

---

# EDS Simplification Library(ver. 1.0)

---

## 1 Outline

This library implements the Equivalent Decision Simplification (EDS) method for simplification of GP individuals in symbolic regression problems. It is designed so that the user can readily add new operators, operands and test points.

The library is written in java. It requires JDK 5.0 or higher.

## 2 Contents

In the Simplification directory, you will find:

- Simplification.jar: jar file:
- src: Directory of source files.
  - function: the functions package, used in the Operator class. Each Operator class has one Function object to calculate its semantics.
  - main: the package containing the main program, SimplifyMain.java
  - operand: the package for operands
  - operator: the package for operators
  - simplifyRule: a package of rule classes for rule based simplification.
  - test: a unit test package
- equivalenceNode.csv: the settings for Equivalent Decision Simplification (EDS)
- operand.csv: the settings for operands
- operator.csv: the setting for operators
- sampleFile: sample file for the GP genotype
- sample1: another sample (3 variables example)
- sample2: another sample (big example)

## 3 How to use

```
java -jar Simplification.jar filename
```

The file named “filename” contains GP genotype information. Each generation must be separated by the word “generation”. Results are saved in a file named “simple\_filename”. For example, using sampleFile, if you execute

```
java -jar Simplification.jar sampleFile
```

you create a file named simple\_sampleFile containing the results. In the present version, the target file must be in the current directory. Thus you *cannot* execute from a different directory, as in:

```
java -jar Simplification.jar someDir/sampleFile
```

## 4 Settings How-to

### Operands

Operand information for the GP system is contained in file “operand.csv”. In this file, each line represents a “symbol, value” pair. If an operand is a variable, the value becomes “val”. If an operand is a constant, the value becomes this numerical value.  $\pi$  and  $e$  are special values, representing the java symbols “Math.PI” and “Math.E”.

In the standard version, 9 operators are supplied: +, -, \*, %(protected division: division by 0 results in 1), sin, cos, tan, ln, log. “Rule Based Simplification” only allows the constants “0”, “1”, “PI”( $\pi$ ),  $E$  ( $e$ ). If you wish to use another constant, you will need to change the source code. For example, adding “2” as an operand, and adding rules like “1+1=2” is not easy in this version. However, using EDS in addition to rule-based simplification, you can check this pattern. Don’t specify any unused operands. Lines which start with ‘#’ are comments, and blank lines are ignored

examples:

```
X,val
1,1.0
PI,Math.PI
E,Math.E
```

### Operators

Operator information for the GP system is contained in file “operator.csv”. In this library, each operator is a java class. In the settings file, each line represents one operator, with the format:

operator symbol, operator class name(include package name), number of operands, class name of “Rule Based Simplification” (include package name), commutative or non-commutative

The 9 built-in operators are +, -, \*, %, sin, cos, tan, ln, log.

In case a function value become infinite (e.g.  $\log(0)$ ), the value is re-set to “1”. If you wish to change this value, please check the Function class in the function package. E.g. if you would like to change the action of protected divide %, please edit the function/ProtectedDiv.java.

If you need more operators, you will need to make your own operator class. In this case, you have to implement the interface “IFunction” in the function package, and write a suitable “calculate” method. If this operator needs “Rule Base Simplification”, you will also need to make a new class which implements the interface “ISimplifyRule” in the simplifyRule package.

Don’t create unused operators. Lines which start with ‘#’ are comment lines, and blank lines are skipped.

Example:

```
+,function.Add,2,simplifyRule.AddRule,commutative
-,function.Sub,2,simplifyRule.SubRule,noncommutative
*,function.Mul,2,simplifyRule.MulRule,commutative
/,function.ProtectedDiv,2,simplifyRule.ProtectedDivRule,noncommutative
S,function.Sin,1,simplifyRule.SinRule,noncommutative
C,function.Cos,1,simplifyRule.CosRule,noncommutative
```

### Equivalent Decision Simplification and Errors

The first line of “equivalenceNode.csv” contains the maximum values which should be treated as zero for equivalent decision simplification, and the relative error that should be accepted in treating values as equivalent (see the paper for details), in the order zero value  $\rightarrow$  relative error.

Subsequent lines contain information about the operands which are used in Equivalent Decision Simplification. The format is the same as “operand.csv”.

Don’t create unused operators. Lines which start with ‘#’ are comment lines, and blank lines are skipped.

Example:

```
1E-9,1E-4
X,val
1,1.0
0,0.0
```

## Individual genotypes

For greater portability, an XML implementation is planned; however in the initial implementation, the individual genotype must be represented in postfix notation. The symbol delimiter is the space character. Test points are specified in the first line. The format for test points is as follows:

val1,num1-1,num1-2,num1-3:val2,num2-1,num2-2....

where val<sub>n</sub> is a variable in the GP genotype. For example,

X,1,2,3

means there is only one variable “X” and it may have substitution values 1, 2, 3 in the Equivalent Decision Simplification test. In a typical symbolic regression problem, where the evolved function is fitted to a set of points, the values would be the x components of the set of points

X,1,2,3:Y,4,5

means there are two variables “X” and “Y”, and all combinations of “X” and “Y” values will be substituted in equivalence testing. In this case, each subtree is checked for equivalence on the points

$$(X, Y) = (1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5) \quad (1)$$

in the Equivalent Decision Simplification check. Any file name may be used, and the result will be saved in the file prefixed by “simple\_”. E.g. if the filename is “tmp”, the simplification result is saved in the file named “simple\_tmp”. Lines which start with ‘#’ are comments lines, and blank lines are skipped. Lines starting with the word “generation”, are saved in the result file.

Example:

X,1,3,5,7:Y,2,4,6:Z,1,2,3,4,5,6

#population information

X 1 + 1 Z \* / x + S

Y Z \* 1 Y - X S + 1 - +

1 Y + X X + \* 1 Y - Z + X / +