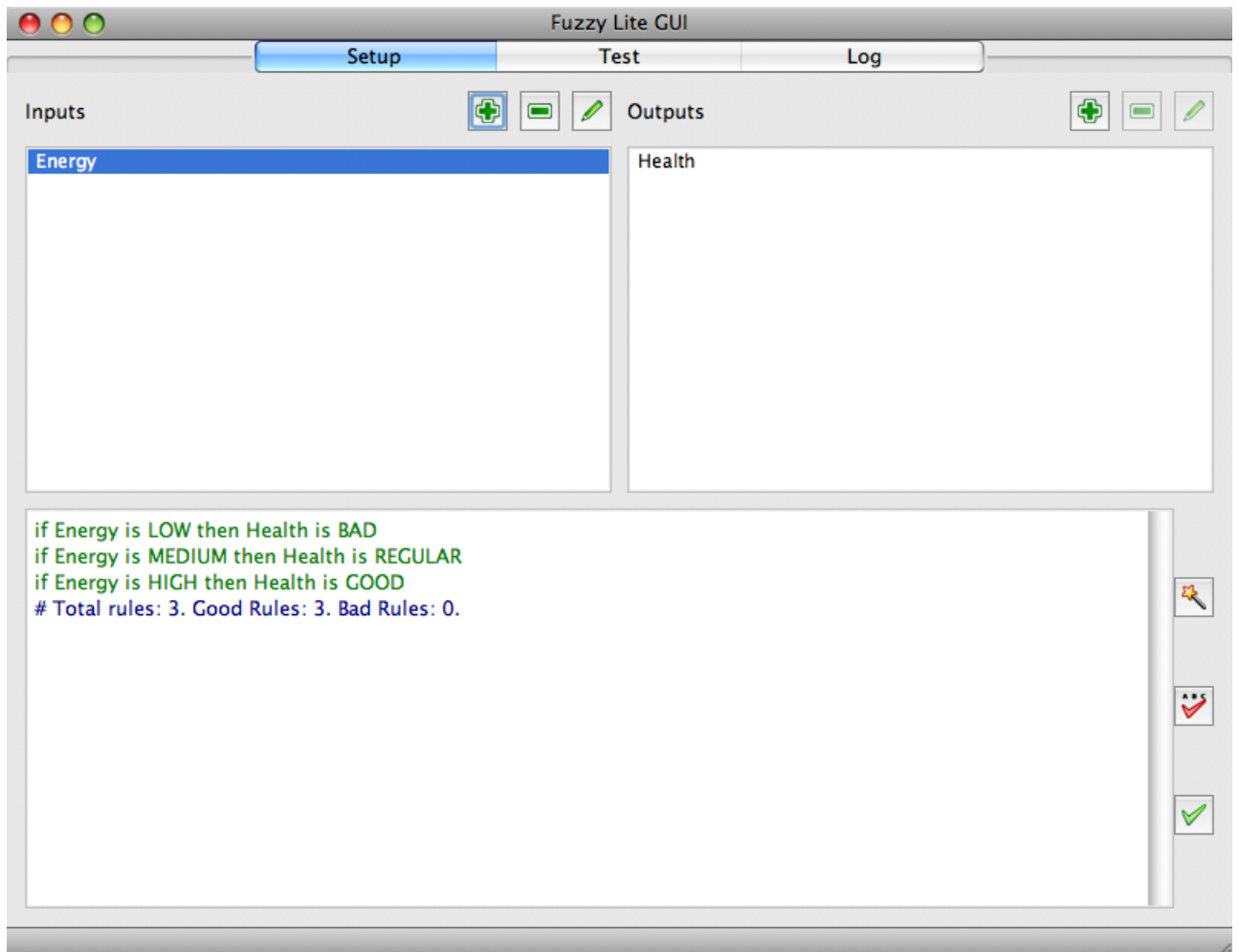
GUI Setup

The following figure shows the Setup part of the GUI build for `FuzzyLite v1.02`.

License of `FuzzyLite v1.02`

```
                      Apache License
                Version 2.0, January 2004
              http://www.apache.org/licenses/
```

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions

to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works

that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS