
INTRODUCTION TO GENETIC PROGRAMMING (PART 2)

Luca Manzoni

LINEAR GP

MOTIVATIONS

- Trees are not the only way of representing programs
 - Also streams of instructions can represent programs
 - Instead of LISP-like structure we now use assembly-like commands
 - Linear GP: a linear stream of “assembly-like” instructions
-

AN EXAMPLE OF LINEAR GP



Registers of a virtual
(or real!) machine

Add R1, R2, R1

Sub R3, R1, R4

Add R4, R3, R2

Mul R1, R1, R2

A Linear GP individual:
a list of instructions for the machine
communicating via registers

LINEAR GP AND GA

- Linear GP seems pretty similar to standard GA

↳ YOU HAVE A SEQUENCE OF INSTRUCTIONS AND ALSO MUTATION IS SIMILAR

- Except that the individuals can be of non-fixed length

- An important difference is that we are evolving programs and we “execute” the individuals

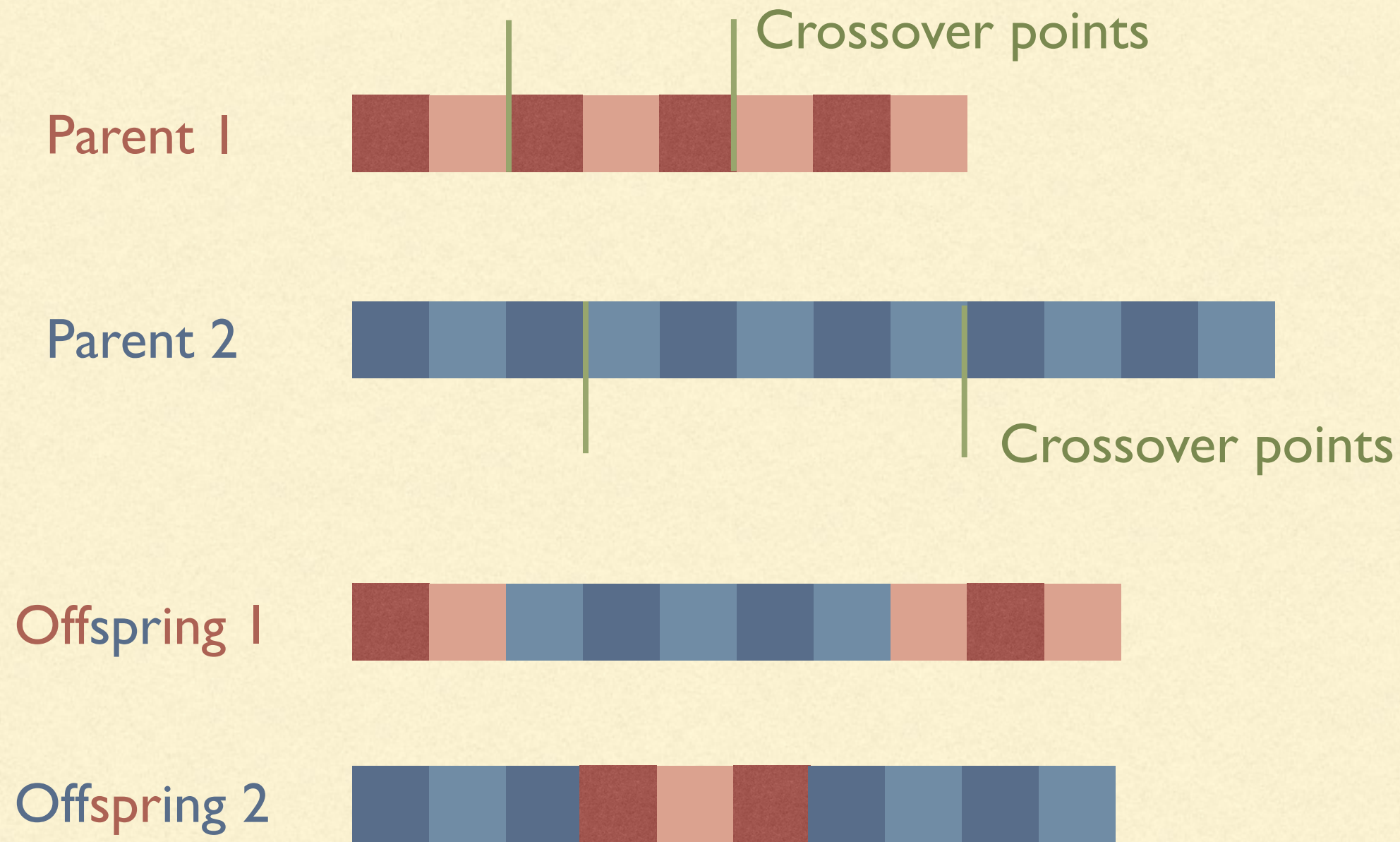
IN GA YOU CAN FIND SEQUENCE OF NUMBERS OR SYMBOLS;
IN GP YOU HAVE SEQUENCE OF SYMBOLS BUT YOU INTERPRET IT AND EXECUTE CODE.

- Most of the operators of GA can be used for Linear GP

- Two points crossover (with possibly different crossover points between the two individuals) is usually employed
-

TWO-POINTS CROSSOVER

LITERALLY WHAT IT MEANS



CARTESIAN GP IS NOT A PHILOSOPHYICAL INTERPRETATION OF THE SUBJECT

An artesian GP is a type of mathematical model that combines genetic programming (GP) with Cartesian genetic programming (CGP). Cartesian GP, like regular GP, is a type of evolutionary algorithm used to evolve computer programs. However, in Cartesian GP, the programs are represented as linear sequences of instructions, similar to a physical circuit layout.

Artesian GP adds an additional layer of complexity by incorporating the concept of water flow in an artesian well. In this model, information flows both vertically and horizontally through the program structure, allowing for more intricate and flexible program designs.

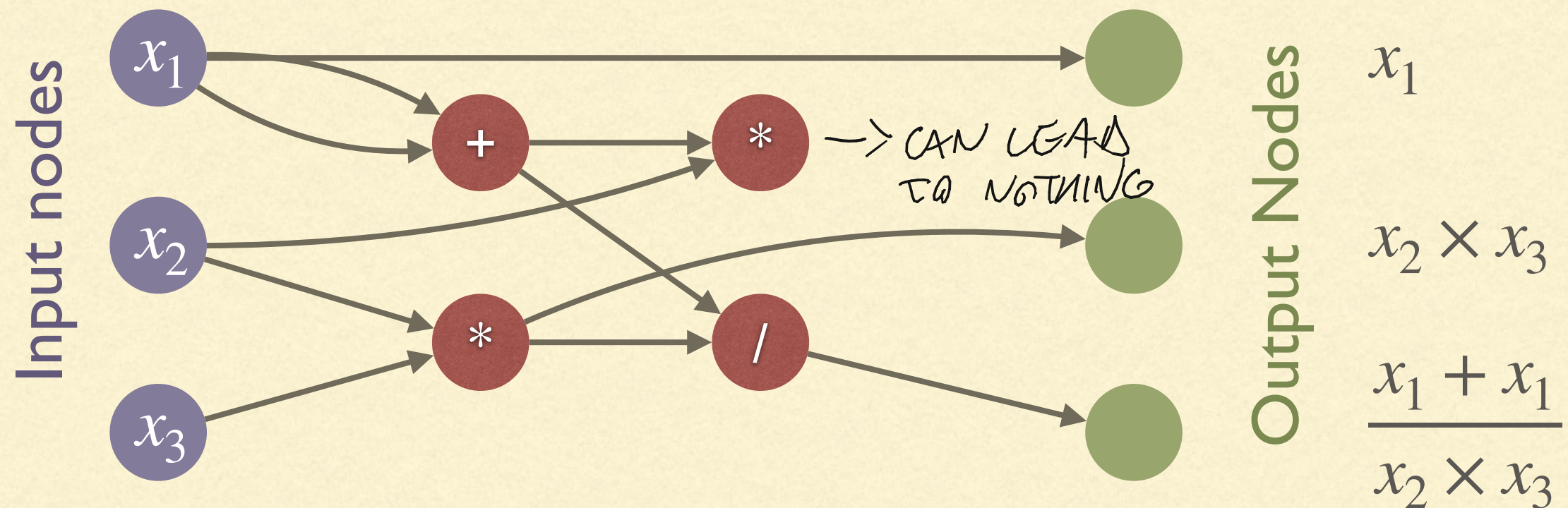
Overall, artesian GP aims to create more efficient and adaptable computer programs by utilizing the unique characteristics of Cartesian GP and incorporating them into a more dynamic and fluid evolutionary process.

MOTIVATIONS

- It is possible to represent programs as circuits/graphs
 - Cartesian GP represents individuals in this way (invented by Julian F. Miller)
 - Naturally suited for problems with many inputs and many outputs (instead of using multiple trees)
 - Has some tracts in common with linear GP (the representation for the circuit/graph is encoded in a linear way)
-

AN INDIVIDUAL OF CGP

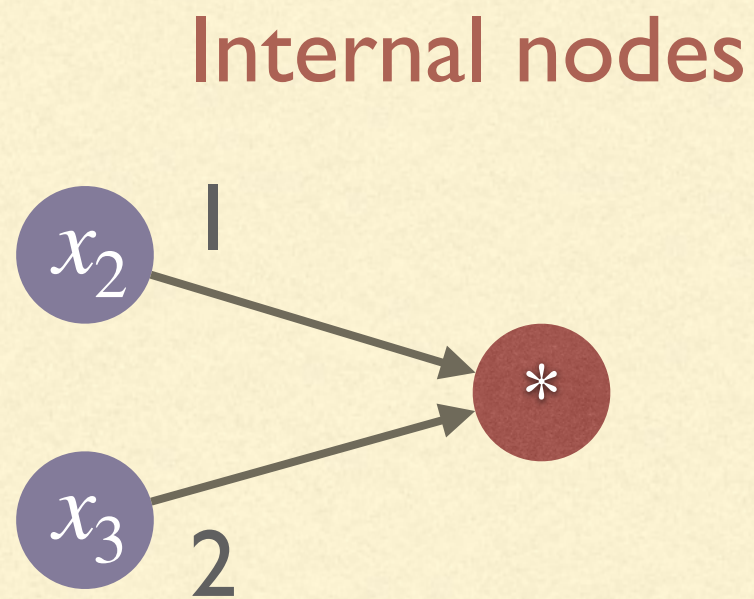
A COMPUTATIONAL GRAPH



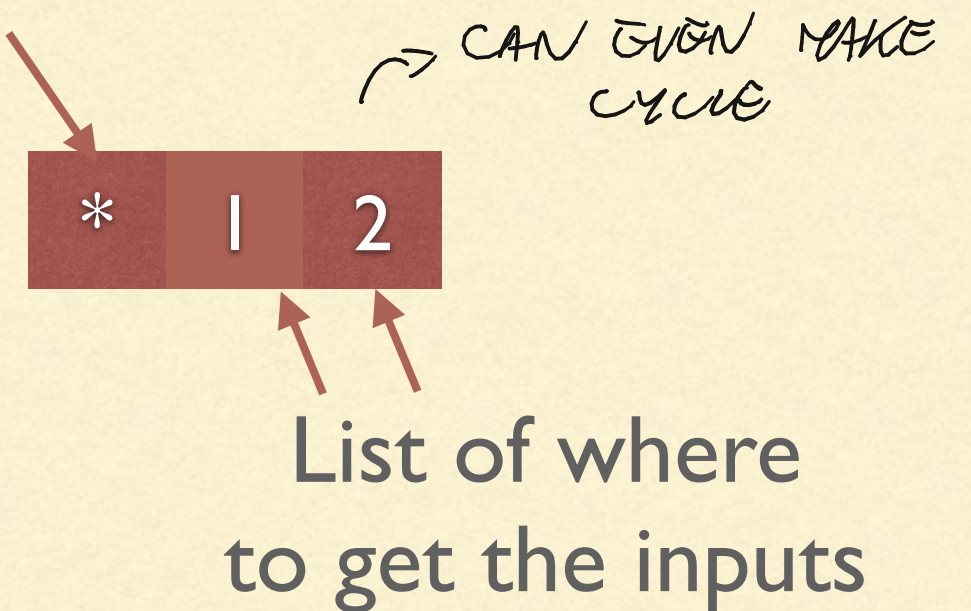
But how is the individual encoded?

\hookrightarrow IT DEPENDS ON
THE KIND OF THE NODE

ENCODING IN CGP



Function



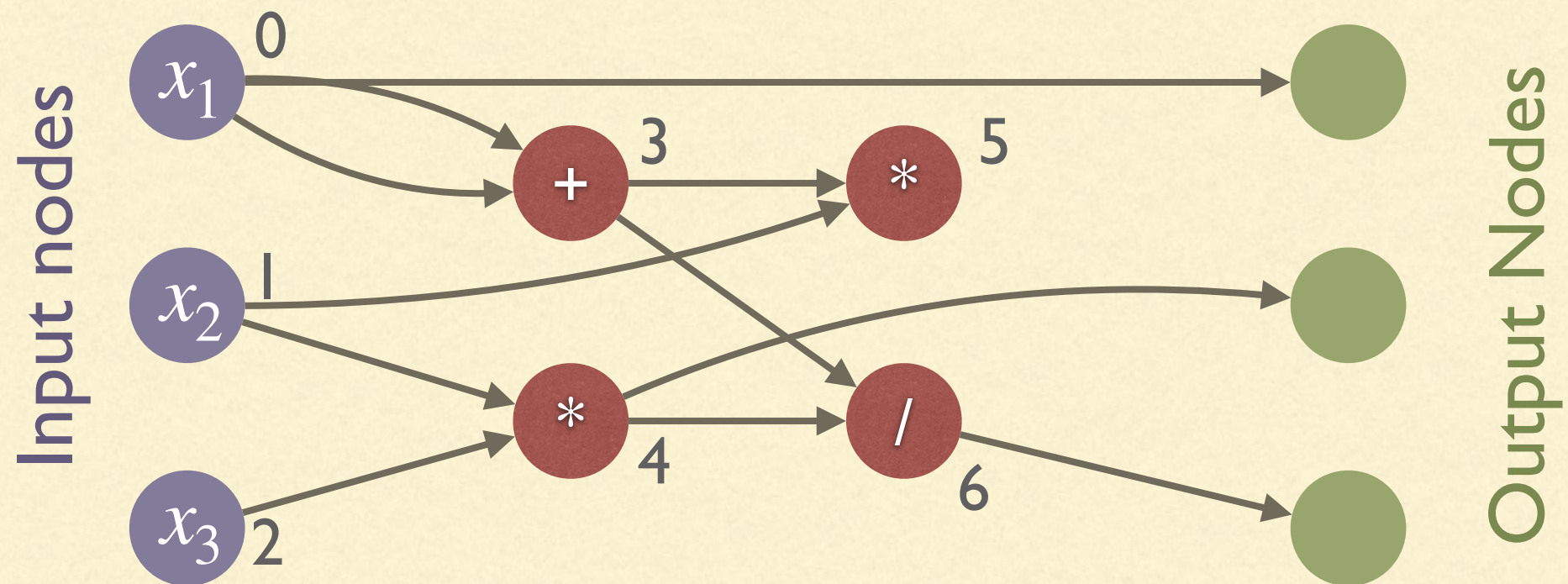
Output nodes



6

Where to get the output

AN INDIVIDUAL OF CGP



JUST A CASE FOR THE NODE



CGP MUTATION



Mutation point

A connection gene \longrightarrow Random valid connection



Mutation point

A function gene \longrightarrow Random function

GRAMMATICAL EVOLUTION

ANOTHER WAY OF PROGRAMMING GP

WHY?

FOR, IF, WHILE, ...
↗

- GP trees allow to express constructs of high-level languages
 - A linear representation is easier to manage
 - Grammatical Evolution combines a linear genome with a tree-based representation (in an indirect way)
 - The idea is to define a *grammar* that is then used to interpret a linear genome as a derivation of that grammar
-

BACKUS NAUR FORM (BNF)

- Invented by John Backus and refined by Peter Naur
 - Used to define context-free grammars, initially for the ALGOL programming language
 - It expresses a language as a set of production rules
 - Each rule maps a non-terminal to a string of zero or more non-terminals and terminals
-

BACKUS NAUR FORM (BNF)

- A grammar in BNF is a quadruple (T, N, P, S) where:
 - T is a set of terminal symbols
 - N is a set of non-terminal symbols
 - P is a set of production rules
 - $S \in N$ is the axiom, i.e., the start symbol
-

AN EXAMPLE OF BNF

Terminal symbols

$$T = \{+, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Non-terminal symbols

$$N = \{S, C, D\} \quad \text{Axiom } S$$

What kind of strings
can be defined with
this grammar?

Production Rules

$$S \rightarrow S + S \mid C$$

$$C \rightarrow D \mid DC$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

AN EXAMPLE OF BNF

$S \rightarrow S + S \rightarrow C + S \rightarrow DC + S \rightarrow 4C + S \rightarrow 4D + S$

$\rightarrow 42 + S \rightarrow 42 + S + S \rightarrow 42 + C + S \rightarrow 42 + D + S$

$\rightarrow 42 + 8 + S \rightarrow 42 + 8 + C \rightarrow 42 + 8 + D \rightarrow 42 + 8 + 6$

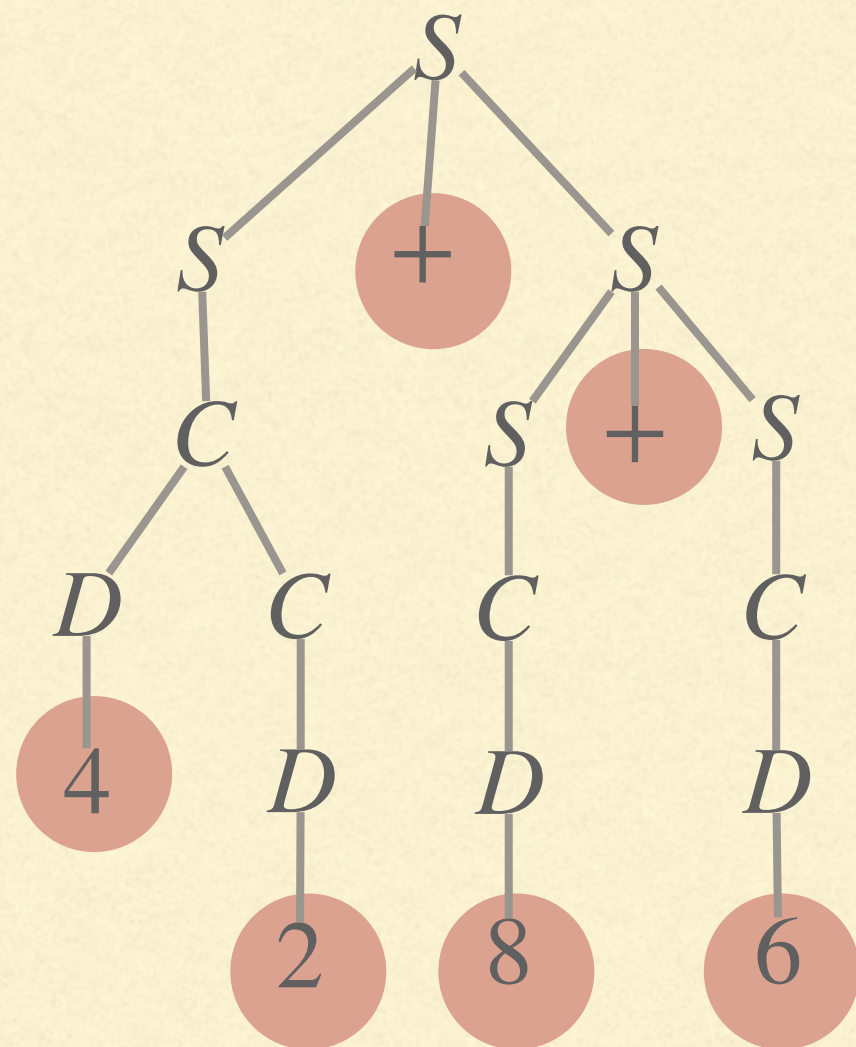
Production Rules

$S \rightarrow S + S \mid C$ *IN PYTHON* $S \rightarrow \text{FOR}, x =, \text{WHILE}, \text{IF}, \dots$

$C \rightarrow D \mid DC$

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

AN EXAMPLE OF BNF



The expansion can also be seen as building a tree

All the non-terminal are internal nodes

The terminals are the leaves

The resulting string is obtained by reading all the leaves in order

AN EXAMPLE OF BNF

- With the previous grammar we can, depending on the expansion that we select, generate all strings that represents the sum of integers numbers
 - Notice that if we want to obtain the result of the sum we need to evaluate the string
 - This means that if we can have a grammar expressing a program (with variables) and evaluate it on a specific input
-

FROM BNF TO GE

How To Move From Grammar To Grammatical Evolution

- What if we want to generate a specific string?
 - We can keep track of which expansion to select among all the possible ones for a non-terminal symbol
 - Something like:
 - Select the expansion number 1, then number 3, then number 7, then number 2, etc.
 - Seems like we can have a vector (1,3,7,2,...)
-

FROM BNF TO GE

- Once we have a vector of integers we can use GA-like evolution!
 - The genome is a sequence of integers
 - To evaluate an individual we need to generate a string via a grammar...
WE JUST ADD ONE STEP
 - ...and then evaluate the string
 - We are sure that the string is a valid program because it was generated by a grammar!
-

TOWARDS GE

$$S \xrightarrow{0 \mid} S + S \mid C$$

We number the RHS
of all production rules

$$C \xrightarrow{0 \mid} D \mid DC$$

But not all production rules
have the same number of RHS!

$$D \xrightarrow{0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9} 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Should our genome be in $\{0,1\}^n$ or in $\{0,1,\dots,9\}^n$?

TOWARDS GE

Should our genome be in $\{0,1\}^n$ or in $\{0,1,\dots,9\}^n$?

$$\begin{array}{cc} 0 & | \\ S \rightarrow S + S & | C \end{array}$$

Neither, we can use any integer!
Simply take it modulus the number of RHS

$$\begin{array}{cc} 0 & | \\ C \rightarrow D & | DC \end{array}$$

Mod 2 = 1

7548397623

$$\begin{array}{cccccccccc} 0 & | & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ D \rightarrow 0 & | & 1 & | & 2 & | & 3 & | & 4 & | & 5 & | & 6 & | & 7 & | & 8 & | & 9 \end{array}$$

AN EXAMPLE

$$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$$

$$OP \rightarrow + \mid \times \mid - \mid \div$$

$$V \rightarrow x_1 \mid x_2 \mid x_3$$

$$N \rightarrow - \text{ } DD \mid - \text{ } D \mid D \mid DD$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

3	5	22	6	9	2	56	18	24	1
---	---	----	---	---	---	----	----	----	---

AN EXAMPLE



$$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$$

S OP S

$$OP \rightarrow + \mid \times \mid - \mid \div$$

$$V \rightarrow x_1 \mid x_2 \mid x_3$$

$$N \rightarrow -DD \mid -D \mid D \mid DD$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$



3	5	22	6	9	1	56	18	24	1
---	---	----	---	---	---	----	----	----	---

0

AN EXAMPLE



$$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$$

$$OP \rightarrow + \mid \times \mid - \mid \div$$

$$V \rightarrow x_1 \mid x_2 \mid x_3$$

$$N \rightarrow -DD \mid -D \mid D \mid DD$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$S \text{ } OP \text{ } S$$



$$N \text{ } OP \text{ } S$$



3	5	22	6	9	1	56	18	24	1
---	---	----	---	---	---	----	----	----	---

0 2

AN EXAMPLE

$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$

$OP \rightarrow + \mid \times \mid - \mid \div$

$V \rightarrow x_1 \mid x_2 \mid x_3$

$N \rightarrow -DD \mid -D \mid D \mid DD$

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$N \text{ } OP \text{ } S$



$D \text{ } OP \text{ } S$



3	5	22	6	9	1	56	18	24	1
---	---	----	---	---	---	----	----	----	---

0 2 2

AN EXAMPLE

$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$

$OP \rightarrow + \mid \times \mid - \mid \div$

$V \rightarrow x_1 \mid x_2 \mid x_3$

$N \rightarrow -DD \mid -D \mid D \mid DD$

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$D \text{ } OP \text{ } S$



$5 \text{ } OP \text{ } S$



3	5	22	6	9	1	56	18	24	1
---	---	----	---	---	---	----	----	----	---

0 2 2 6

AN EXAMPLE

$S \rightarrow (S \text{ OP } S) \mid V \mid N$

$OP \rightarrow + \mid \times \mid - \mid \div$


$V \rightarrow x_1 \mid x_2 \mid x_3$

$N \rightarrow -DD \mid -D \mid D \mid DD$

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$5 \text{ OP } S$

\downarrow
 $5 \times S$



3	5	22	6	9	1	56	18	24	1
0	2	2	6	1					

AN EXAMPLE



$$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$$

$$OP \rightarrow + \mid \times \mid - \mid \div$$

$$V \rightarrow x_1 \mid x_2 \mid x_3$$

$$N \rightarrow -DD \mid -D \mid D \mid DD$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$5 \times S$$



$$5 \times V$$



3	5	22	6	9	1	56	18	24	1
---	---	----	---	---	---	----	----	----	---

0 2 2 6 1 1

AN EXAMPLE

$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$

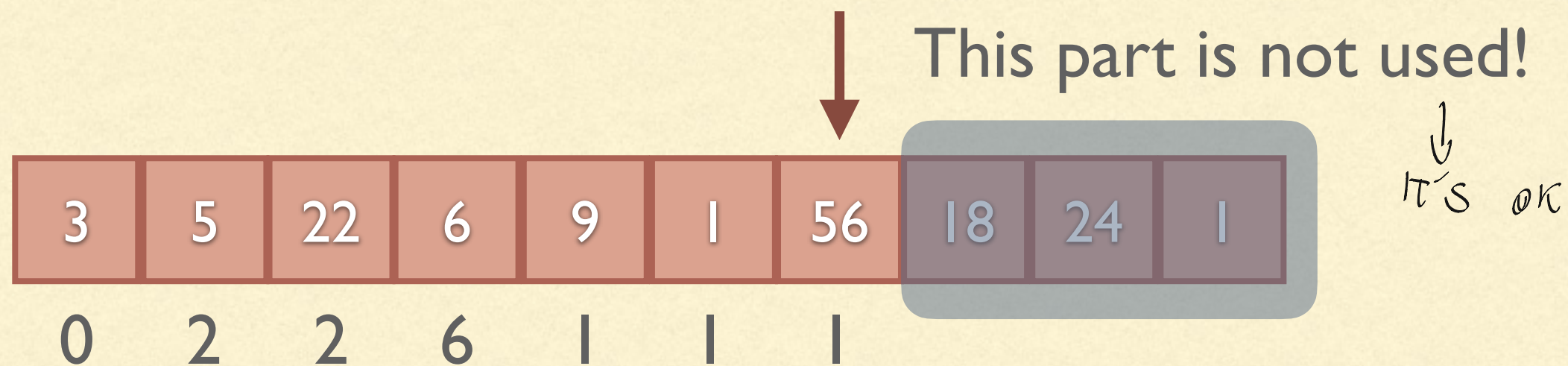
$OP \rightarrow + \mid \times \mid - \mid \div$

$V \rightarrow x_1 \mid x_2 \mid x_3$

$N \rightarrow -DD \mid -D \mid D \mid DD$

$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$5 \times V$
 \downarrow
 $5 \times x_2$



GE PECULIARITIES

- The individuals might not be all of the same length
 - The effective length of an individual can be shorter than the length of the genome (previous example)...
 - ...but it can also be longer (in this case we “wrap around” and continue from the beginning)
 - We must have a maximum length for the expansion!
-

AN EXAMPLE



$$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$$

S OP S

$$OP \rightarrow + \mid \times \mid - \mid \div$$

$$V \rightarrow x_1 \mid x_2 \mid x_3$$

$$N \rightarrow -DD \mid -D \mid D \mid DD$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$



0

0

AN EXAMPLE



$$S \rightarrow (S \text{ } OP \text{ } S) \mid V \mid N$$

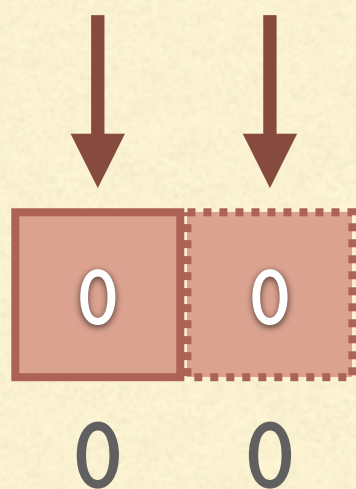
$$OP \rightarrow + \mid \times \mid - \mid \div$$

$$V \rightarrow x_1 \mid x_2 \mid x_3$$

$$N \rightarrow -DD \mid -D \mid D \mid DD$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$\begin{array}{c} S \text{ } OP \text{ } S \\ \downarrow \\ (S \text{ } OP \text{ } S) \text{ } OP \text{ } S \end{array}$$



The expansion will never end!

We will simply assign a very bad fitness to this individual

GE PECULIARITIES

- Notice that there is no one-to-one mapping between genotype and the phenotype
 - Multiple genotypes can be expanded to represent the same derivation tree/phenotype
 - Notice that the “meaning” of a part of an individual depends from all the genes before it (since they define in which part of the derivation we are)
-

GE: OPERATORS

- We can use the same operators as GA for crossover and mutation
 - We can also use specific operators for crossover:
 - Sensible crossover is a one-point crossover that uses the actual length of an individual
↳ SEARCH THE USED PART
 - Ripple crossover
 - Homologous crossover → ...
-

HOMOLOGOUS CROSSOVER

- In homologous crossover we align the history of rule activations in two individuals
- We perform a one-point crossover in the area where they differ



LIBRARIES

- Grammatical evolution is implemented in PonyGE2 library
 - <https://github.com/PonyGE/PonyGE2>
 - You need to clone/download the repository (you cannot use pip or anaconda for the installation)
-