Evolutionary Algorithms: Genetic Programming

Jiří Kubalík
Department of Cybernetics, CTU Prague

http://cw.felk.cvut.cz/doku.php/courses/a4m33bia/start

Genetic Programming (GP)

:: **GP shares with GA** the philosophy of survival and reproduction of the fittest and the analogy of naturally occurring genetic operators.

:: **GP differs from GA** in a representation, genetic operators and a scope of applications.

:: **GP is extension of the conventional GA** in which the structures undergoing adaptation are trees of dynamically varying size and shape representing hierarchical computer programs.

:: **Applications**

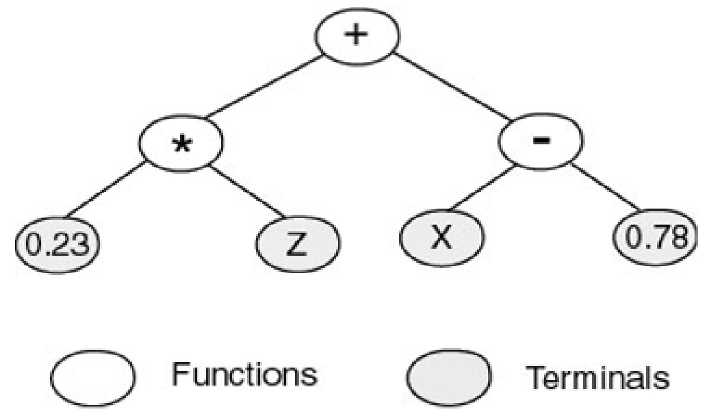
- learning programs,
- learning decision trees,
- learning rules,
- learning strategies,
- ...

GP: Representation

:: All possible trees are composed of **functions** (inner nodes) and **terminals** (leaf nodes) appropriate to the problem domain

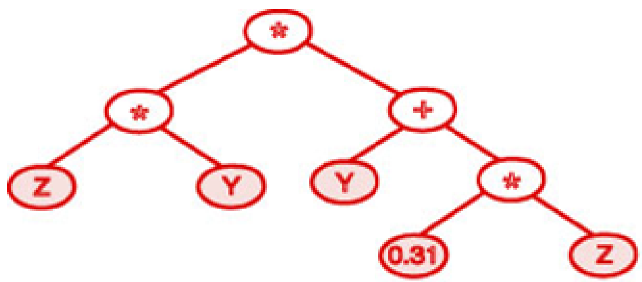
- **Terminals** – inputs to the programs (independent variables), real, integer or logical constants, actions.
- **Functions**
  - arithmetic operators (+, -, *, /),
  - algebraic functions (sin, cos, exp, log),
  - logical functions (AND, OR, NOT),
  - conditional operators (If-Then-Else, cond?true:false),
  - and others.

Example: Tree representation of a LISP S-expression $0.23 * Z + X - 0.78$

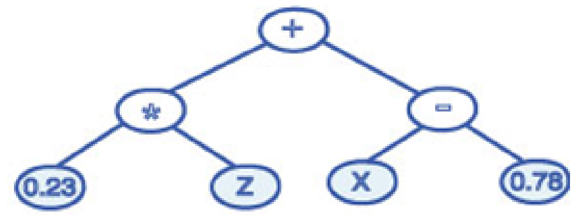


:: **Closure** – each of the functions should be able to accept, as its argument, any value that may be returned by any function and any terminal.

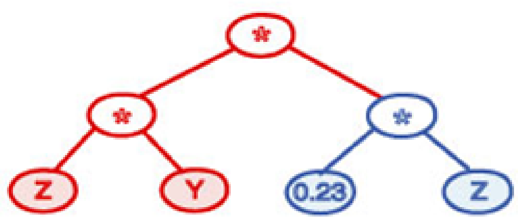
GP: Crossover



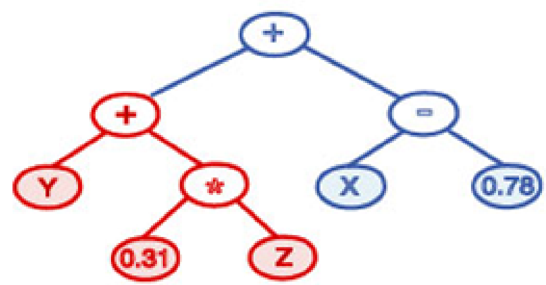
Parent 1: Z * Y * (Y + 0.31 * Z)



Parent 2: 0.23 * Z + X - 0.78



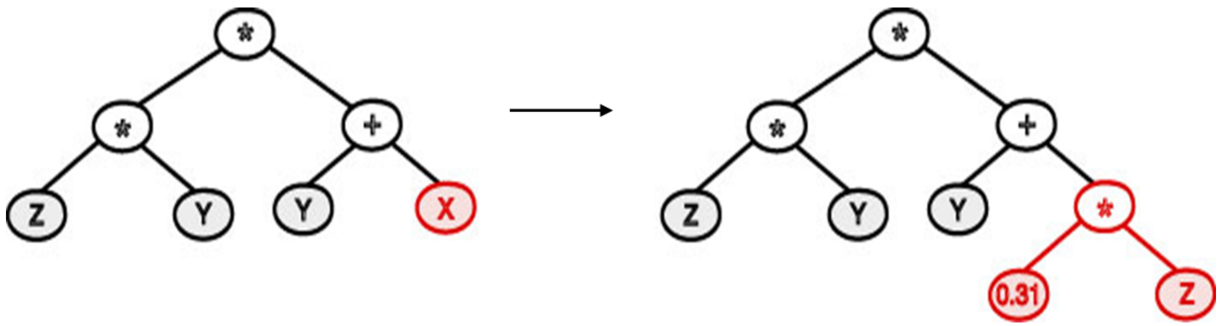
Child 1: 0.23 * Y * Z^2



Child 2: Y + 0.31 * Z + X - 0.78

GP: Mutation and Other Operators

:: **Mutation** replaces selected subtree with a randomly generated new one.



:: **Other operators**:

- permutation,
- editing,
- encapsulation,
- decimation,
- ...

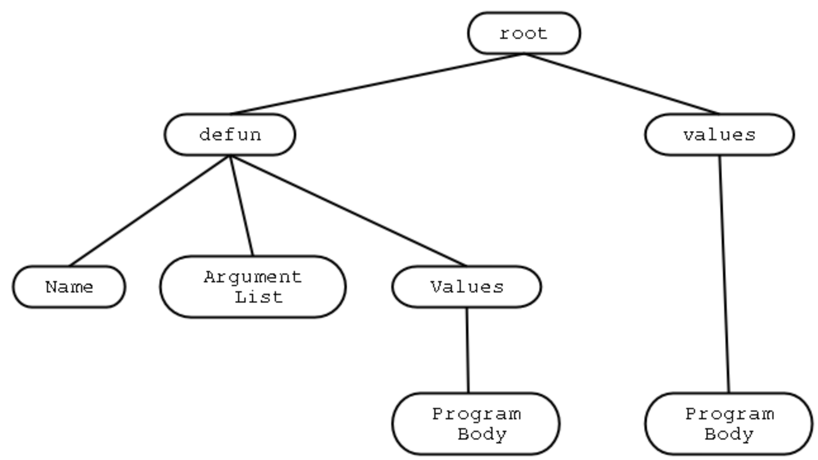
Automatically Defined Functions

:: **Automatically Defined Functions** – idea similar to subroutines in programming languages.

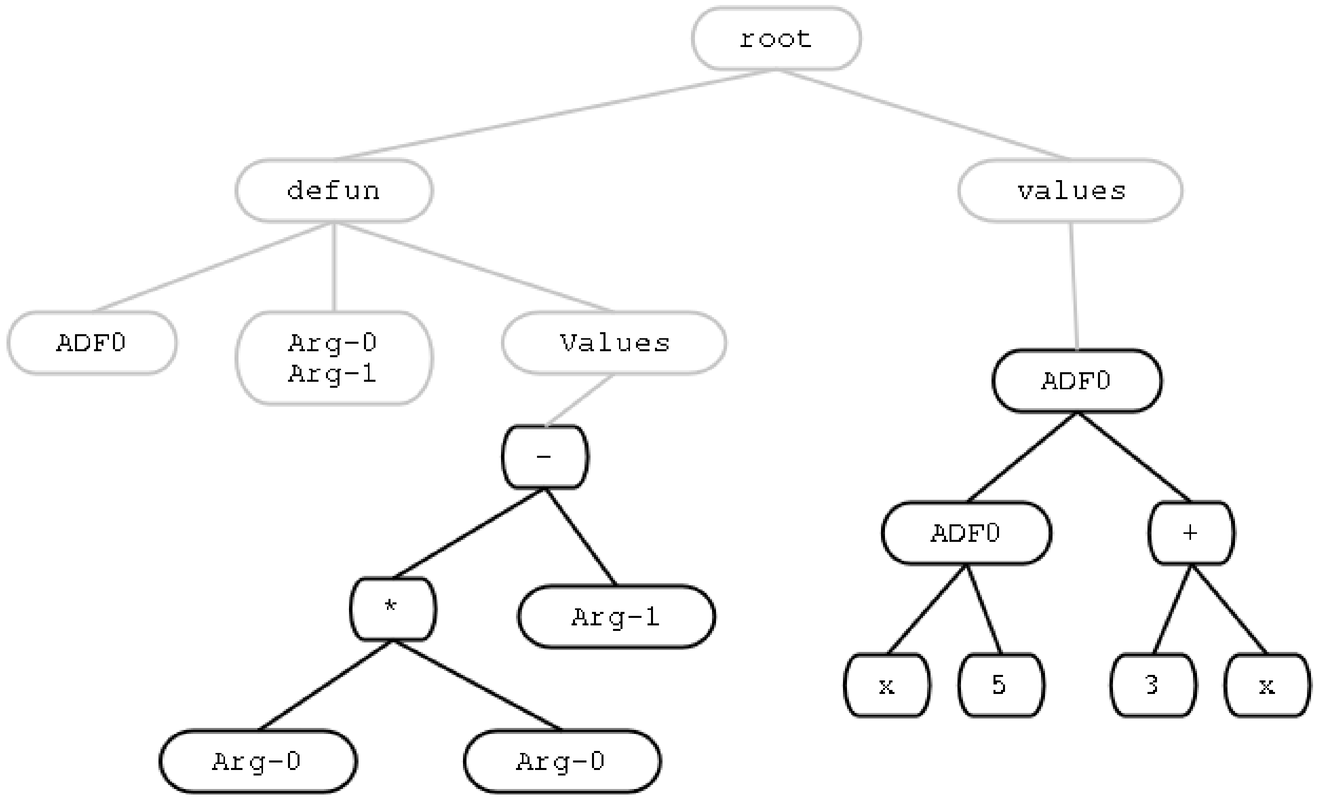
- Reuse eliminates the need to "reinvent the wheel" on each occasion when a particular sequence of steps may be useful.
- Code encapsulation – protection from crossover and mutation.
- Simplification – less complex code, easier to evolve.
- ADFs are the focus of *Genetic Programming II: Automatic Discovery of Reusable Programs* (Koza, 1994).

:: **Structure partly fixed:**

- Function defining branches (ADFs).
- Result-producing branch (a calling program). Typically, the automatically defined functions are invoked with different instantiations of their dummy variables.



Automatically Defined Functions: Example



Terminal set: Arg-0 Arg-1
Function set: task-dependent
(e.g. +,-,*)

Terminal set: task-dependent
(e.g. X, 1, 2, 3, 4, 5)
Function set: ADF0 + task-dependent
(e.g. ADF0, +, -, *)

GP: Initialisation

:: Maximum initial depth of trees D_{max} is set.

:: **Full method** (each branch has $depth = D_{max}$):

- nodes at depth $d < D_{max}$ randomly chosen from function set F,
- nodes at depth $d = D_{max}$ randomly chosen from terminal set T.

:: **Grow method** (each branch has $depth \leq D_{max}$):

- nodes at depth $d < D_{max}$ randomly chosen from $F \cup T$,
- nodes at depth $d = D_{max}$ randomly chosen from T.

:: Common GP initialization:

- **Ramped half-and-half** – grow & full method each deliver half of initial population.

GP: Selection

:: **Parent selection typically fitness proportionate.**

:: **Over-selection** in very large populations

- rank population by fitness and divide it into two groups:
  - group 1: best x% of population,
  - group 2: other (100-x)%.
- 80% of selection operations chooses from group 1, 20% from group 2,
- for pop. size = 1000, 2000, 4000, 8000 x = 32%, 16%, 8%, 4%.
- motivation: to increase efficiency, %'s come from rule of thumb.

:: **Survivor selection**

- Typical: generational scheme (thus none)
- Recently steady-state is becoming popular for its elitism.

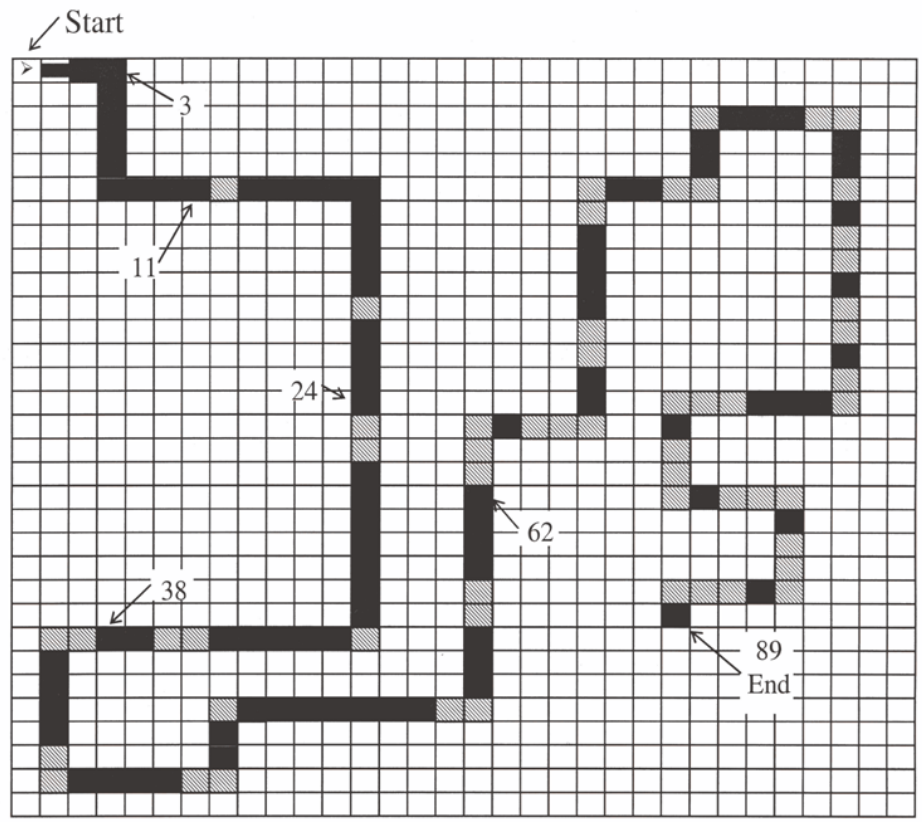
Artificial Ant Problem

:: **Santa Fe trail**

- 32×32 **grid** with 89 food pieces.
- **Obstacles**
  - $1\times, 2\times$ strait,
  - $1\times, 2\times, 3\times$ right/left.

:: **Ant capabilities**

- **detects** the food right in front of him in direction he faces.
- **actions** observable from outside
  - MOVE – makes a step and eats a food piece if there is some,
  - LEFT – turns left,
  - RIGHT – turns right,
  - NO-OP – no operation.



:: **Goal** is to find a strategy that would navigate an ant through the grid so that it finds all the food pieces in the given time (600 time steps).

Artificial Ant Problem: GP Approach

:: **Terminals**

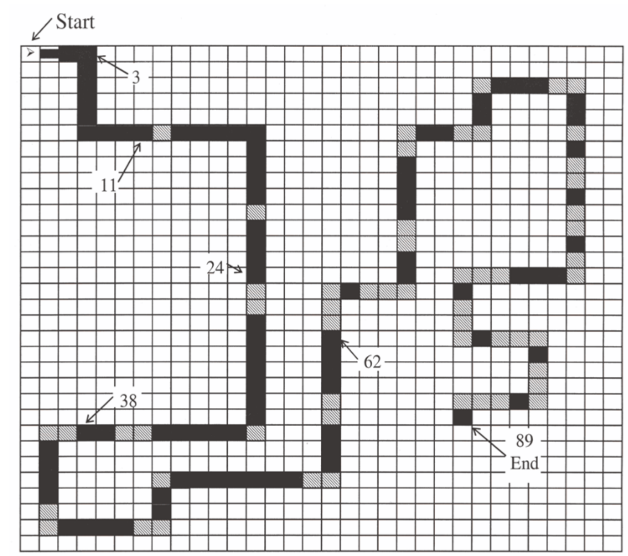
- motorial section,
- T = MOVE, LEFT, RIGHT.

:: **Functions**

- conditional IF-FOOD-AHEAD – food detection, 2 arguments (is/is_not food ahead),
- unconditional PROG2, PROG3 – sequence of 2/3 actions.

:: Ant repeats the program until time runs out (600 time steps) or all the food has been eaten.

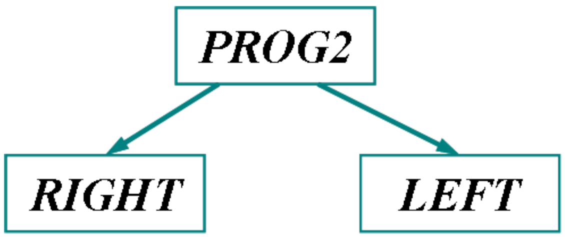
Santa Fe trail



Artificial Ant Problem: GP Approach cont.

:: Typical solutions in the initial population

- this solution



  completely fails in finding and eating the food,

- similarly this one

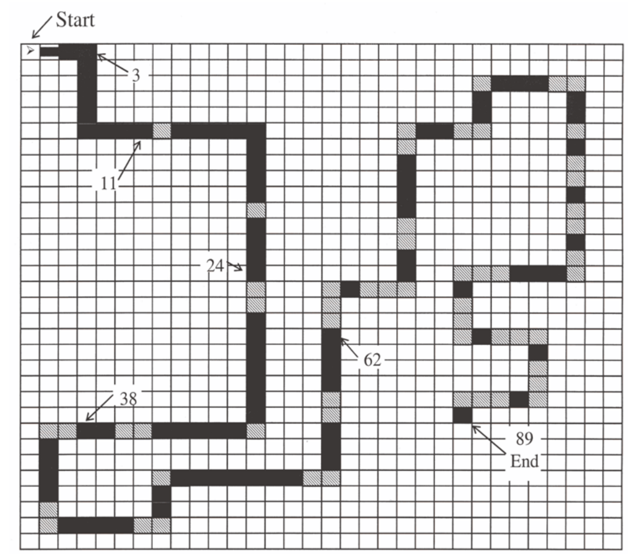
  (IF-FOOD-AHEAD (LEFT)(RIGHT)),

- this one

  (PROG2 (MOVE) (MOVE))

  just by chance finds 3 pieces of food.

Santa Fe trail



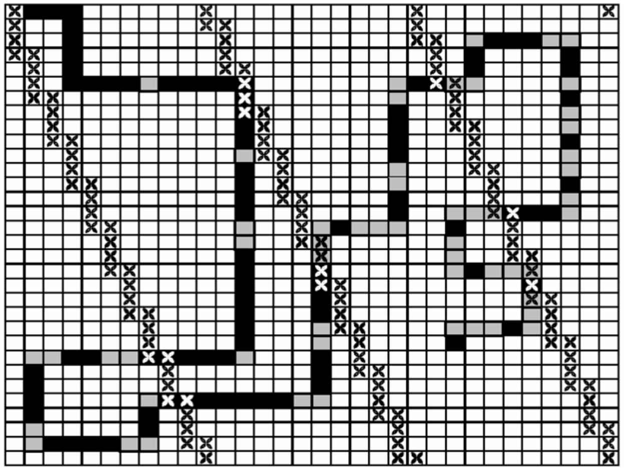
Artificial Ant Problem: GP Approach cont.

:: More interesting solutions

- **Quilter** – performs systematic exploration of the grid,

  (PROG3 (RIGHT)
  (PROG3 (MOVE) (MOVE) (MOVE))
  (PROG2 (LEFT) (MOVE)))

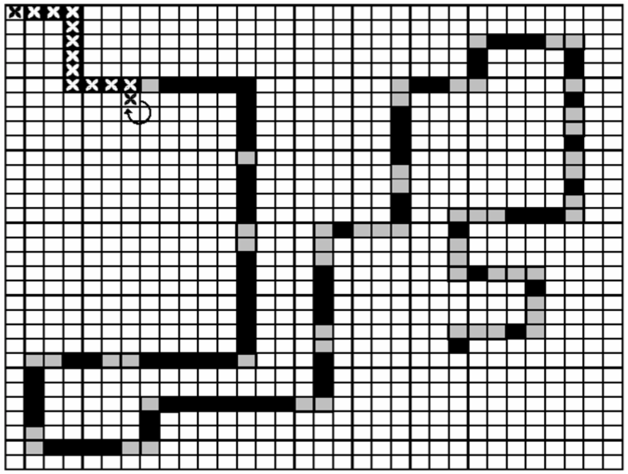
Quilter performance



- **Tracker** – perfectly tracks the food until the first obstacle occurs, then it gets trapped in an infinite loop.

  (IF-FOOD-AHEAD (MOVE) (RIGHT))

Tracker performance

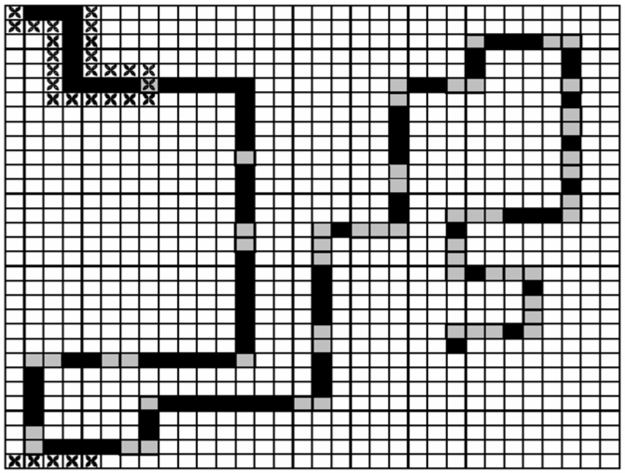


Artificial Ant Problem: GP Approach cont.

- **Avoider** – perfectly avoids food!!!

```
(I-F-A  (RIGHT)
        (I-F-A    (RIGHT)
                  (PROG2 (MOVE) (LEFT))))
```

Avoider performance



:: Average fitness in the initial population is 3.5

Artificial Ant Problem: GP result

:: In generation 21, the following solution was found that already navigates an ant so that he eats all 89 food pieces in the given time.

```
(I-F-A  (MOVE)
        (PROG3  (I-F-A   (MOVE)
                         (RIGHT)
                (PROG2   (RIGHT)
                         (PROG2   (LEFT)
                                  (RIGHT))))
                (PROG2   (I-F-A   (MOVE)
                                  (LEFT))
                         (MOVE))))
```

This program solves every trail with the obstacles of the same type as occurs in Santa Fe trail.

:: **Compare the computational complexity with the GA approach!!!**

GA approach: $65.536 \times 200 = 13 \times 10^6$ trials.

vs.

GP approach: $500 \times 21 = 10.500$ trials.

GP: Trigonometric Identity

:: **Task** is to find an equivalent expression to $cos(2x)$.

:: **GP implementation:**

- **Terminal set** $T = \{x, 1.0\}$.
- **Function set** $F = \{+, -, *, \%, sin\}$.
- **Training cases**: 20 pairs (x_i, y_i), where x_i are values evenly distributed in interval $(0, 2\pi)$.
- **Fitness**: Sum of absolute differences between desired y_i and the values returned by generated expressions.
- **Stopping criterion**: A solution found that gives the error less than 0.01.

GP: Trigonometric Identity cont.

:: **1. run, 13^{th} generation**

$$(-(-1(*(sin x)(sin x))))(*(sin x)(sin x)))$$

which equals (after editing) to $1–2 * sin^2 x$

:: **2. run, 34^{th} generation**

$$(-1(*(*(sin x)(sin x))2))$$

which is just another way of writing the same expression.

:: **3. run, 30^{th} generation**

$$\begin{array}{lll} (sin & (-(-2(*x2)) & \\ & (sin(sin(sin(sin(sin(sin(*(sin & (sin 1)) \\ & & (sin 1)) \\ ))))))))) & & \end{array}$$

Note that the expression on the second and third row is almost equal to $\pi/2$ so the discovered identity is

$$cos(2x) = sin(\pi/2–2x)$$

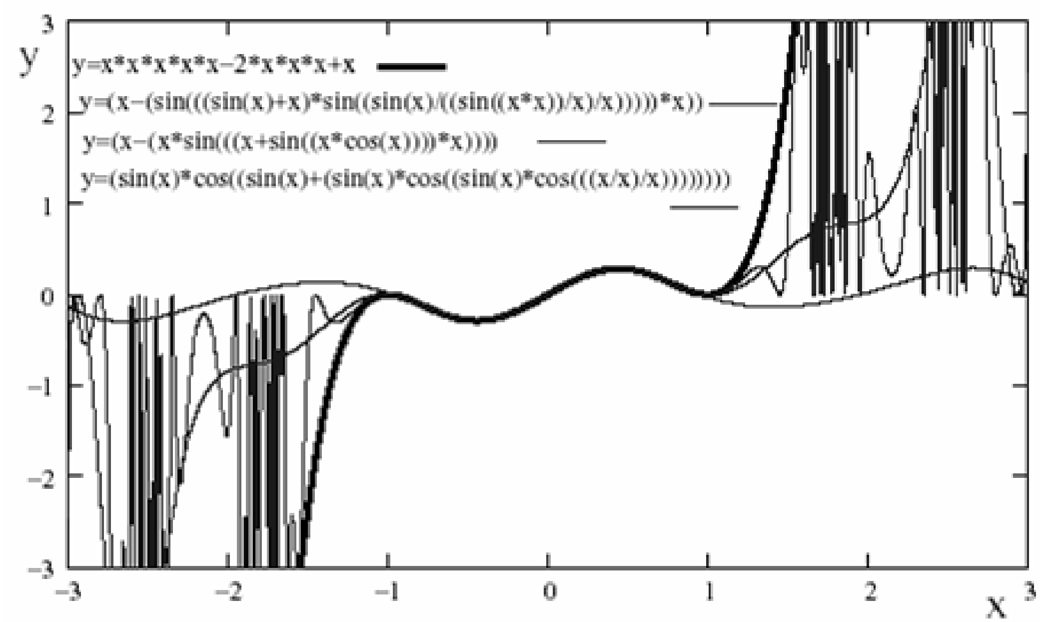
.

GP: Symbolic Regression

:: **Task** is to find a function that fits to training data evenly sampled from interval $< -1.0, 1.0 >$.

:: **GP implementation:**

- **Terminal set** $T = \{x\}$.
- **Function set** $F = \{+, -, *, \%, sin, cos\}$.
- **Training cases**: 20 pairs (x_i, y_i), where x_i are values evenly distributed in interval $(-1, 1)$.
- **Fitness**: Sum of errors calculated over all (x_i, y_i) pairs.
- **Stopping criterion**: A solution found that gives the error less than 0.01.



:: **Besides the desired function** other three were found

- with a very strange behavior outside the interval of training data,
- though optimal with respect to the defined fitness.

GP: Decision Trees

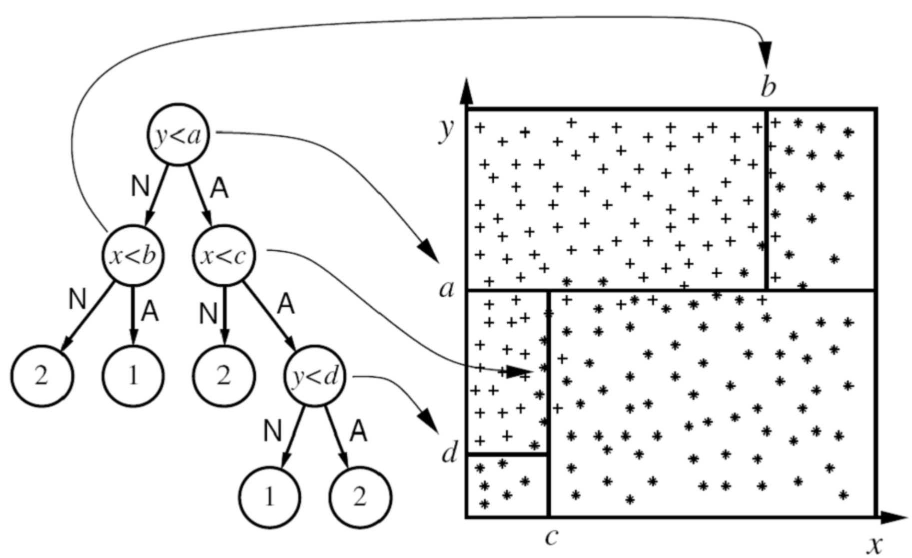
:: **Classical decision trees (DT)**

- Inner nodes represent simple decisions of type

$$att > const, att = const$$

$\Rightarrow$ **axis-parallel splits**, in some cases not very efficient way to partition the attribute space.

- Leaf node indicates a class to which the sample, which corresponds to the decisions made along the branch from root to the leaf, belongs.
- Standard learning algorithms – Quinlan's ID3 (Iterative Dichotomiser 3).

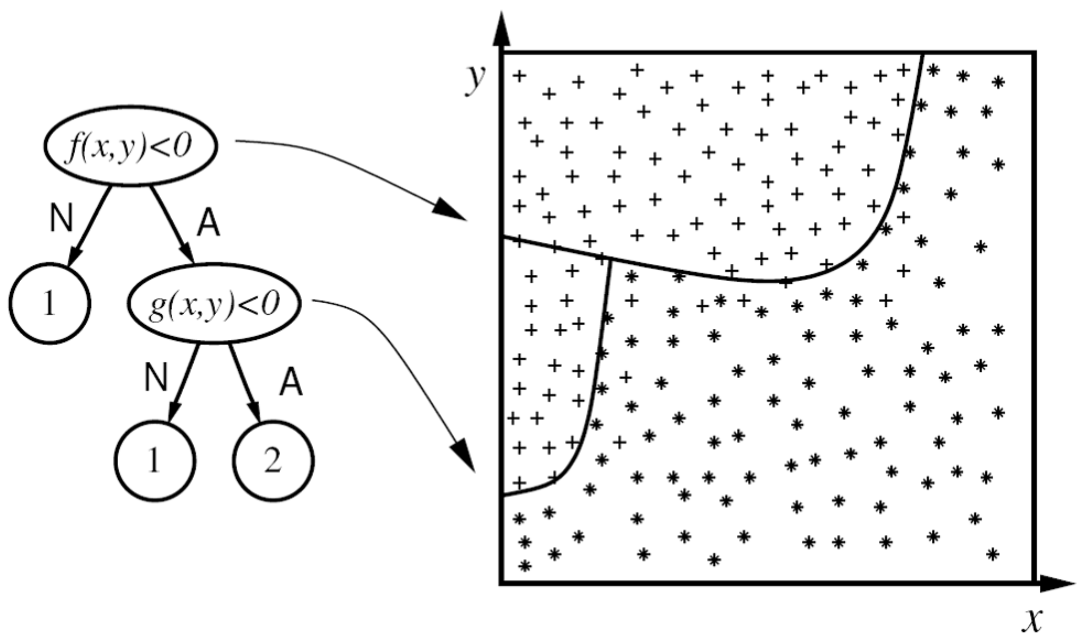


GP: Decision Trees cont.

:: **Oblique/Multivariate DT**

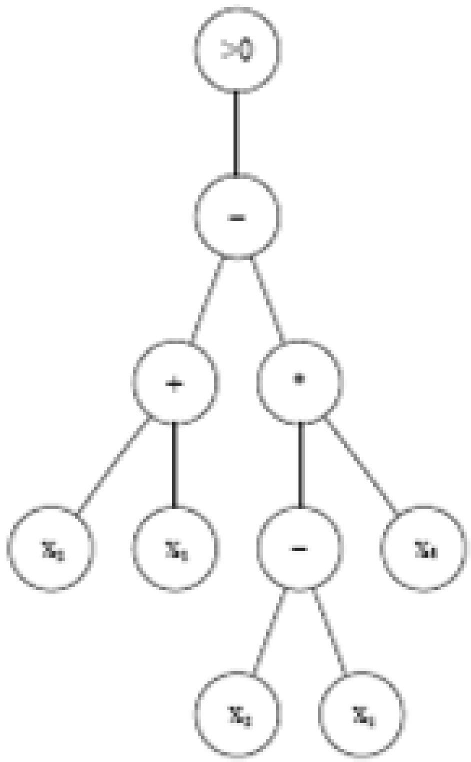
- Inner (decision) nodes represent complex rules (functions).
- More flexible splits are possible.
- But may be hard to understand and interpret.

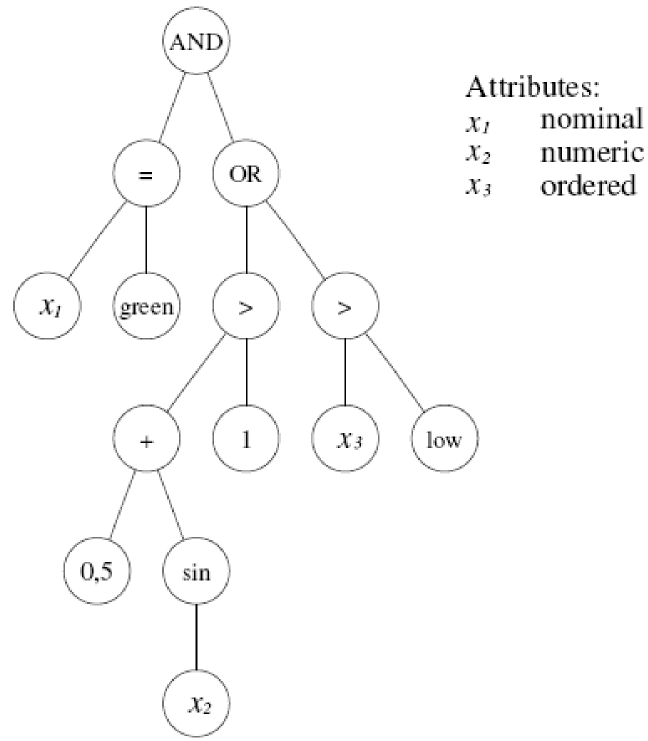
:: This looks much better, but how to find the rules?



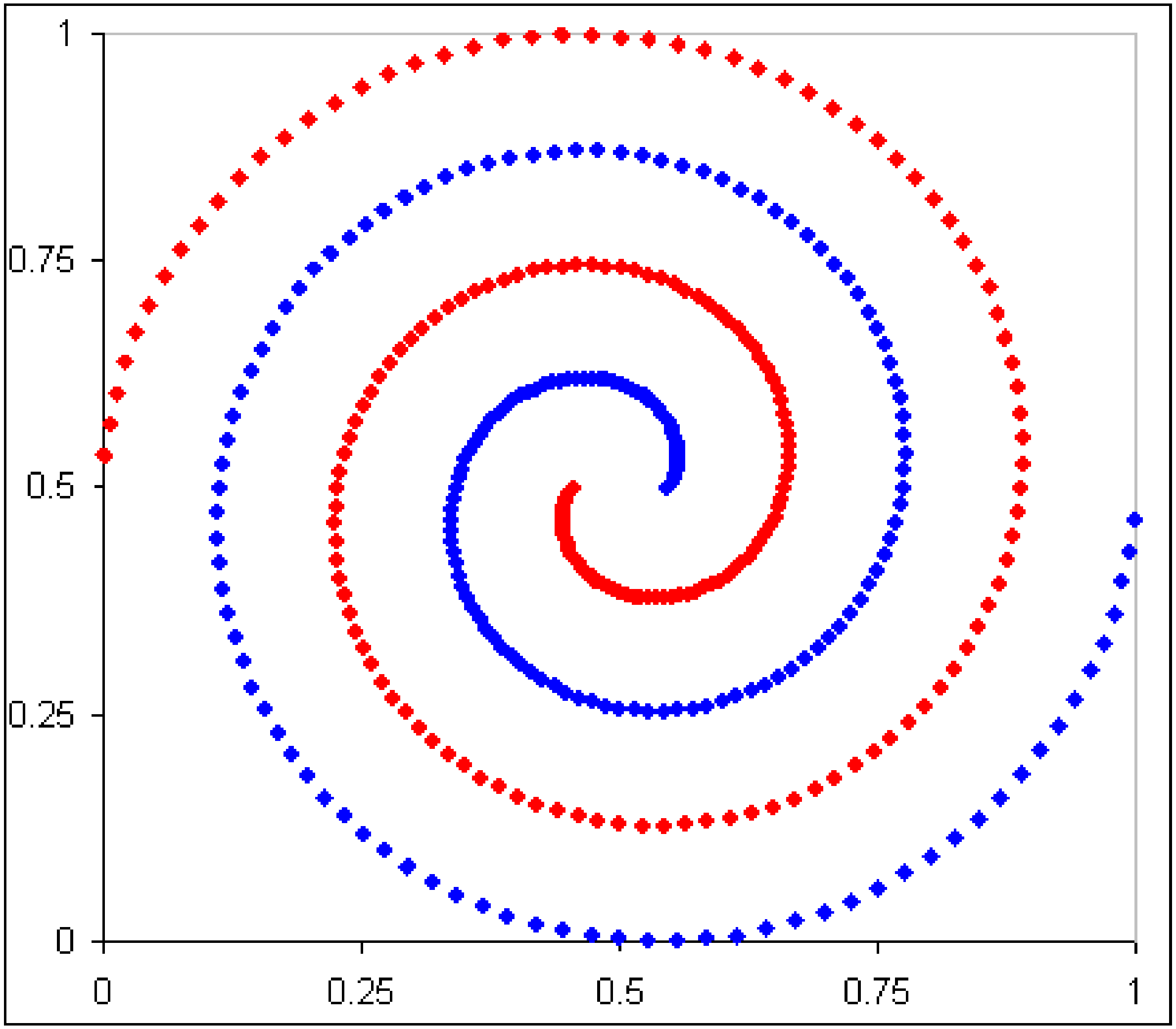
Multivariate DT: Rule Structure

:: Root node returns boolean value $\{true, false\}$





Intertwined Spirals Problem



Multivariate DT: Effect of Terminals and Functions Selection

:: **Without any prior knowledge A**

- Terminals: $T_a = \{x, y, R\}$,
- Functions: $F_a = \{+, -, ?, (> 0)\}$.

:: **With prior knowledge about the circular characteristics of data**

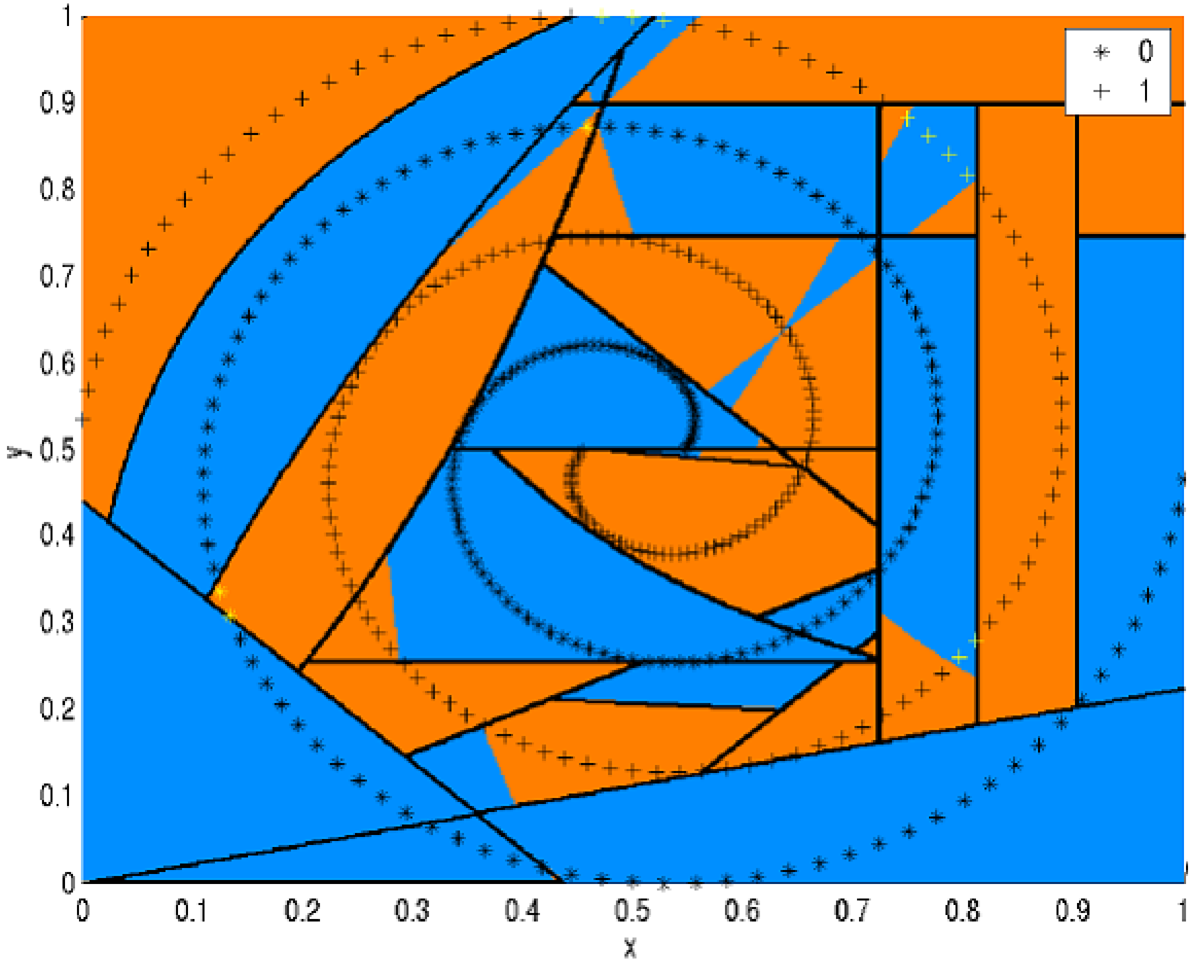
- Polar coordinates.
- Terminals: $T_b = \{\varrho, \varphi, R\}$, where ϱ and φ are radius and phase of data points,

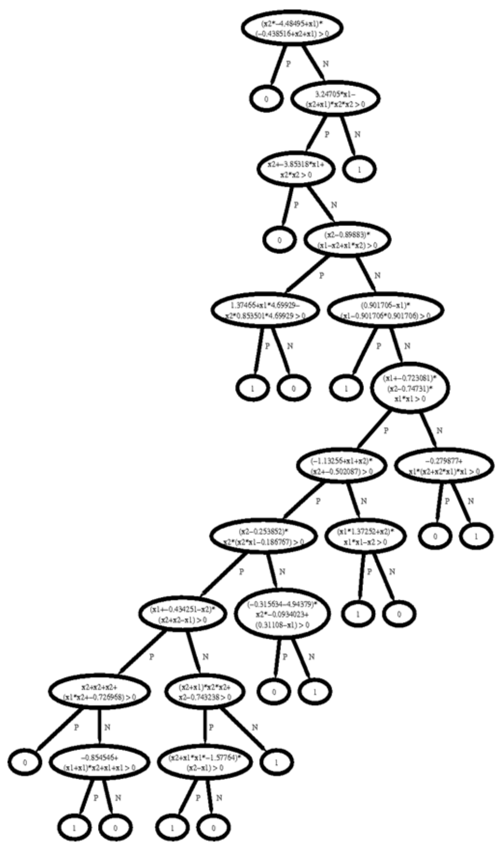
$$\varrho = \sqrt{x^2 + y^2} \text{ and } \varphi = \arctan \tfrac{y}{x}$$

- Functions: $F_b = \{+, -, *, (> 0), sin\}$.

Multivariate DT: Classifiers Evolved with $\{T_a, F_a\}$

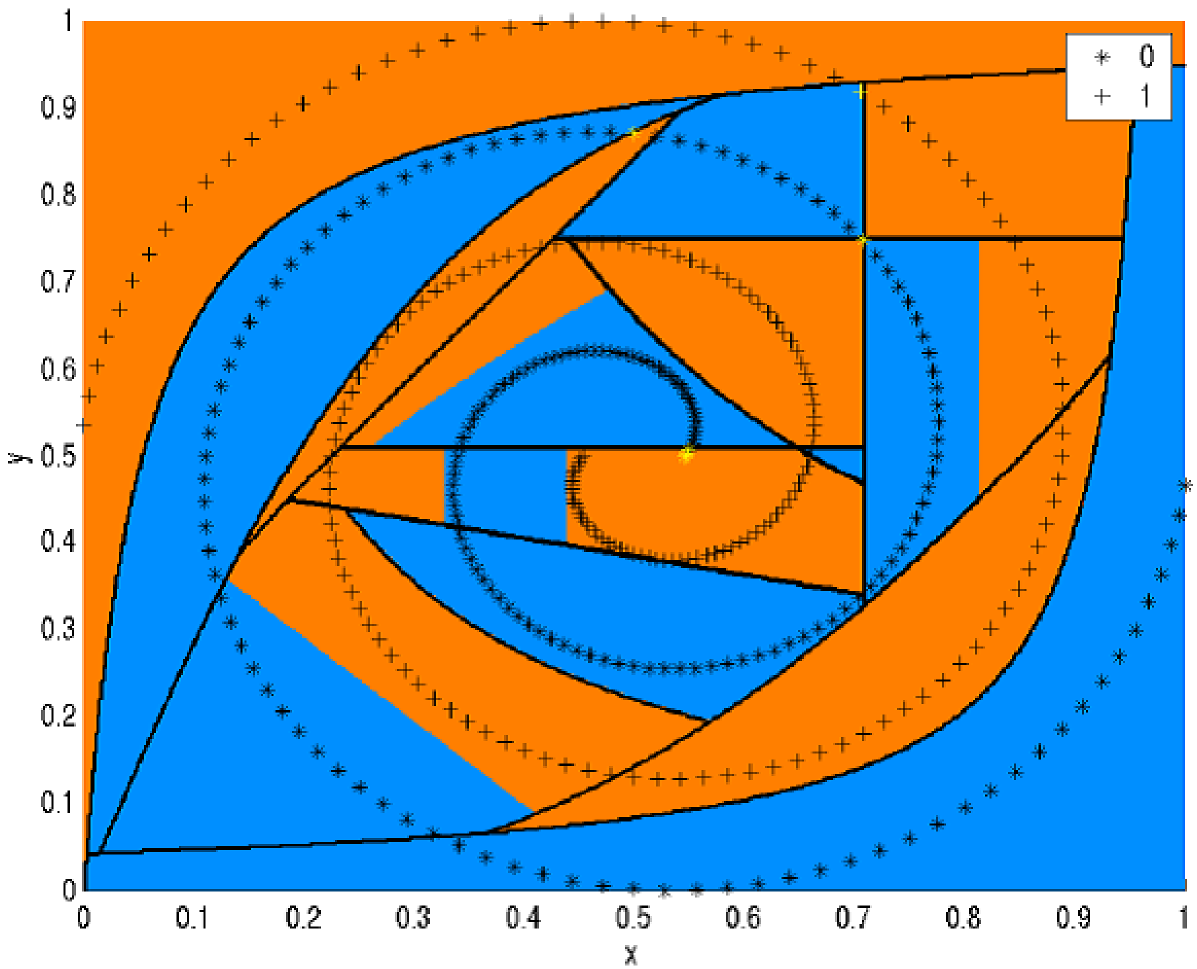
This does not look nice.

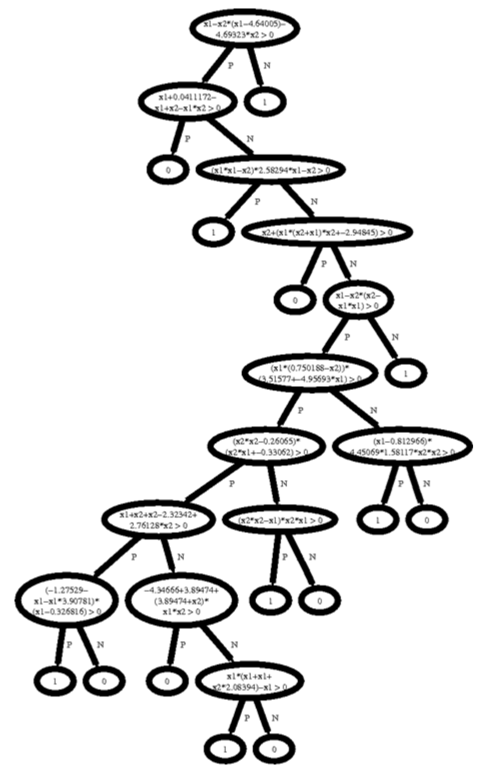




Multivariate DT: Classifiers Evolved with $\{T_a, F_a\}$ cont.

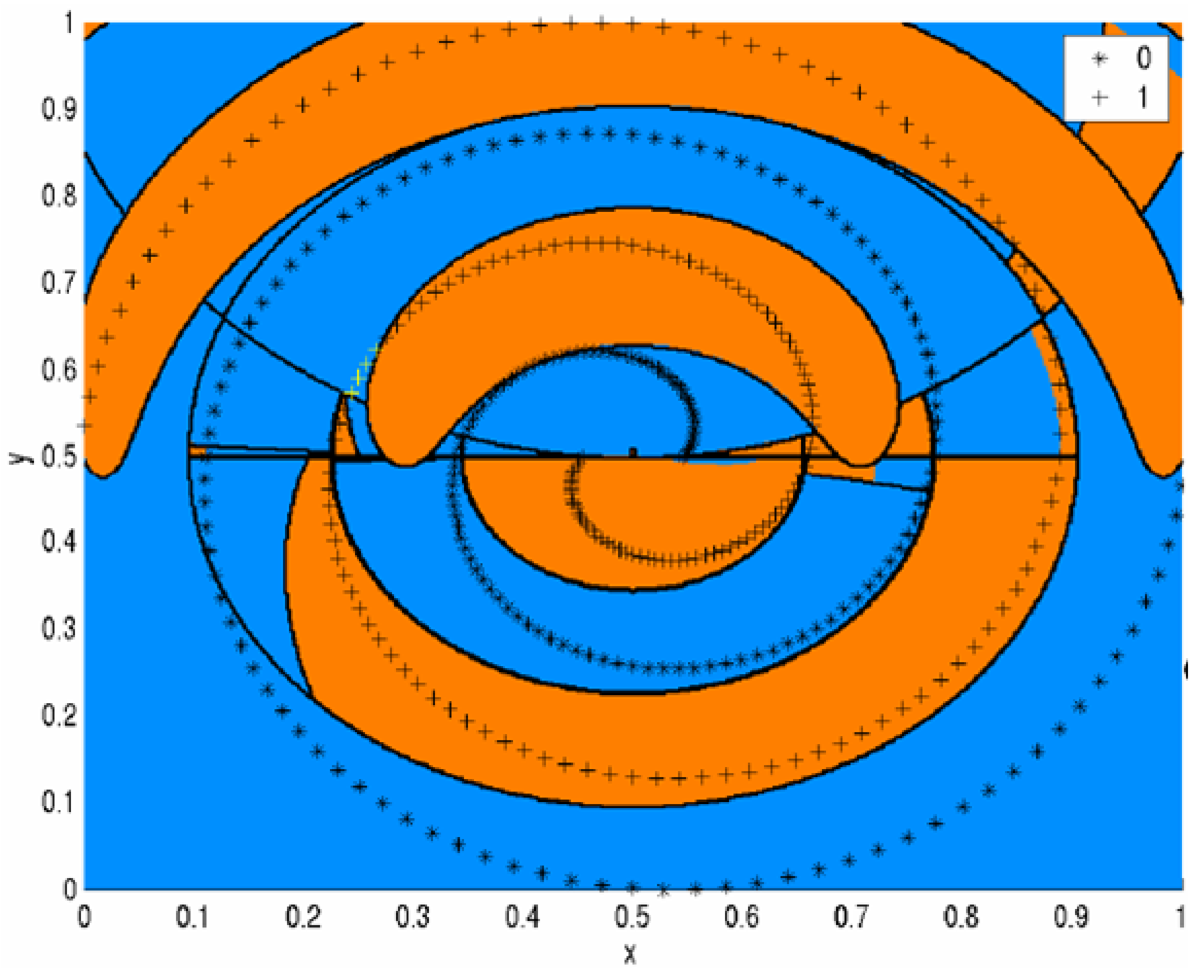
Neither does this one.

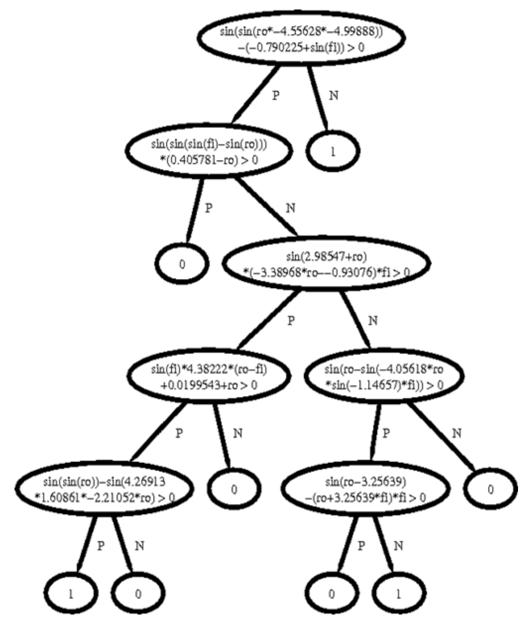




Multivariate DT: Classifiers Evolved with $\{T_b, F_b\}$

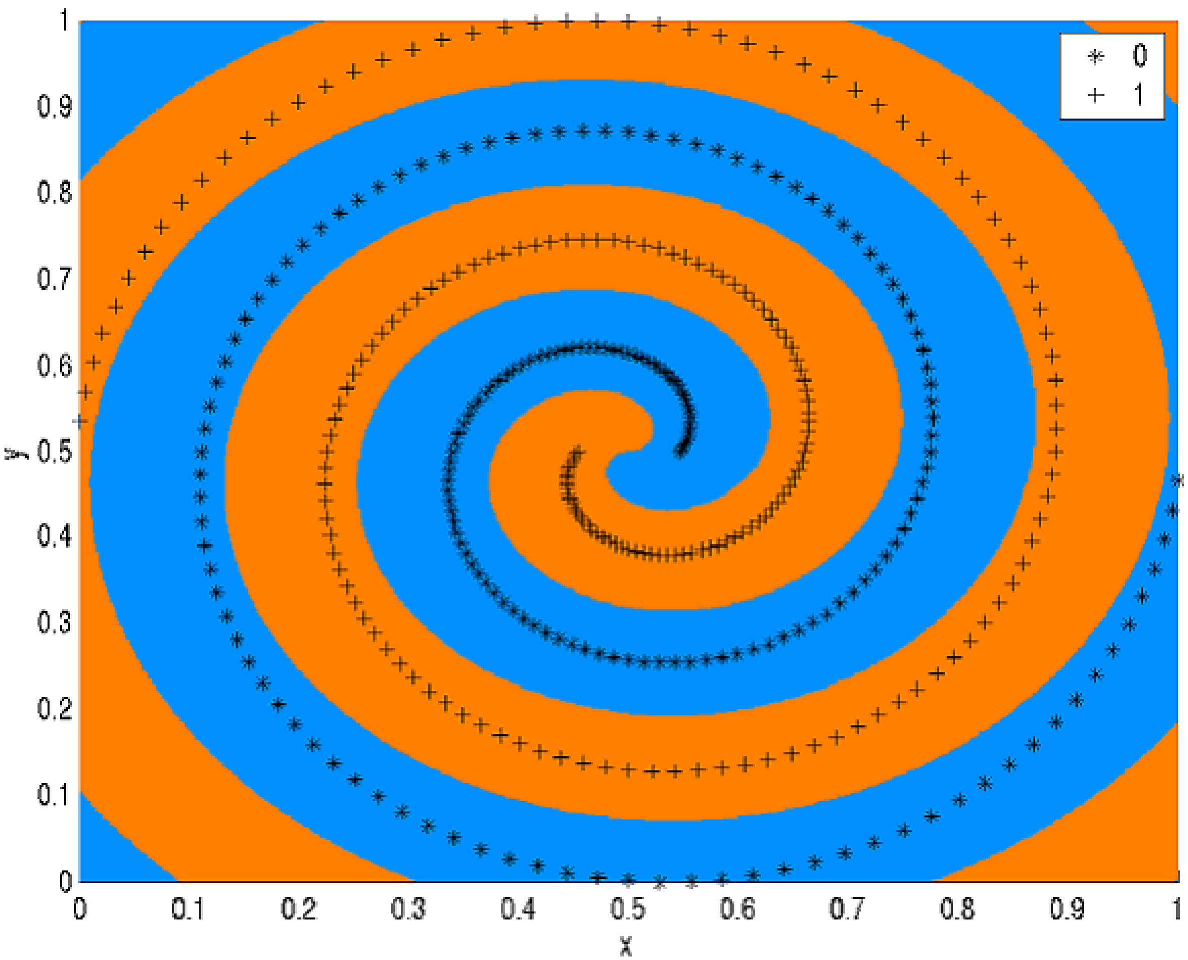
This is better already.

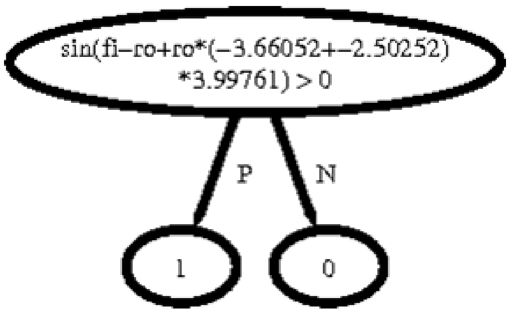




Multivariate DT: Classifiers Evolved with $\{T_b, F_b\}$ cont.

Wow, this one is really nice!!!





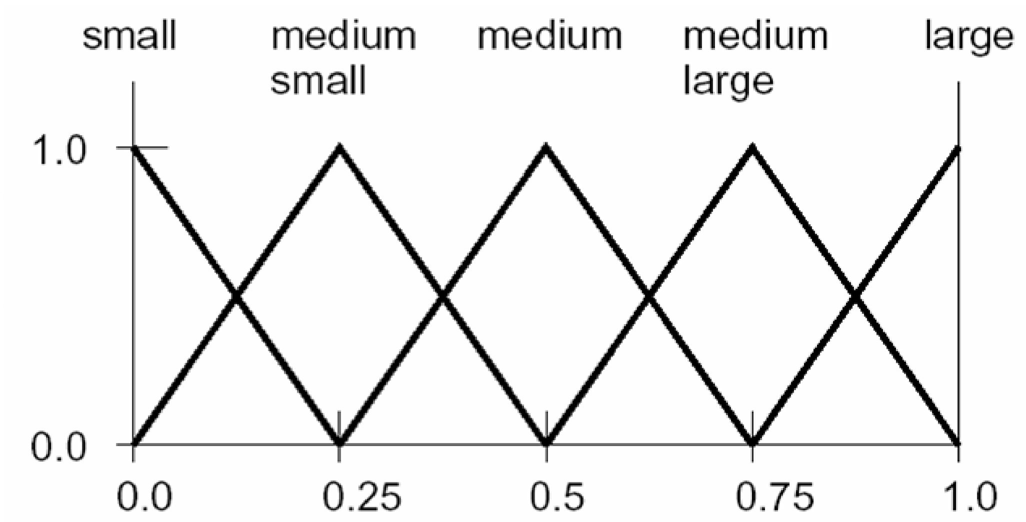
GP: Evolving Fuzzy-rule based Classifier

:: **Classifier** consists of fuzzy if-then rules of type

IF ($x1$ is $medium$) and ($x3$ is $large$) THEN $class = 1$ with $cf = 0.73$

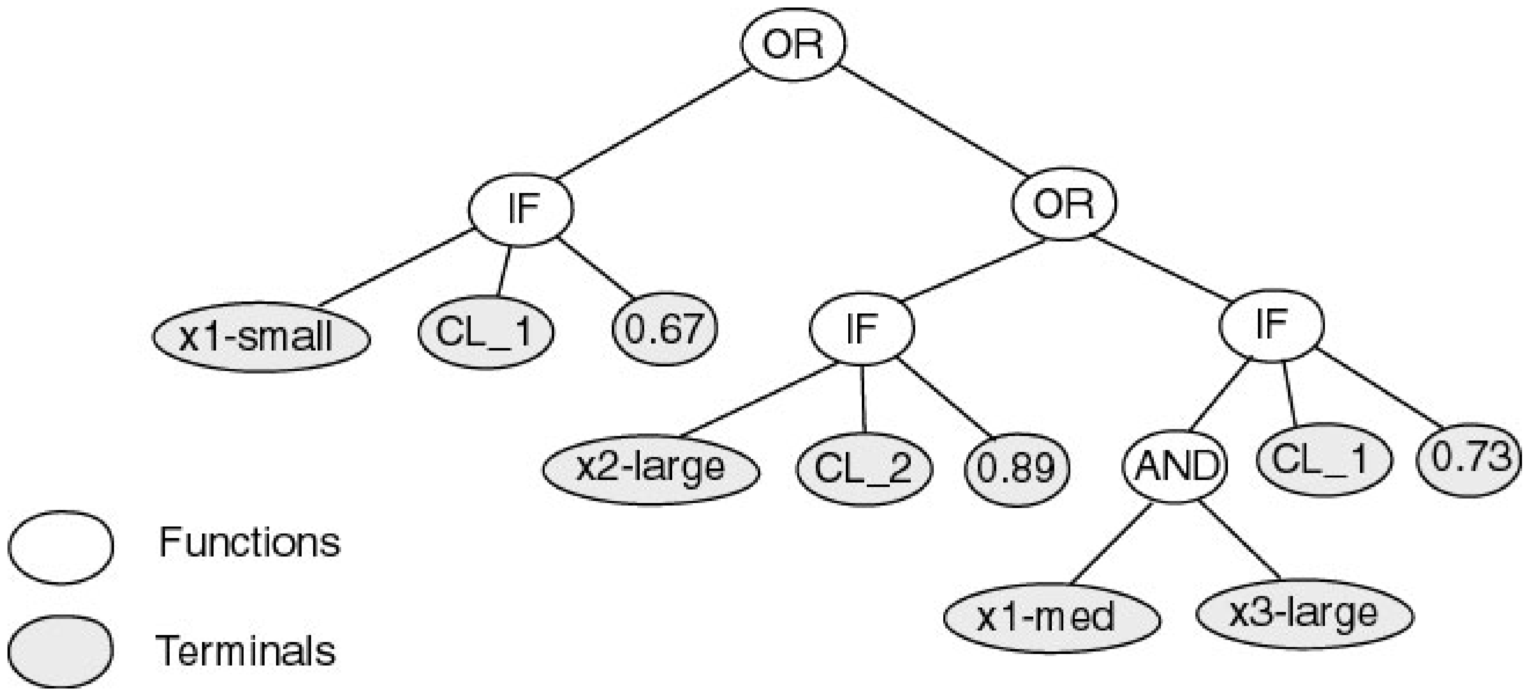
:: **Linguistic terms** – small, medium small, medium, medium large, large.

:: **Fuzzy membership functions** – approximate the confidence in that the crisp value is represented by the linguistic term.



Rule Base Example

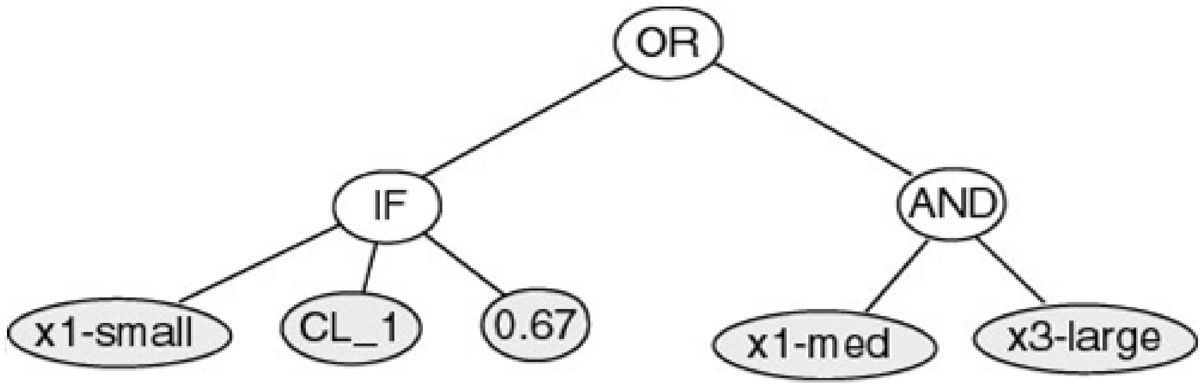
:: Three rules connected by OR.



Illegal Tree

:: **Tree does not represent a correct rule base.**

- obviously due to the fact that the **closure** property does not hold here.



:: This might happen very often since crossover, mutation etc. are just blind operators.

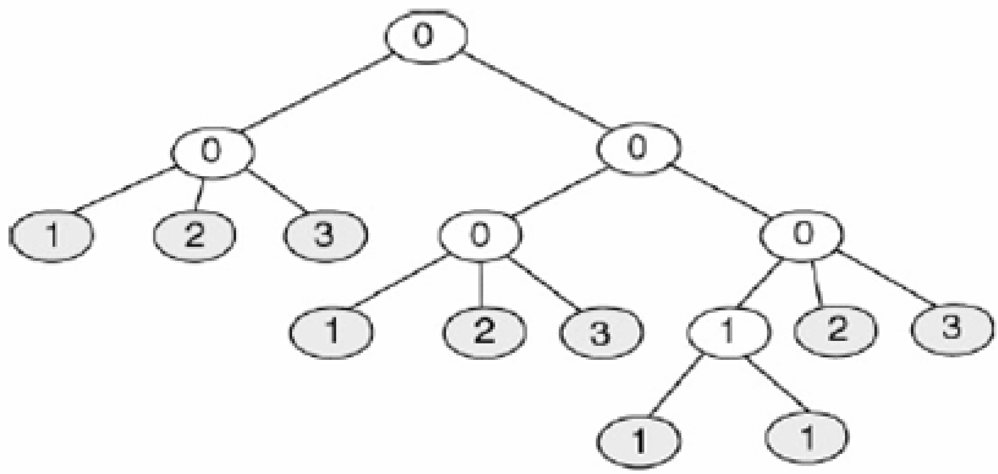
What can we do?

Strongly typed GP

:: **Strongly typed GP**

- prevents generating illegal individuals,
- quite a big overhead $\Longrightarrow$ inefficient for large trees.

F / T	Output	Input
OR	0	0, 0
IF	0	1, 2, 3
AND	1	1, 1
IS	1	None
CLASS	2	None
CF	3	None



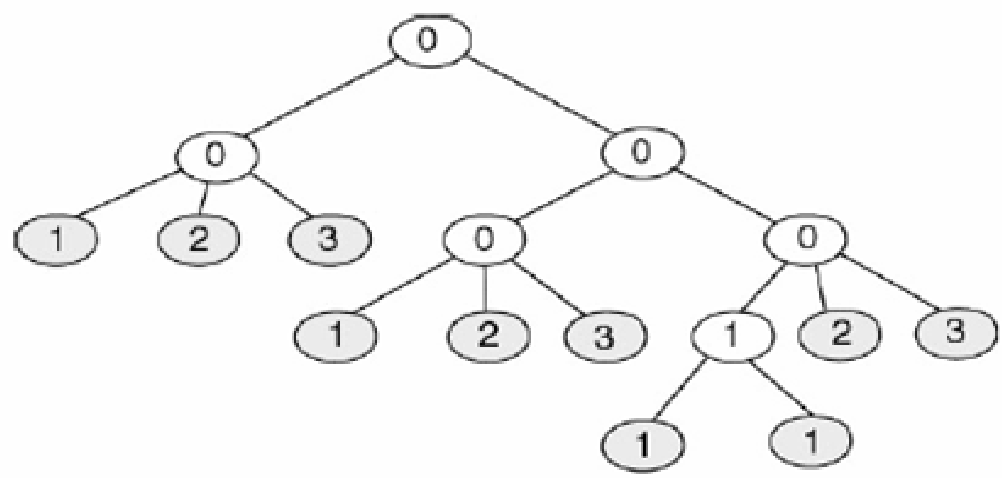
:: Can be done any better?

Strongly typed GP

:: Strongly typed GP

- prevents generating illegal individuals,
- quite a big overhead $\Longrightarrow$ inefficient for large trees.

F / T	Output	Input
OR	0	0, 0
IF	0	1, 2, 3
AND	1	1, 1
IS	1	None
CLASS	2	None
CF	3	None



:: Can be done any better?

Grammatical Evolution (GE) – designed to evolve programs in any language, that can be described by a context free grammar. GE evolves tree structures, but operates on simple linear string chromosomes :-o

Reading

- Poli, R., Langdon, W., McPhee, N.F.: *A Field Guide to Genetic Programming*, 2008, http://www.gp-field-guide.org.uk/