GENETIC PROGRAMMING

Introduction

- Earlier ideas in 70's, 80's
- Developed in 90's, J. Koza
- J. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, 1992.
- Can evolutionary approaches can be used to write partial or complete computer programs?

Application Areas

- Classifier systems,
- Symbolic regression,
- Grammar induction,
- Engineering design, i.e. electronics, mechanics.
- In general, most machine learning tasks involving supervised or reinforcement learning.

GP in summary

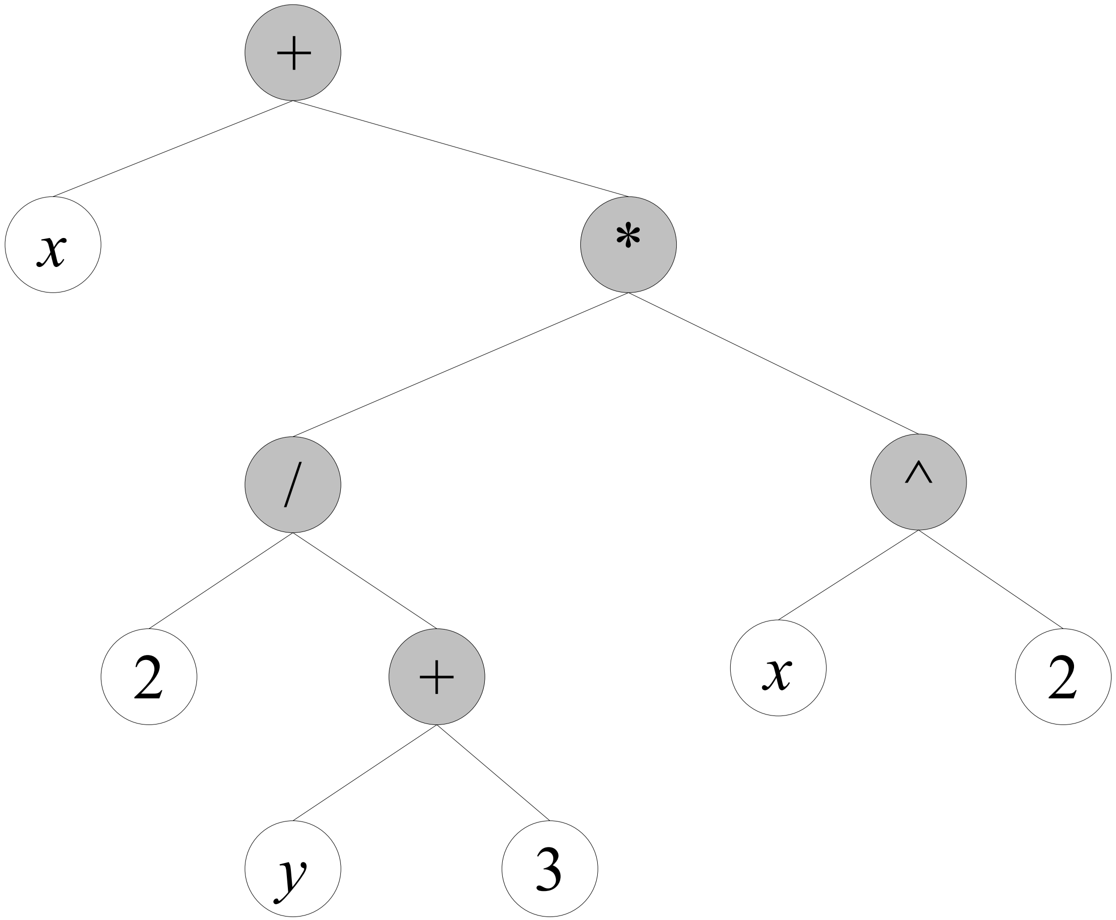
- Search space: partial computer programs, expressions, complex data structures.
- Representation: Usually trees. Linear and graph respresentations may also be defined. Usually genotype is very similar to phenotype.
- Genetic operators: Special operators involving manipulation of data structures.

Tree Representation

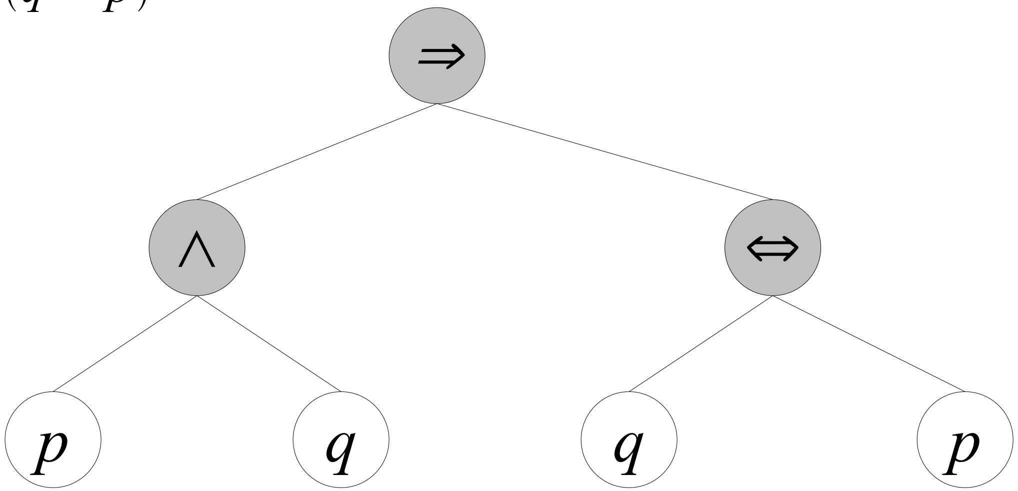
- Function set for intermediate nodes.
- Terminal set for leaf nodes.
- More suitable to functional P.L.'s, evolving expressions, decision trees, classifier systems...
- Examples:
  - Arithmetic formula $x+\frac{2}{y+3}x^2$
  - Locigal formula $(p \wedge q) \Rightarrow (q \Leftrightarrow p)$
  - Program

```
i=0
while (i<100) {
   i=i+i;
}
```

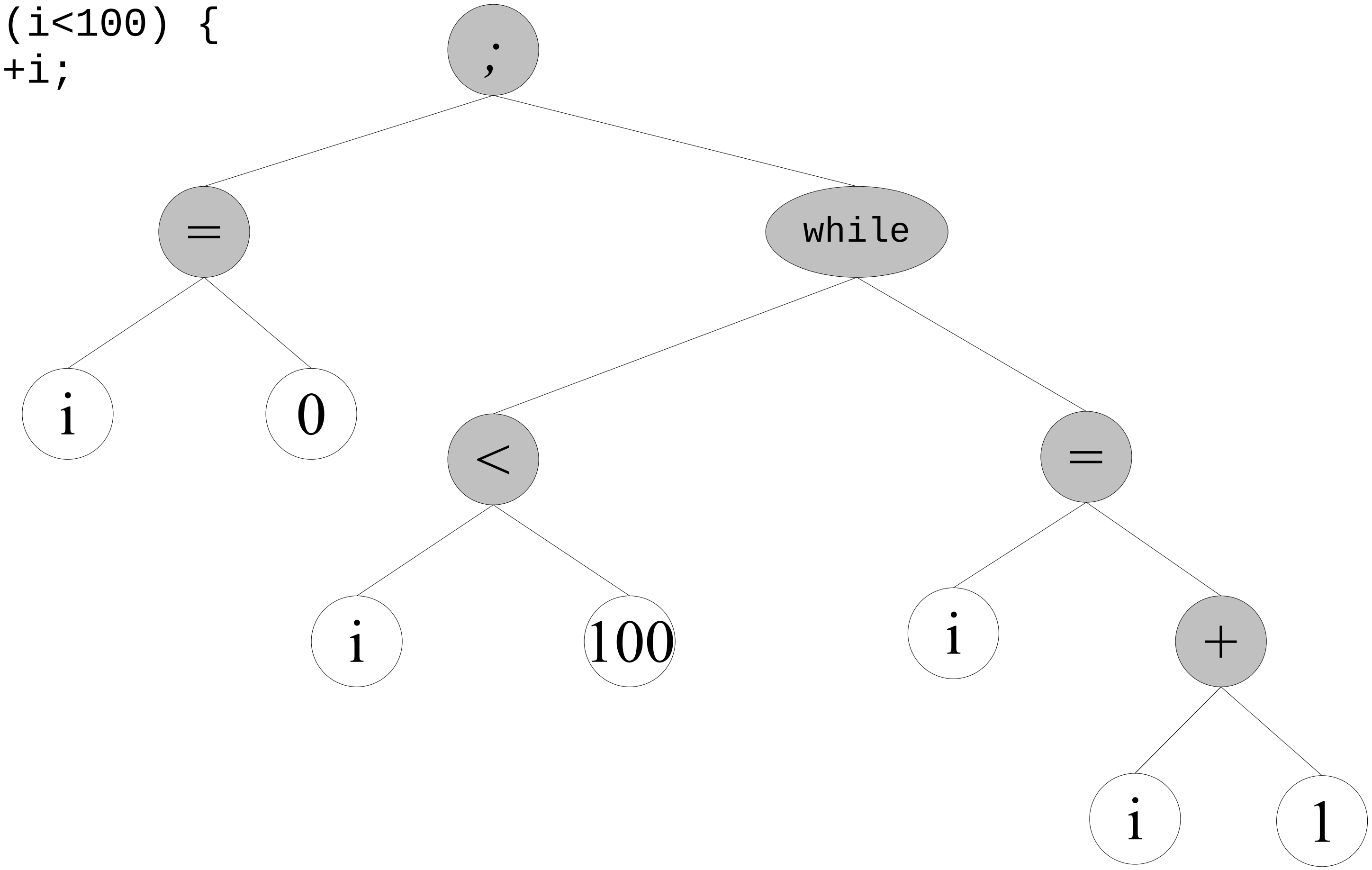
$$x + \frac{2}{y+3} x^2$$



$(p \wedge q) \Rightarrow (q \Leftrightarrow p)$



```
i=0
while (i<100) {
   i=i+i;
}
```



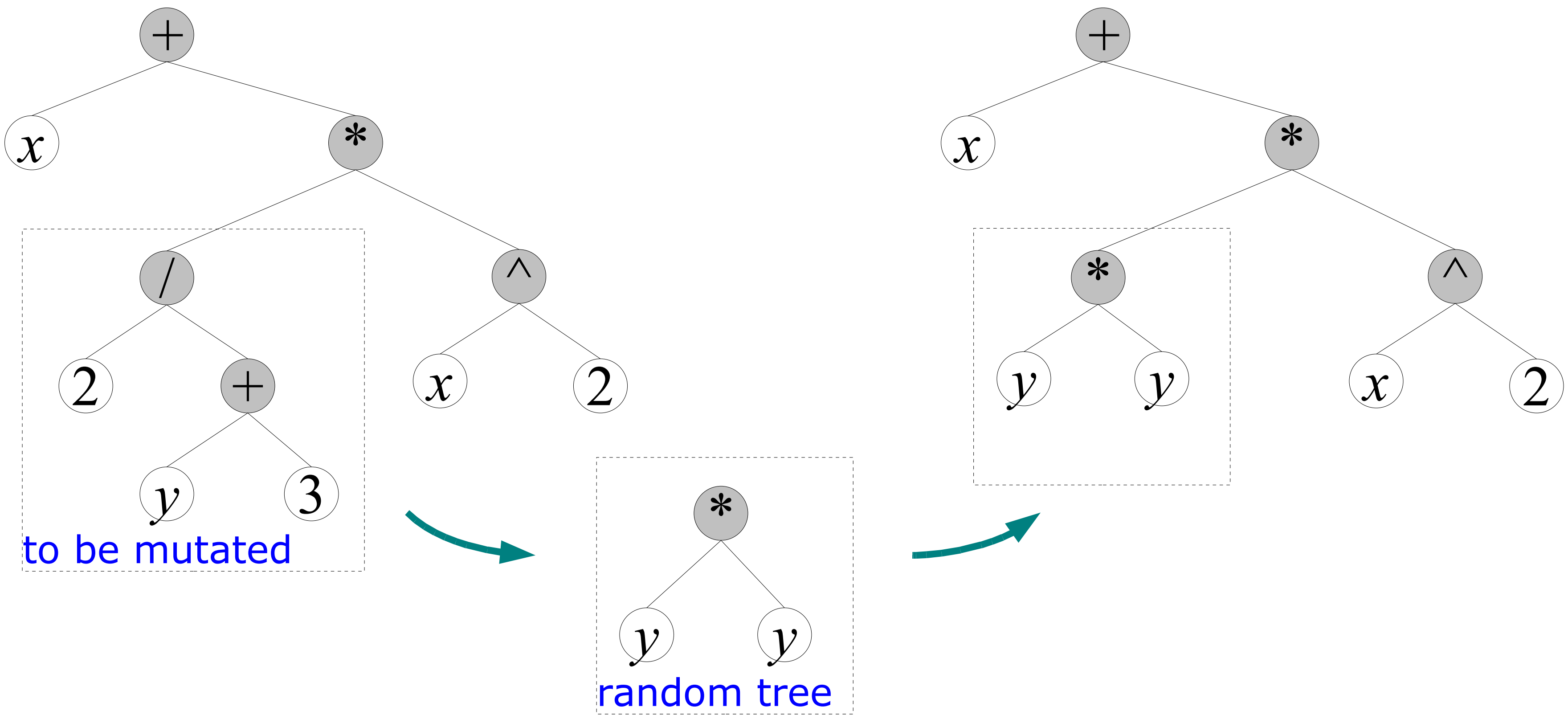
- $t \in T$ (terminals), $f \in F$ (functions)
  t is a correct expression
  $f(e_{1,} e_{2,} \ldots, e_n)$ is a correct expression iff $f \in F, arity(f) = n,$ and
  $\forall i, e_i$ is a correct expression

- Usually GP expressions are not typed. Any function get any other expression as parameters.
- *Strongly typed GP* is proposed to deal with such problems.

Genetic Process

- Initialization:
  - Involves creation of a tree population with internal function nodes and leaf terminal nodes randomly.
  - A depth limit is forced during random operation.
  - Generation of full trees vs growing trees. (regular shape vs rregular shape)
  - The ramped half-and-half method: use different tree depth classes, and for each class create half of the population full and half as grown.

- Mutation:
  - Pick a random node and its subtree and replace it with a random subtree.



- Crossover:
  - Pick a random subtree from each parent. Divide each into subtree and the root subtree partitions. Swap and combine each root subtree with the others subtree.

parents:

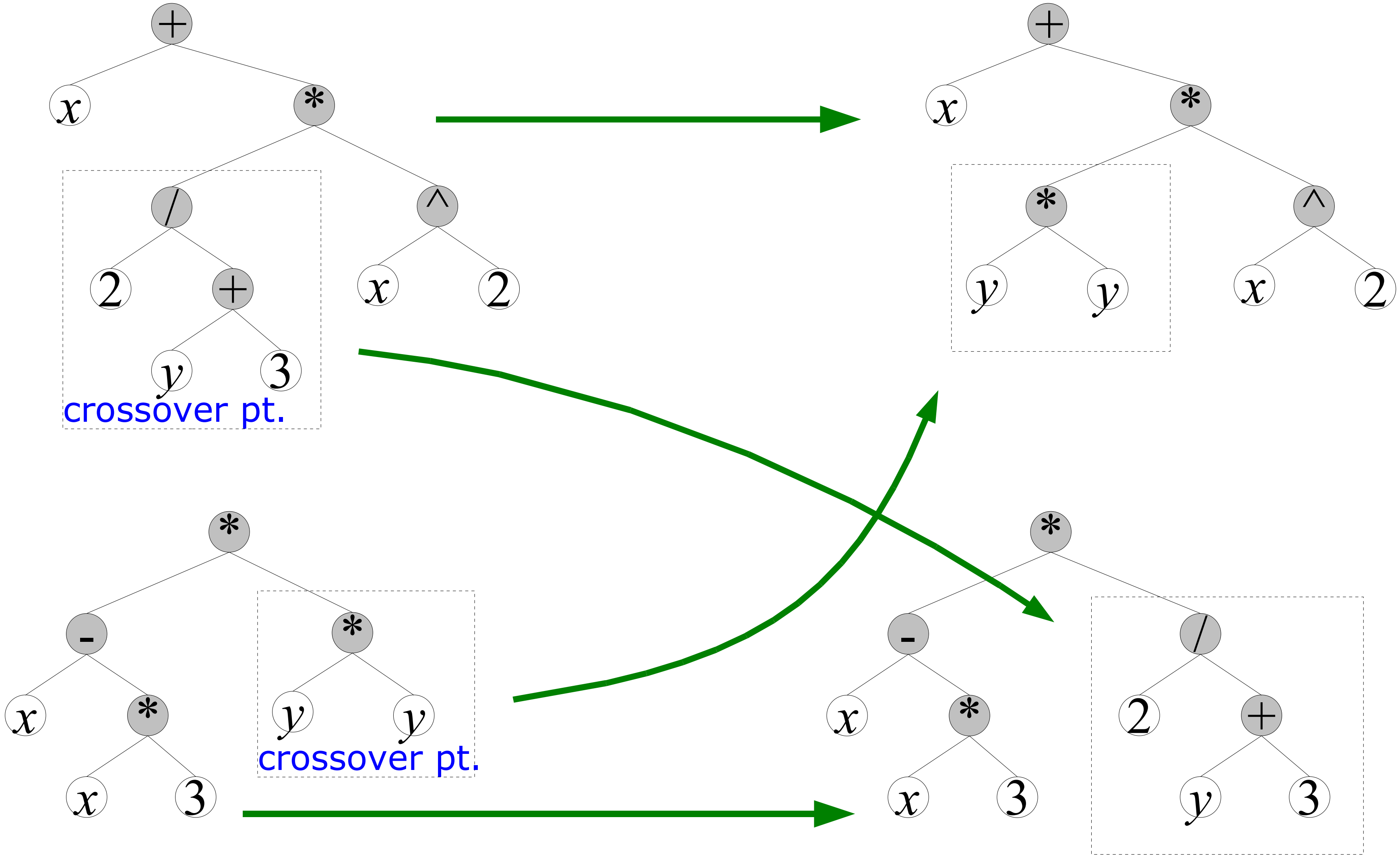
$$x+\frac{2}{y+3}\cdot x^2$$

$$(x-(x\cdot 3))\cdot(y\cdot y)$$

offsprings:

$$x+(y\cdot y)\cdot x^2$$

$$(x-(x\cdot 3))\cdot\frac{2}{y+3}$$



Linear Representation

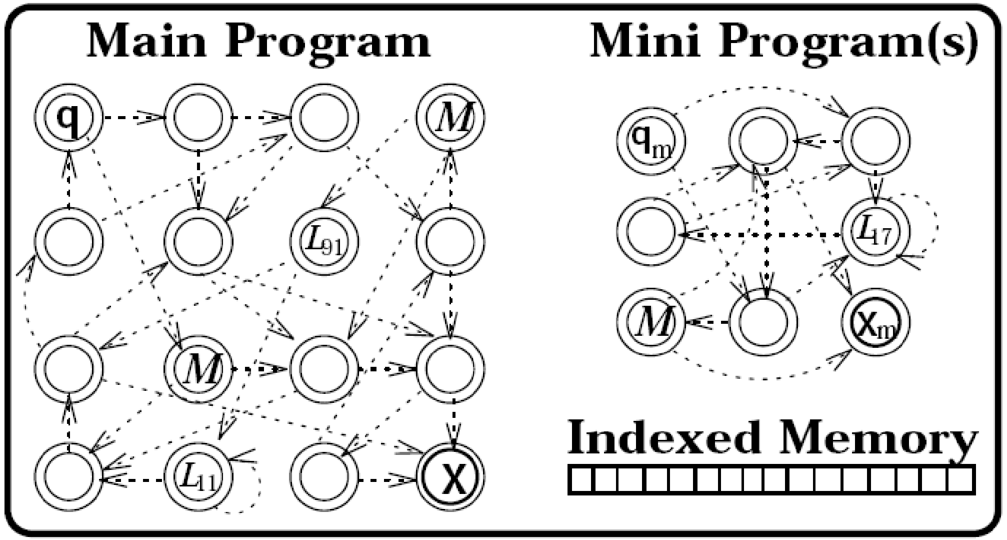
- Define an alphabet of machine codes for possible operations in the language. (AIMGP)
- An index memory or set of registers.
- Represent each program as a sequence of machine codes.
- Machine codes are executed with memory access to evaluate a genome.
- Use special mutations (operand/operator mutations) and standard crossover operation like any GA with integer representation.

Graph Representations (PADO)

- PADO (Parallel Algorithm Discovery Orchestration)
- A graph of nodes connected to each other with edges denoting program flow.
- Each node executes an instruction and based on some decision expression follows one of the outgoing edges.
- Indexed memory access.
- Special mutation and crossover operations are required.

- Stack based basic set of commands. Action set and branching decision to another node.

- Mini program and a library of Automatically Defined Functions



- Crossover: partition each parent into two classes. Mark internal, outgoing and incoming edges of each class. Crossover classes with preserving internal edges and combining incoming/outgoing edges.

Problems of GP

- Crossover and mutation can be too destructive.
- Uncontrolled tree growth 'Survival of the Fattest'
- Intelligent crossover operations proposed.
- Evaluating fitness: too slow, executing programs or simulations, halting problem.

Improving Crossover

- %75 of the offsprings fitnesses are less than half of the fitness of their parents!
- Biology:
  - Speciation, only species of same kind reproduce
  - Semantics preserving. A phenotype is crossovered with same phenotype.
  - Homologous, structure preserving. Gene positions are crossed-over at codons and gene boundaries are marked.
- Simple GP:
  - Any subtree is cross-overed with any other.
  - New subtree can be put in any context. No similarity.
  - No speciation. A random individual can be anything.

Brood Recombination

- Attehberg, 1994.
- Make N crossovers instead of 1. Take best 2 of 2N offsprings according to their fitnesses.
- Calculate N times more fitnesses but a higher chance to find a good crossover.

Intelligent Crossover

Different Tree Operators

- Mutations:
  - Point mutation: single node exchanged with a random terminal/non terminal value
  - Permutation: arguments of a terminal node is shuffled
  - Hoist: A subtree is taken as a whole individual
  - Expansion mutation: A terminal node is replaced with a random subtree.
  - Colapse subtree mutation: Subtree is replaced with a terminal.
  - Subtree mutation: A subtree is replaced by another random subtree.
  - Gene duplication: Subtree substituted for random terminal.

- Crossovers:
  - Subtree exchange: exchange two random subtrees from individuals.
  - Self crossover: exchange subtrees in the individual itself.
  - Module crossover: exchange modules between individuals.
  - Context-preserving crossovers: exchange subtrees if structure matches with some degree.

Improving the Evolvability

- Modularization: Logically closed entities working as a black box having interface with other modules.
- The solution consists of one or more modules and the main body.
- Modules -> building blocks.
- Shorter programs -> less destruction probability.
- Types:
  - Automatically Defined Functions (Koza, 1994)
  - Encapsulation (Koza, 1992)
  - Module Acquisition (Angeline & Pollack, 1992)

Automaticall Defined Functions

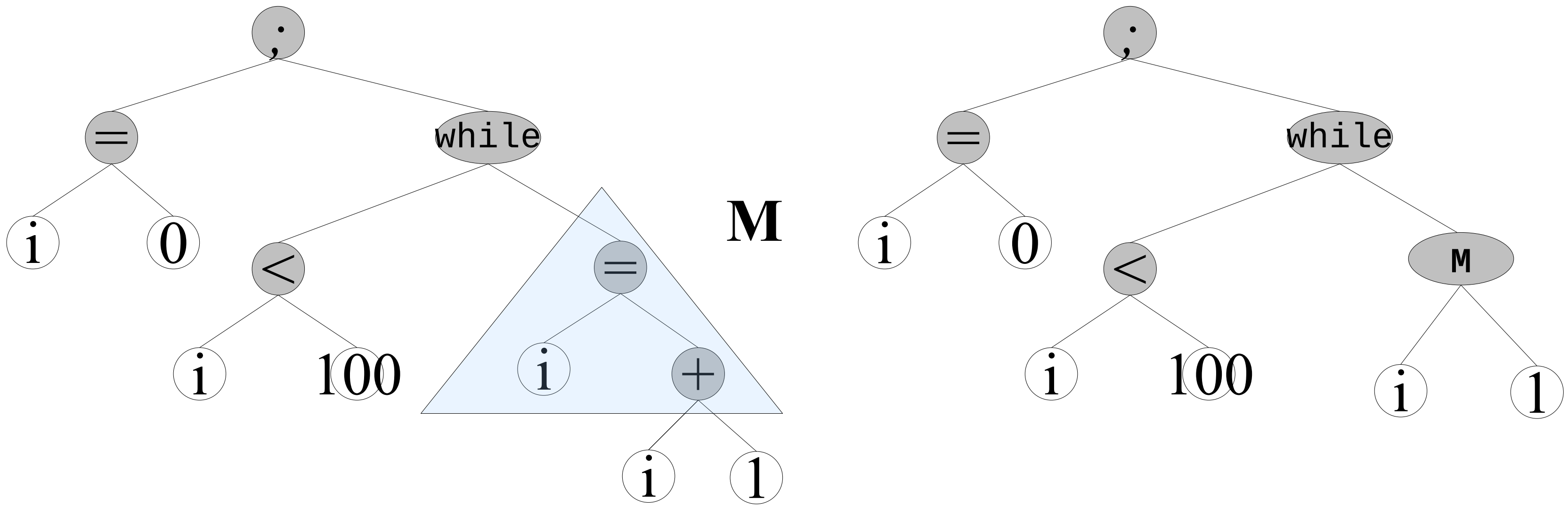
- Each individual consists of:
  – Result producing branch
  – Function defining branch
- Usually structure is fixed (number of functions/arities etc)
- Seperate genetic operators applied for function(s) and body in an isolated manner.
- Architecture altering methods can be applied but probably not useful.

Encapsulation

- Select a non-terminal node, define and bind it as a new terminal globally.
- If subtree contains usefull operations it is beneficial.

Module Acquisition

- A subtree is selected and part of the tree up to a depth level is defined as module. The part outside of this subtree is considered as arguments.
- Also referred to as compression.



Strongly Type Genetic Programming

- Programming Language analogy: a strict type system in representation helps in reproduction of valid individuals.
- Each subtree has a anotated type. Crossover and mutation preserve types.