CSE/ECE 848
Introduction to
Evolutionary Computation

Module 3 - Lecture 11 - Part 2
Genetic Programming - Operators

Wolfgang Banzhaf, CSE
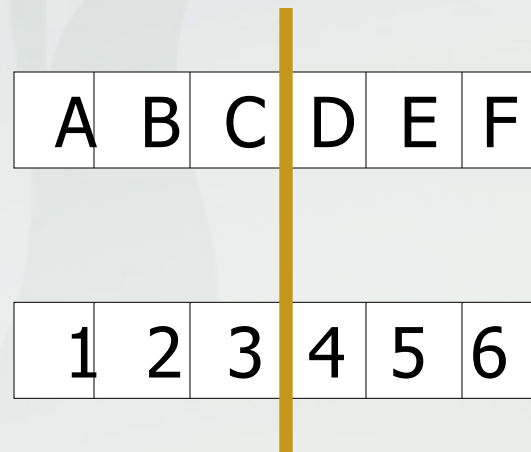**John R. Koza Chair in Genetic Programming**

CSE/ECE 848 Introduction to
Evolutionary Computation

Difference 4: Crossover, GA Style

- In GA, crossover selects the same random point on two strings, then recombines the strings into two new strings

A	B	C	D	E	F

1	2	3	4	5	6

CSE/ECE 848 Introduction to
Evolutionary Computation

Crossover, GP Style

- In GP, two different random points can be selected, and two new trees created

CSE/ECE 848 Introduction to
Evolutionary Computation
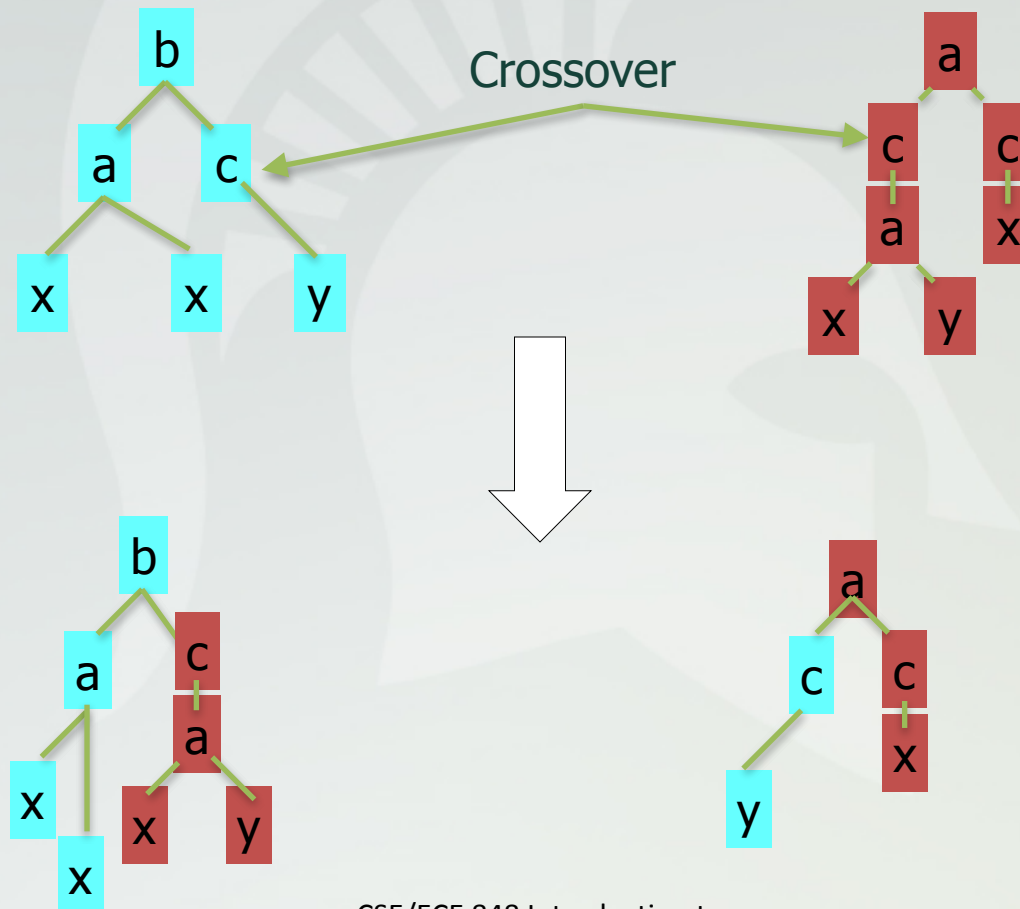
Crossover II

- Thus the functions must be carefully written so that any combination of functions produces an answer (for example, division by zero returns 0)

- This is called the Closure Principle

- As compared to GA crossover, GP crossover seems very disruptive (random insertion of a subprogram into a program at an arbitrary point)

- It generates many issues in GP

CSE/ECE 848 Introduction to
Evolutionary Computation

Difference 5: Mutation

- A GA simply selects a random position and changes the feature value there
- A GP selects a random tree position and grows a new tree there

Mutation point

CSE/ECE 848 Introduction to
Evolutionary Computation

Mutation II

- Typically a max depth is set for the size of the tree

- Further, restrictions can be placed on the kind of the tree that is created

- Also, the probability of mutation (as for crossover) can be adjusted to the depth of the tree (e.g. by treating nodes deeper in the tree differently)

CSE/ECE 848 Introduction to
Evolutionary Computation

Difference 6: Initialization

- In making an initial population of trees, you can make either full trees or grow trees

- Full trees are indeed full trees (all leaves at the same level)

- Grow trees are trees grown randomly from a root node, creating a random (non-full) shape

CSE/ECE 848 Introduction to
Evolutionary Computation
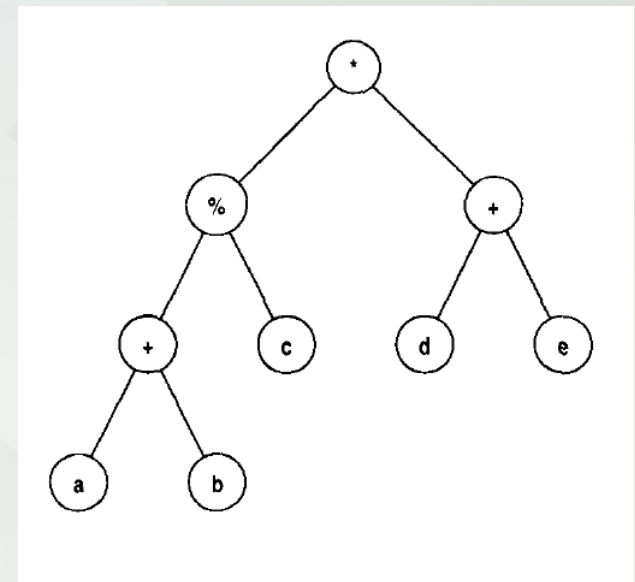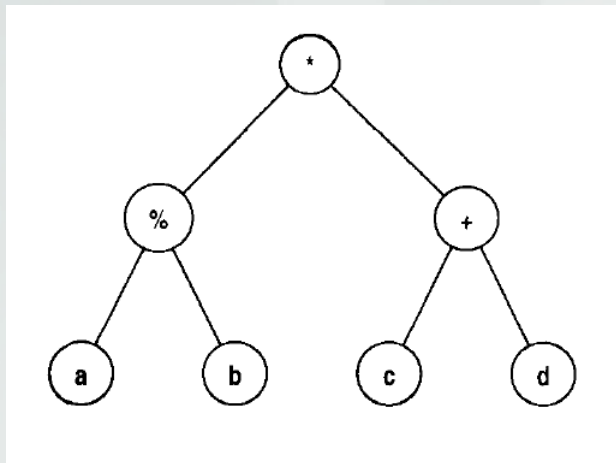
Initialization II

- To avoid prejudicing the initial population towards full or grow trees, you typically do ramped-half-and-half using user determined minDepth and maxDepth parameters.

- At each depth D between minDepth and maxDepth, create:

    - popsize/(maxDepth-minDepth)/2 full trees

    - popsize/(maxDepth-minDepth)/2 grow trees each of depth D.