At this point I am creating a genetic program that will convert the model creation to a completely automated process.

The environment is presented a specific set of variables that they can be combined to a decision tree having as leaf nodes the signal (BUY / SELL / DO NOTHING / FREE MARGIN etc) while each internal node will be an boolean expression based in the provided variables, functions and expressions.

Variables

(1) Scalar: Any variable that can be represented in a single numeric format

Some examples might be CURRENT BID, CURRENT MINUTE etc

(2) Collection: A vector of doubles

An example might be the prices for the specific bar

Collection Functions

A collection function receives as first argument a collection and returns a Scalar. Based in the nature of the function it might receive more parameters.

Examples of a collection function are the following:

Average(collection)

MinValue(collection)

MaxValue(collection)

MovingAverage(collection, duration: Scalar)

Length(collection)

GetValueAt(collection, index)

GetRandomValue(collection)

A more complicated example is the following:

Assuming that we need to select a random minute to trade, like we do in the fortieth minute strategy. In this case our logic will look like this:

- Provide a collection containing all the minutes of the hour:

AvailableMinutes = $\{1, 2, 3, ... 59\}$

- The genetic program can then create the following:

GetRandomValue(AvailableMinutes)

Which will later be used for the creation of an expression that will be used in the decision tree.

Expressions

An expression is a binary function always returning a boolean

We have two types of expressions:

(1) Comparisons

A comparison takes two arguments of type Scalar and returns TRUE / FALSE

Examples:

MORE

LESS

EQUAL

(2) Logical Operations

A logical operation combines two expressions returning TRUE / FALSE Examples:

AND

OR

Random Expression Genetator

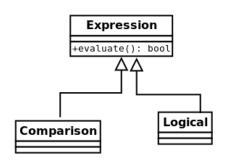
A function returning an expression using any of the variables of the environment

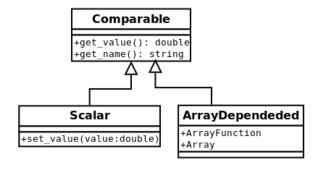
Random Program Genetator

A function returning a complete program resembling a decision tree having as nodes expressions and leafs signals

API for the genetic program

Initial Design of the genetic program





Operator +AND +OR +LESS +MORE +EQUAL

+BUY +SELL +FREE_MARGIN +DO NOTHING

Environment

+add_scalar(name:string): void

+add_array(name:string)

+add_array_function(name:string,function_ptr

GeneticNode

+expression: Expression
+left_node: GeneticNode
+right_node: GeneticNode
+evaluate(): Signal

Array

+get_name(): string

ArrayFunction

+get_name(): string

Factory

+create_expression(): Expression

+pick_scalar(): Scalar

+pick_array_depended(): ArrayDepended
+pick_comparable(): Comparable

+create_program(): GeneticNode