UCD CASL

Complex & Adaptive Systems Laboratory

Dr. Michael O'Neill
Dr. Miguel Nicolau

Genetic Programming

COMP30290 Natural Computing
COMP41190 Natural Computing and Applications

Genetic Programming

Automatic Programming

- Assemblers;
- Compilers 2GL;
- Automatic Parallelisation.

Arthur Samuel, 1959:

"Tell the computer what to do, not how to do it."

Genetic Programming

John Koza's AP attributes...

- Start with **high-level problem description** that results in a solution in the form of a computer program;
- Automatically determine the program's **size and architecture**;
- Automatically organise a group of **instructions** so that they may be **re-used** by a program;
- **Problem-independence**;
- **Scalability** to larger versions of the same problem;
- Capability of producing **human competitive results**;
- Evolutionary Automatic Programming / Genetic Programming.

Genetic Programming

Individual **is** OR **represents/encodes** a program

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char* argv){
        float x=0.0, y=0.0, z=0.0;
        x=atof(argv[1]);
        y=atof(argv[2]);
        z=atof(argv[3]);
        x = 2.0*sin(y) + 4.0*sin(x);
        z = (x*x) + exp(z);
        printf("The answer is: z=%f\n",z);
        return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void turnLeft(float degrees);
void turnRight(float degrees);
void moveForward(float distance);

int main(int argc, char* argv){
        turnLeft(90);
        if(sensorValue(0) > 1000)
                moveForward(10);
        else
                turnRight(90);
        return(0);
}
```

Representation

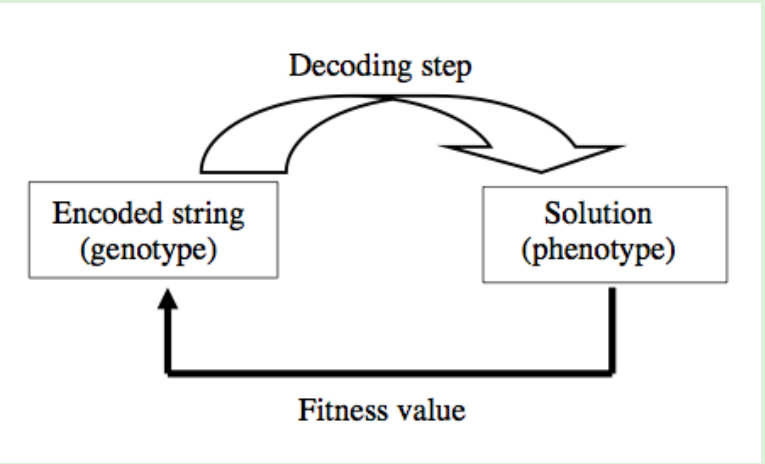
Recall GAs

- Binary string;
- Fixed-length chromosomes;
- Problem specific encoding.
- But how to encode a program?...

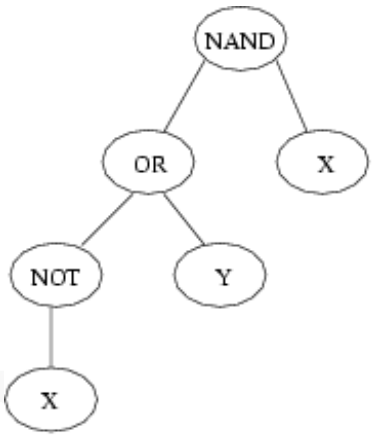
Representation

The Road to GP

- Historical Perspective on Representation
  - Friedberg (1958/1959);
  - Cramer (1985);
  - Koza (1989).
- Distinguishing features:
  - Variable-length;
  - GP operates on solution directly.



Representation

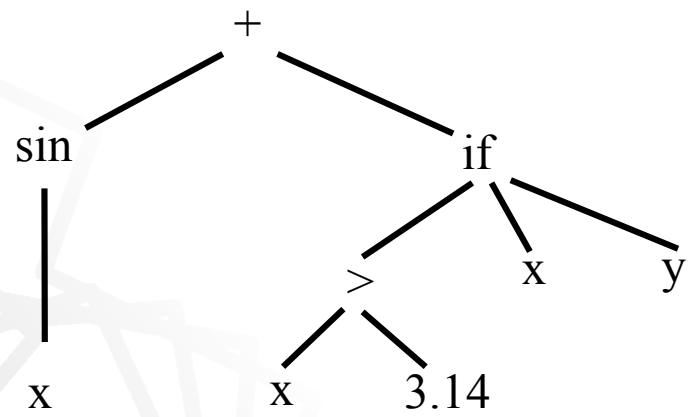


(NAND (OR (NOT X) Y) X)

GP trees

- Koza (1992) popularised Lisp S-expressions;
- Expressions (trees) generated from:
  - *Function Set*: boolean, arithmetic, loops, user-defined functions...
  - *Terminal Set*: inputs, constants, variables...

Representation



Different types of functions/terminals

- Conditionals;
- `(if (> x 3.14) x y)`
- `IF 'condition' THEN 'return X' ELSE 'return y'`

Representation

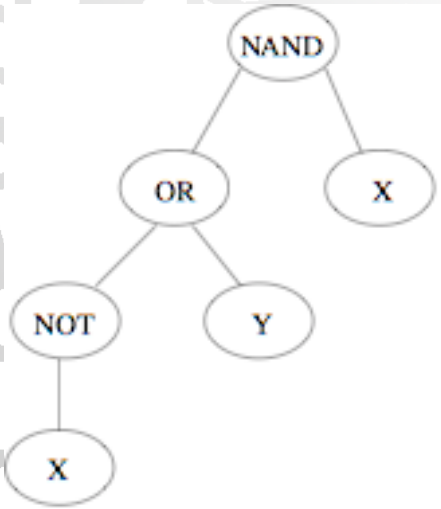
Features

- Sufficiency:
  - Function + Terminal sets: powerful enough to represent a solution.
- Parsimony:
  - Smaller Function + Terminal sets are better.
- Closure:
  - Each function should gracefully handle all values it ever receives;
  - `( / 5 0 )` ?!?

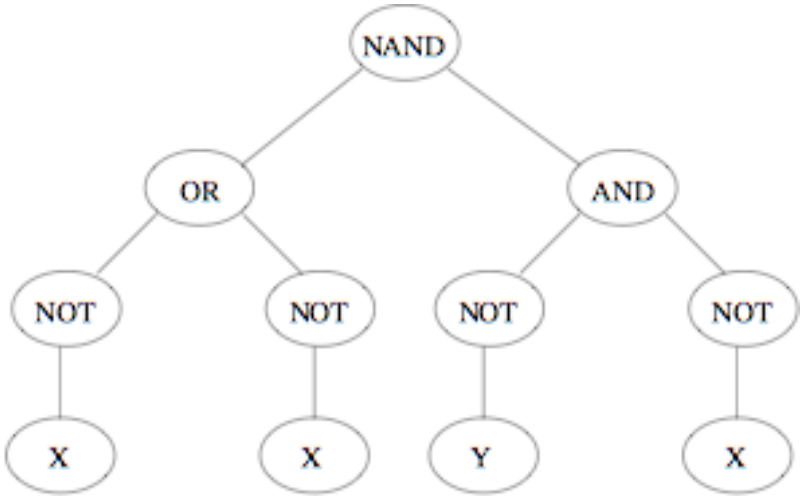
Initialisation

Tree Creation

- Max Depth - Maximum Program Size;
- **Grow** and **Full** initialisation;
- Desire structural diversity:
  - **Ramped half-and-half**.

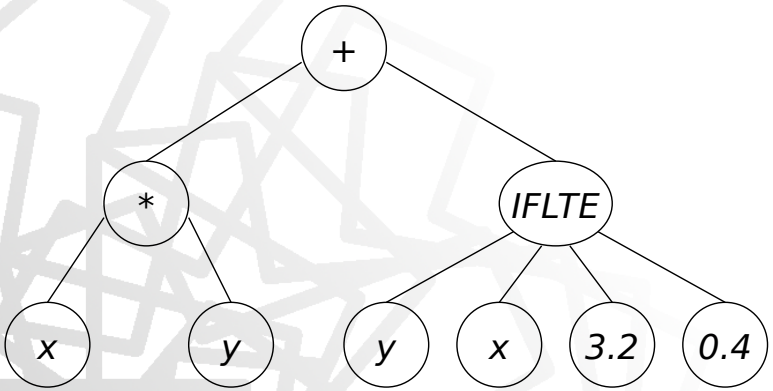


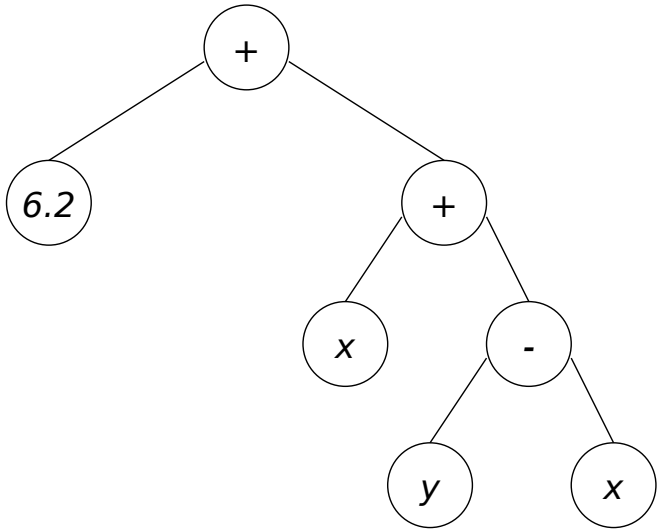
Grow – Depth 4



Full – Depth 4

Initialisation

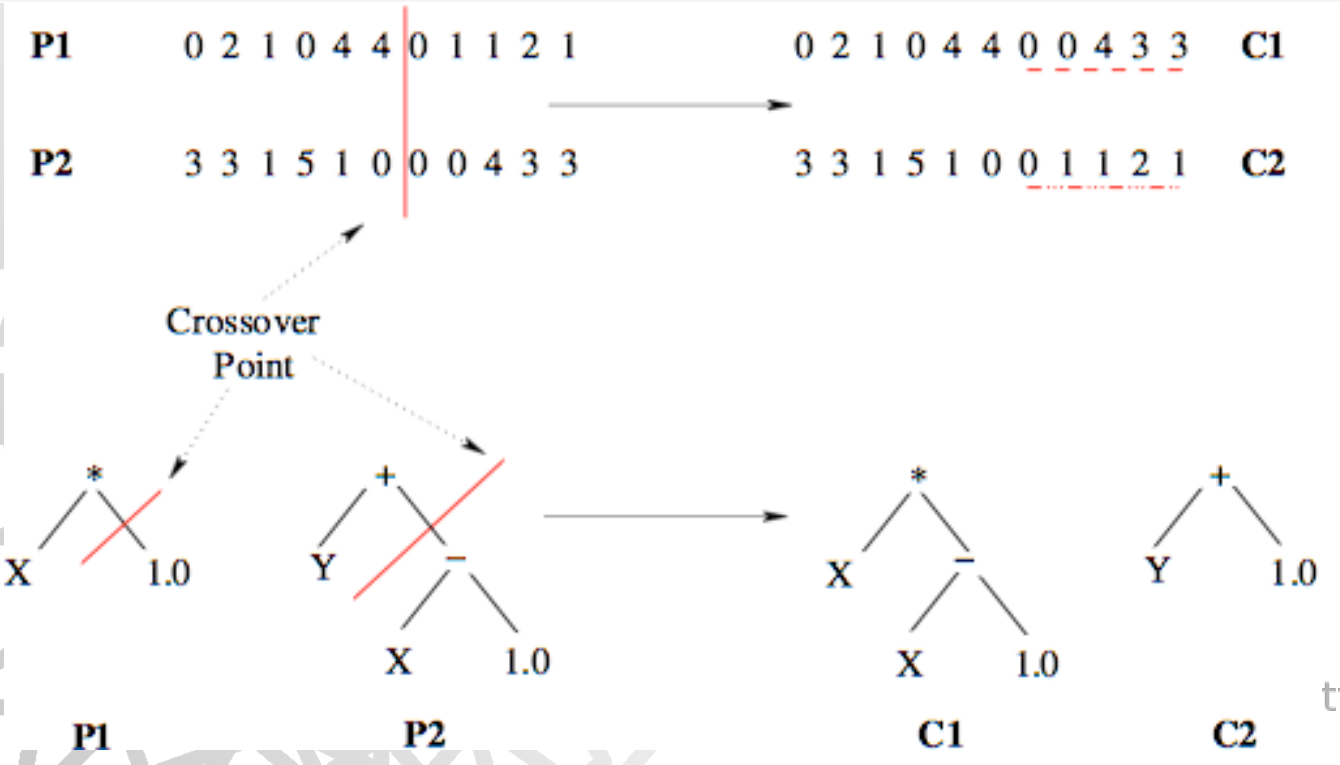




Genetic Operators

Operators

- Crossover;
- Mutation;
- Reproduction.

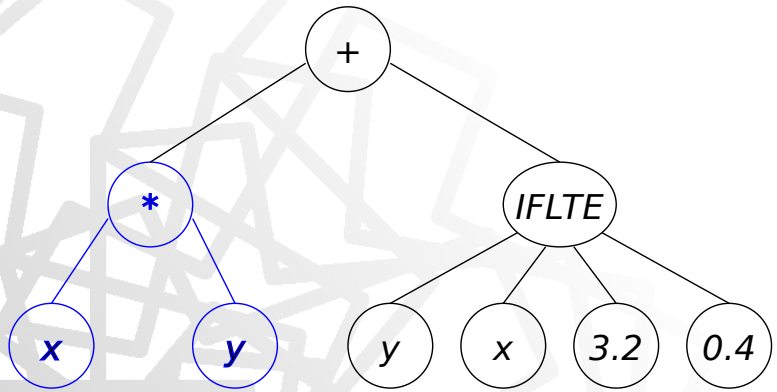


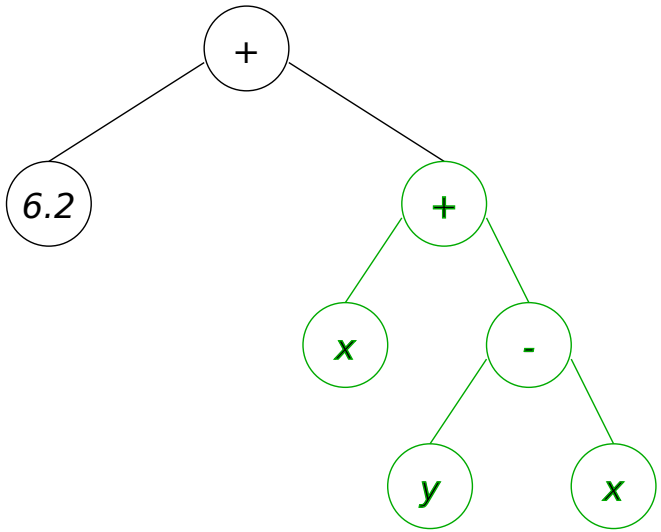
Genetic Operators

Crossover

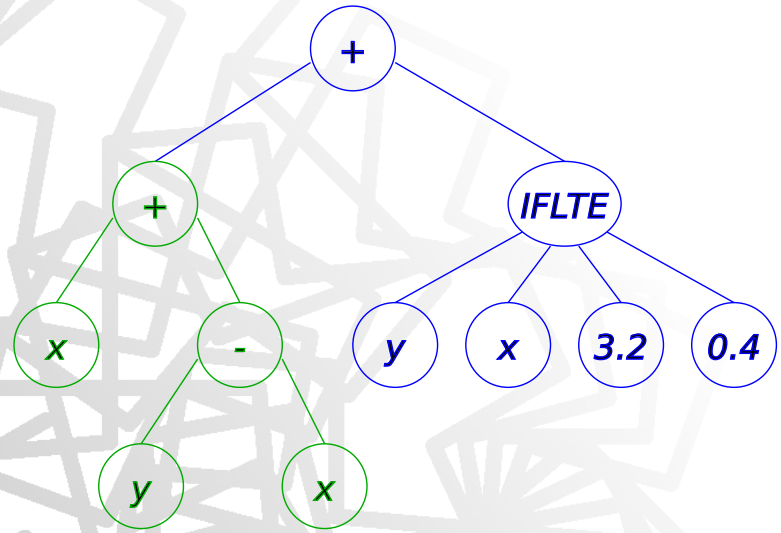
1. Select two parents;
2. For first parent, randomly pick node from 1 to n;
3. Independently pick node from 1 to n for second parent;
4. Swap sub-trees.

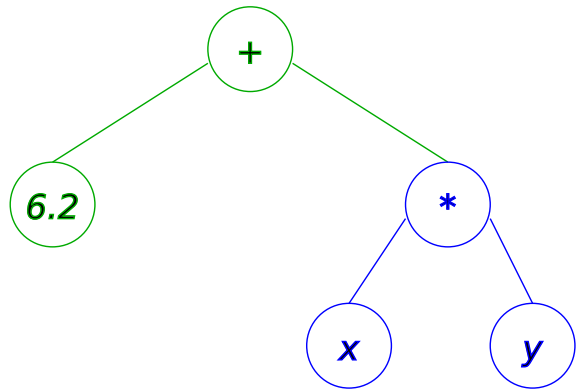
Crossover





Crossover



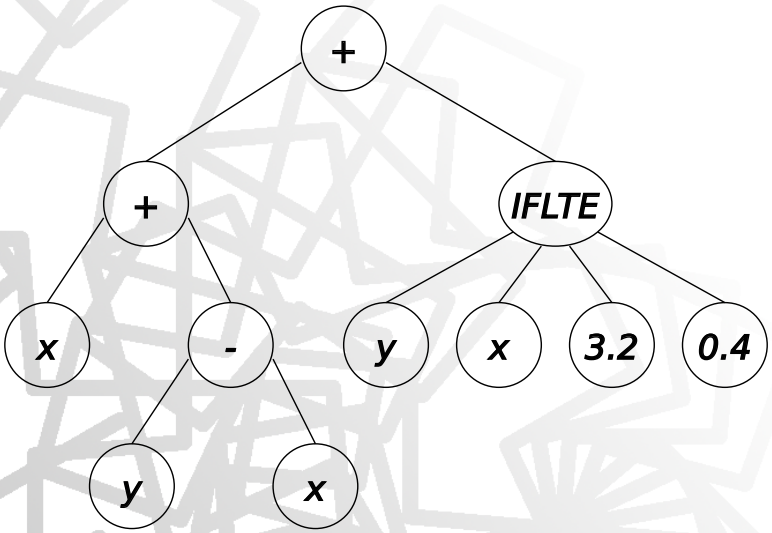


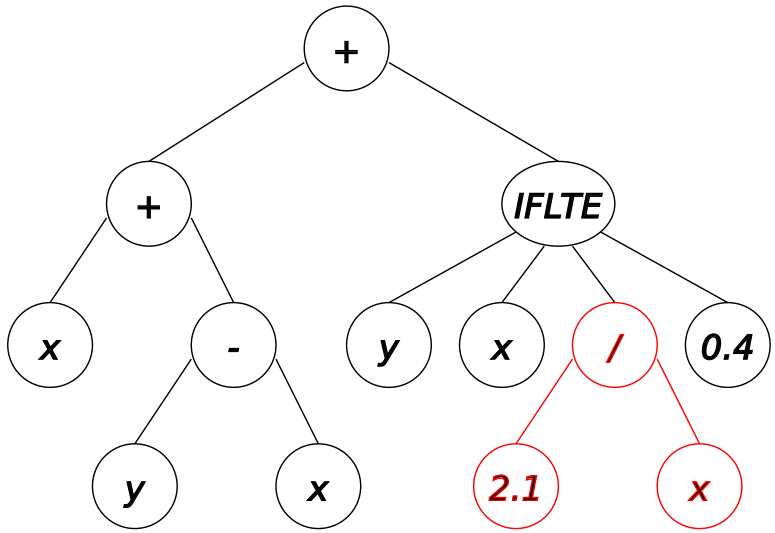
Genetic Operators

Mutation

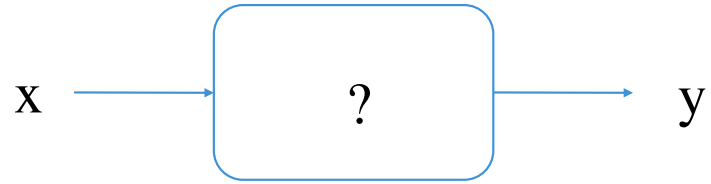
1. Randomly pick node from 1 to n;
2. Delete subtree at selected node;
3. Grow new subtree (as per initialisation).

Mutation





Parametrisation



Problem Definition

- Find function that gives output (y) for specific input (x);
- $y = f(x)$;
- E.g. $f(x) = x^2 + x$

Preparatory Steps

- Define function and terminal set;
- Define fitness measure;
- Define evolutionary parameters:
  - Max-depth, popsize, xover prob., mut. prob., reproduction prob., selection & replacement strategies, termination criteria.

Fitness Measure

Fitness Function

- Problem specific;
- Design to give graded and continuous feedback for selection:
  - Continuous fitness function;
  - Standardised fitness;
  - Normalised fitness.

Fitness Measure

$$f(x) = x^2 + x$$

Fitness Function

case #	Input	Output
1	1	2
2	2	6
3	4	20
4	7	56
5	9	90

- Simple discrete form: $f_p = \sum_{i=1}^{n} (p_i == o_i)$
- Simple continuous form: $f_p = \sum_{i=1}^{n} |p_i - o_i|$
- Mean Squared Error: $f_p = \frac{\sum_{i=1}^{n} (p_i - o_i)^2}{n}$

Fitness Measure

Fitness Function

case #	Input	Output (o_i)	Prediction (p_i)	Error	Squared Error
1	1	2	1	1	2
2	2	6	4	2	4
3	4	20	16	4	16
4	7	56	49	7	49
5	9	90	81	9	81
				23	151 / 5 = 30.2

- Different fitness functions = different fitness landscapes;
- Co-evolution:
  - No explicit fitness value;
  - Fitness relative to other individuals.
- Multi-objective fitness functions.

Next Classes

- Notes on Selection (available online).

- Lecture Tuesday 24th September 15h - 16h;
  - GP applications/variations;
  - Structure of 2-page proposal.