

# Algoritmos Evolucionários baseados em Gramáticas

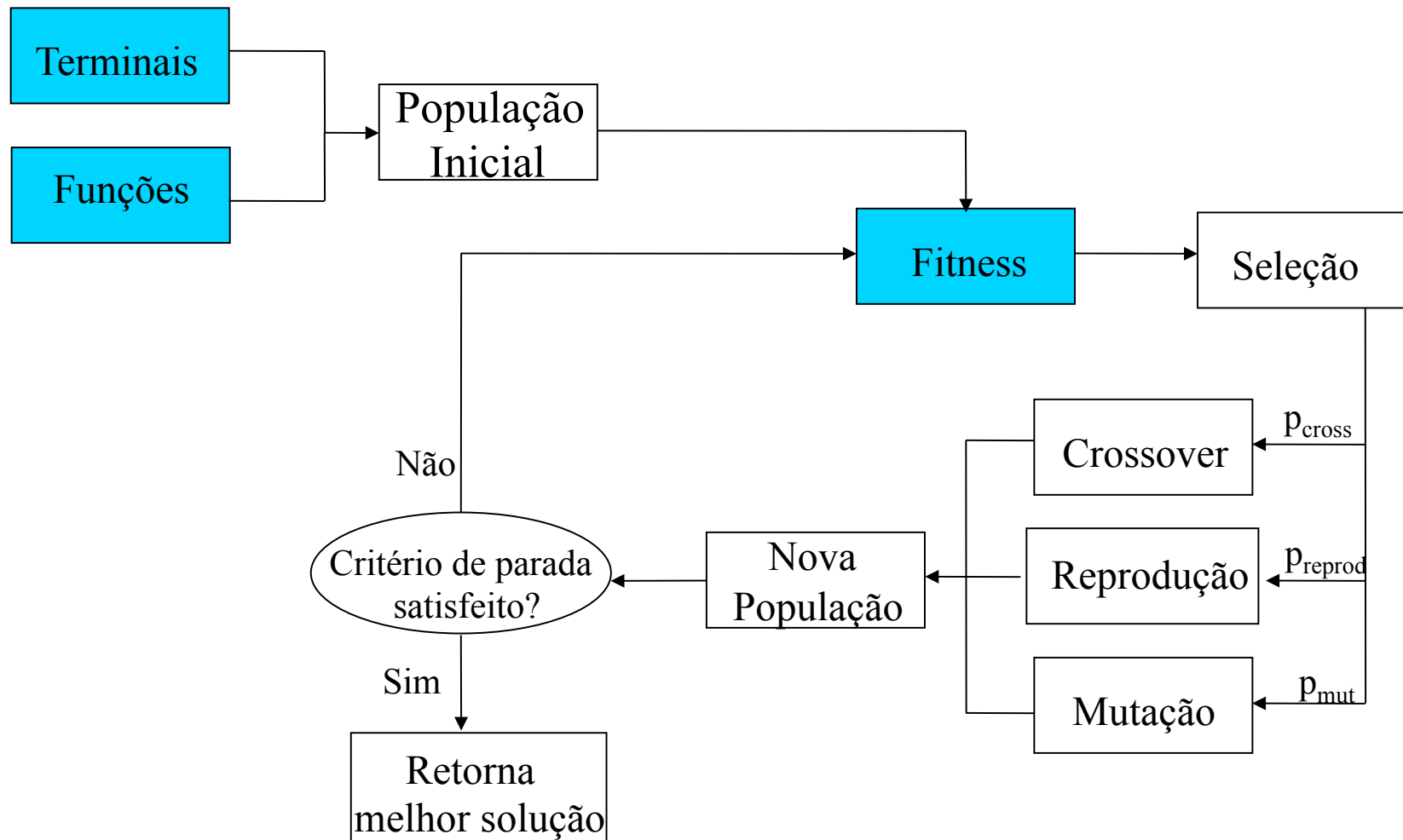
Computação Natural

Gisele L. Pappa

# Introdução

- Na aula passada, estudamos três propriedades que devem ser respeitadas ao criar um conjunto de funções de um PG, incluindo *fechamento*
- As dificuldades impostas pelo fechamento levaram a criação de uma novas vertentes dentro da GP:
  - GP restrito a sintaxe
  - GPs baseados em gramáticas

# Programação Genética

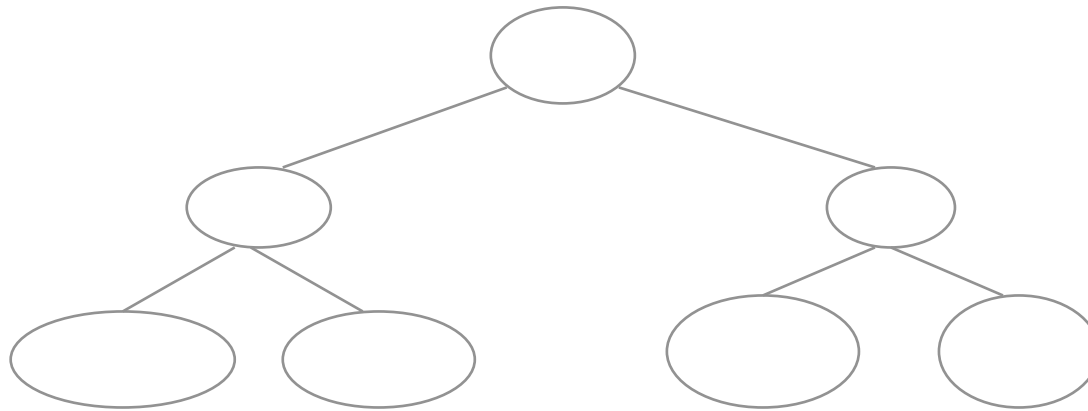


# GP Restrito a Sintaxe

- Para cada função do conjunto de funções, especificar o tipo de dados de seus argumentos e o tipo de dados retornado
- Cada terminal é também associado a um tipo de dados
- Cruzamento e mutação são modificados com respeito a restrições nos tipos de dados

# GP Restrito a Sintaxe

- 



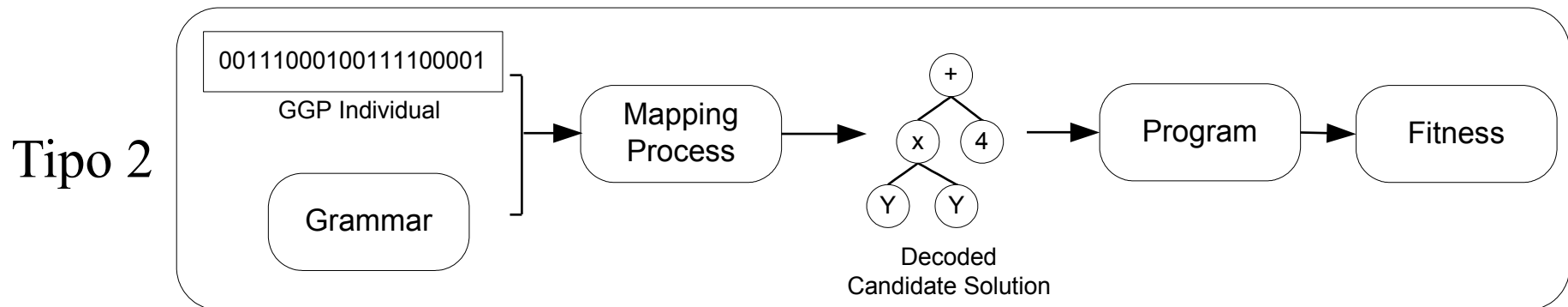
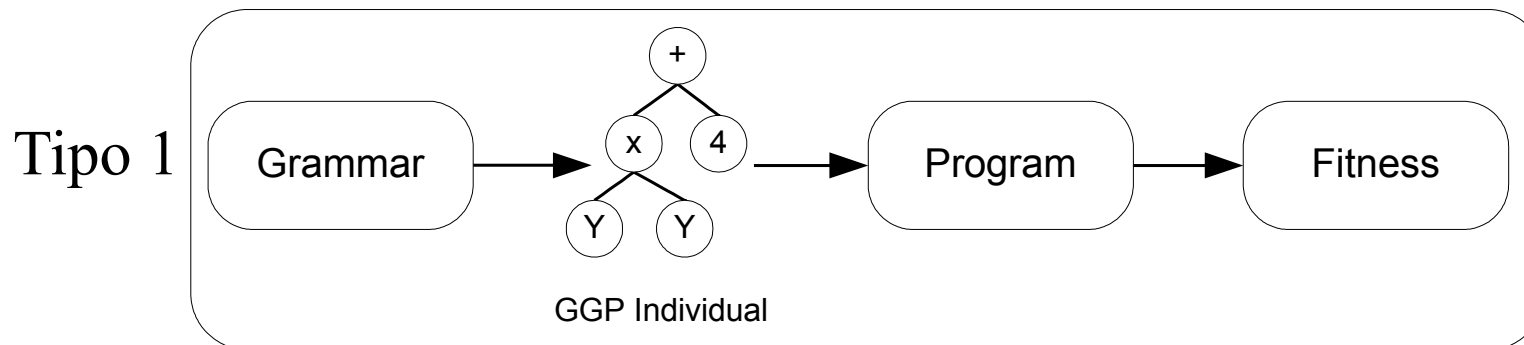
Função	Tipo de dados dos argumentos	Tipo de dado retornado
$+$ , $-$ , $*$ , $/$	(real, real)	real
$>$ , $<$	(real, real)	boolean
AND, OR	(boolean, boolean)	boolean

# GP baseada em gramática

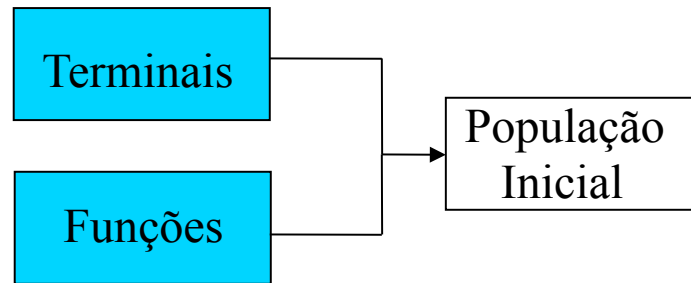
- Além de garantir a propriedade de fechamento, permite **incorporar ao espaço de busca domínio sobre o problema**
- GP baseada em gramática podem ser divididas em 2 grandes classes de acordo com:
  - Tipo de representação utilizado
  - Tipo da gramática utilizada
    - Livre de contexto, lógica, etc

# PG baseada em gramática

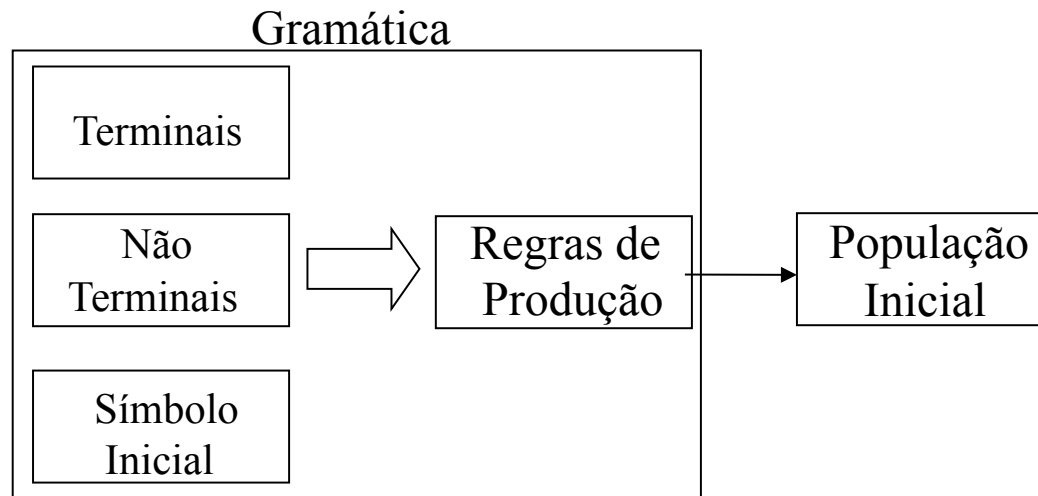
- Classificação de acordo com a representação



# PG baseada em Gramáticas (Tipo 1)



PG Tradicional



PG baseada em Gramáticas

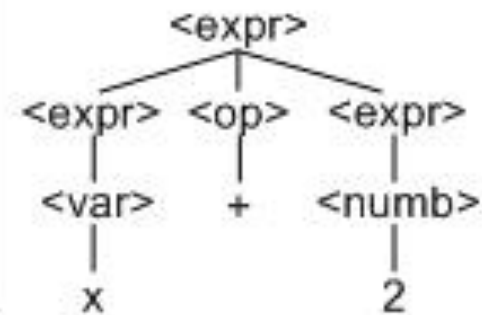


# Exemplo de Gramática

## CFG Grammar

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \mid$  (1)  
 $\langle \text{numb} \rangle \mid$  (2)  
 $\langle \text{var} \rangle$  (3)  
 $\langle \text{op} \rangle ::= + \mid$  (4)  
 $-$  (5)  
 $\langle \text{var} \rangle ::= x \mid$  (6)  
 $y$  (7)  
 $\langle \text{numb} \rangle ::= 2 \mid$  (8)  
 $4$  (9)

## Derivation Tree for expression x+2



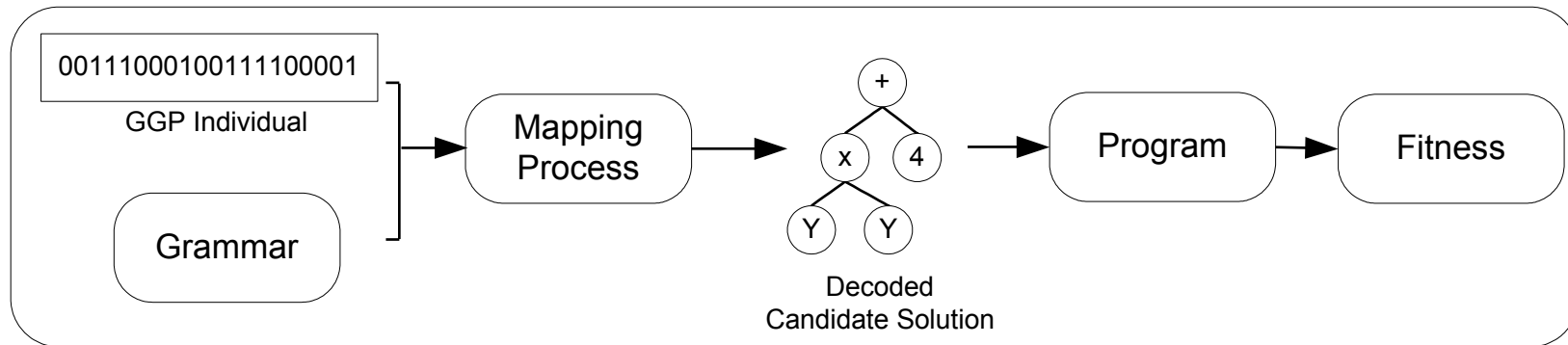
## Derivation Steps followed to produce x+2

$\langle \text{expr} \rangle \xrightarrow{1} \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \xrightarrow{3} \langle \text{var} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \xrightarrow{6} x \langle \text{op} \rangle \langle \text{expr} \rangle \xrightarrow{4}$   
 $x + \langle \text{expr} \rangle \xrightarrow{2} x + \langle \text{numb} \rangle \xrightarrow{8} x + 2$

# Diferenças em relação a PG convencional

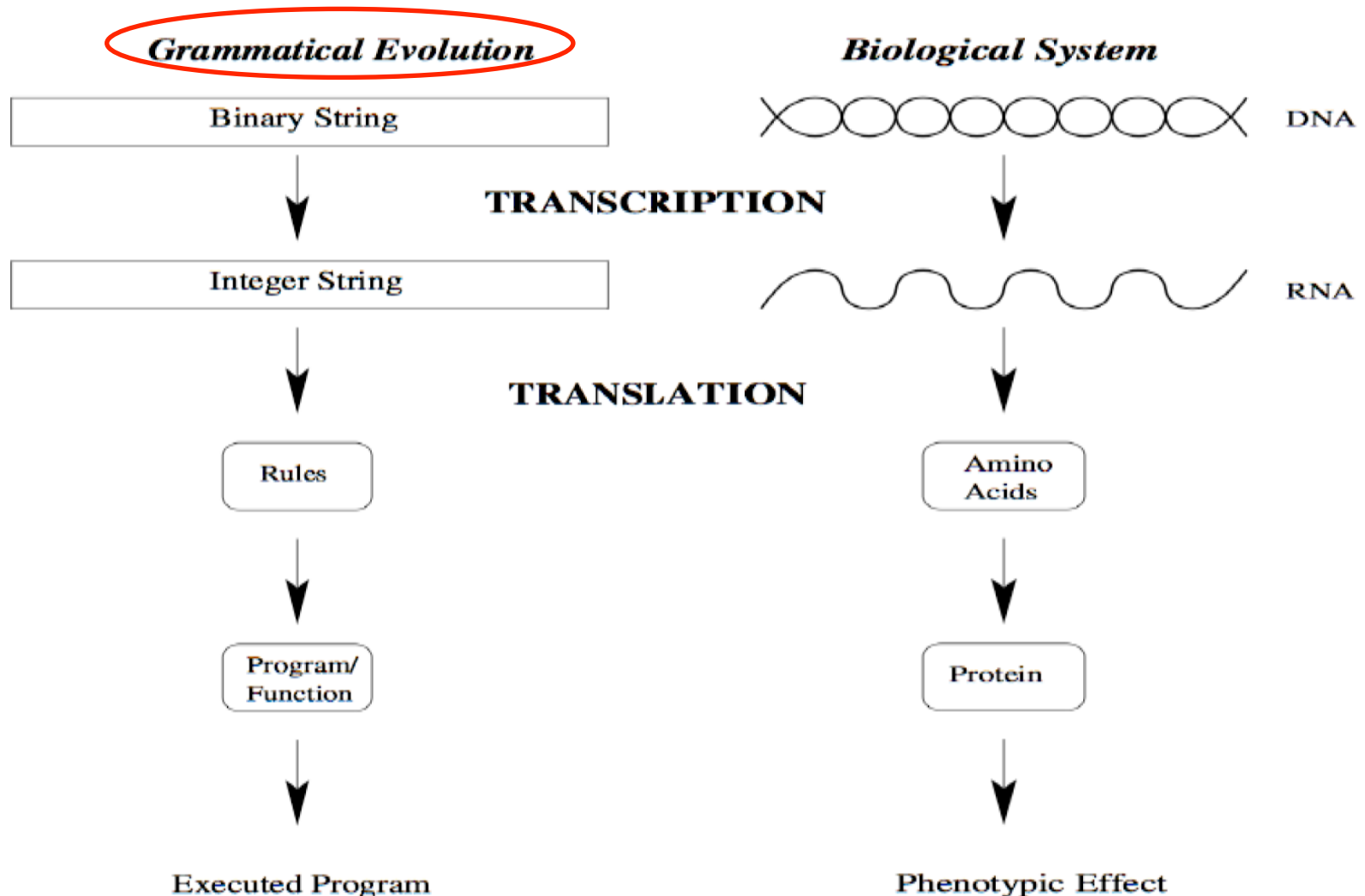
- Indivíduos criados através de mutação e crossover devem respeitar as regras de produção da gramática
- Crossover não tem um poder tão destrutivo

# PG baseada em Gramáticas (Tipo 2)



- Indivíduos são normalmente representados por um string binário
- Existe um mapeamento do genótipo para o fenótipo baseado em processos biológicos

# Mapeamento inspirado na biologia




# Código Genético

Codon	U	C	A	G	
U	UUU - Phe	UCU - Ser	UAU - Tyr	UGU - Cys	U
	UUC - Phe	UCC - Ser	UAC - Tyr	UGC - Cys	C
	UUA - Leu	UCA - Ser	UAA - Stop	UGA - Stop	A
	UUG - Leu	UCG - Ser	UAG - Stop	UGG - Trp	G
C	CUU - Leu	CCU - Pro	CAU - His	CGU - Arg	U
	CUC - Leu	CCC - Pro	CAC - His	CGC - Arg	C
	CUA - Leu	CCA - Pro	CAA - Gln	CGA - Arg	A
	CUG - Leu	CCG - Pro	CAG - Gln	CGG - Arg	G
A	AUU - Ile	ACU - Thr	AAU - Asn	AGU - Ser	U
	AUC - Ile	ACC - Thr	AAC - Asn	AGC - Ser	C
	AUA - Ile	ACA - Thr	AAA - Lys	AGA - Arg	A
	AUG - Met	ACG - Thr	AAG - Lys	AGG - Arg	G
G	GUU - Val	GCU - Ala	GAU - Asp	GGU - Gly	U
	GUC - Val	GCC - Ala	GAC - Asp	GGC - Gly	C
	GUA - Val	GCA - Ala	GAA - Glu	GGA - Gly	A
	GUG - Val	GCG - Ala	GAG - Glu	GGG - Gly	G

- Degeneração de código genético (diferentes codons mapeam o mesmo aminoácido)

Code	Name	Code	Name
Phe	Phenylalanine	Leu	Leucine
Tyr	Tyrosine	Cys	Cysteine
Trp	Tryptophan	Pro	Proline
His	Histidine	Gln	Glutamine
Arg	Arginine	Ile	Isoleucine
Met	Methionine	Thr	Threonine
Asn	Asparagine	Lys	Lysine
Ser	Serine	Val	Valine
Ala	Alanine	Asp	Aspartic Acid
Glu	Glutamic Acid	Gly	Glycine

# Código Genético

GENETIC CODE	PARTIAL PHENOTYPE	GE Codon GE (8 bits)	Regra
<b>Codon</b> (A group of 3 Nucleotides)	<b>Amino Acid</b> (Protein Component)		
G G C G G A G G G	 Glycine	00000010 00010010 00100010	<line>

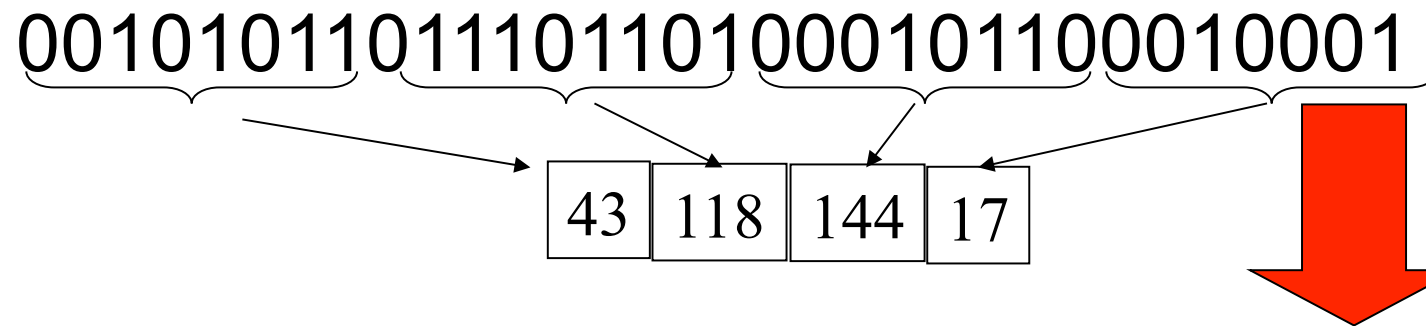
- Para uma regra com duas escolhas

$\langle \text{code} \rangle ::= \langle \text{line} \rangle \mid (0)$

$\langle \text{code} \rangle \langle \text{line} \rangle (1)$

O valor do codon de um GE mod número de regras  
determina o número da regra

# Exemplo de Mapeamento



Indivíduo

## Gramática

A  $\langle \text{seq} \rangle ::= \langle \text{vowel} \rangle \quad (0)$   
|  $\langle \text{seq} \rangle \langle \text{vowel} \rangle \quad (1)$

B  $\langle \text{vowel} \rangle ::= a \quad (0)$   
| e  $(1)$   
| i  $(2)$   
| o  $(3)$   
| u  $(4)$

## Processo de Decodificação

$\langle \text{seq} \rangle$	$43 \% 2 = 1$
$\langle \text{seq} \rangle \langle \text{vowel} \rangle$	$118 \% 2 = 0$
$\langle \text{vowel} \rangle \langle \text{vowel} \rangle$	$144 \% 5 = 4$
u $\langle \text{vowel} \rangle$	$17 \% 5 = 2$
ui	

# Mapeamento

- O que acontece se eu termino de ler o genótipo e meu indivíduo ainda apresenta não-terminais?
  - Uso o conceito de “**wrapping**”
    - Reaproveitamento de material genético (inspirado na sobreposição de genes comum bactérias)



# Implicações da Degeneração de Código Genético

- Aparecimento de mutações neutras
- Variações no genótipo não tem efeito no fenótipo

	U	C	A	G	
U	UUU - Phe	UCU - Ser	UAU - Tyr	UGU - Cys	U
	UUC - Phe	UCC - Ser	UAC - Tyr	UGC - Cys	C
	UUA - Leu	UCA - Ser	UAA - Stop	UGA - Stop	A
	UUG - Leu	UCG - Ser	UAG - Stop	UGG - Trp	G
C	CUU - Leu	CCU - Pro	CAU - His	CGU - Arg	U
	CUC - Leu	CCC - Pro	CAC - His	CGC - Arg	C
	CUA - Leu	CCA - Pro	CAA - Gln	CGA - Arg	A
	CUG - Leu	CCG - Pro	CAG - Gln	CGG - Arg	G
A	AUU - Ile	ACU - Thr	AAU - Asn	AGU - Ser	U
	AUC - Ile	ACC - Thr	AAC - Asn	AGC - Ser	C
	AUA - Ile	ACA - Thr	AAA - Lys	AGA - Arg	A
	AUG - Met	ACG - Thr	AAG - Lys	AGG - Arg	G
G	GUU - Val	GCU - Ala	GAU - Asp	GGU - Gly	U
	GUC - Val	GCC - Ala	GAC - Asp	GGC - Gly	C
	GUA - Val	GCA - Ala	GAA - Glu	GGA - Gly	A
	GUG - Val	GCG - Ala	GAG - Glu	GGG - Gly	G

# Representação

- Trabalha com vetores de bits de tamanho variável
- Ao gerar a população inicial, determina um número máximo de codons que cada indivíduo pode ter

# Operadores Genéticos

- Mutação de um ponto
- Crossover de um ponto
- Duplicação de codons
  - Seleccionados aleatoriamente e inseridos antes do último codon

# Principais características de Evolução de Gramáticas

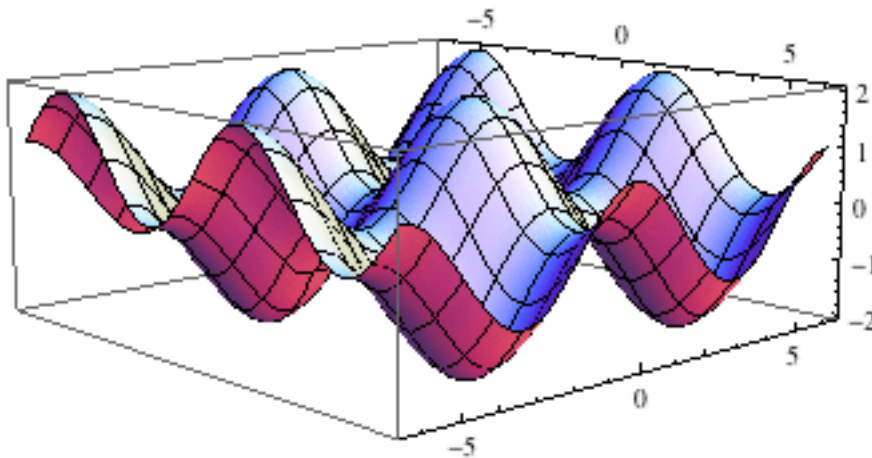
- Separa o genótipo do fenótipo
- Degeneração de código genético
  - Ajuda a manter a diversidade da população
  - Ajuda a preservar a funcionalidade dos programas através de mutações neutras
- Operador *wrapping*
  - Reusar código genético

# Críticas (Problemas)

- Novamente não existe semântica
- Cruzamento não faz muito sentido
  - Estaremos trocando bits que não fazem referência alguma a gramática
  - Operador *wrap* também faz com que o efeito do cruzamento seja amplificado
- Não existe localidade nos operadores, característica importante em EAs

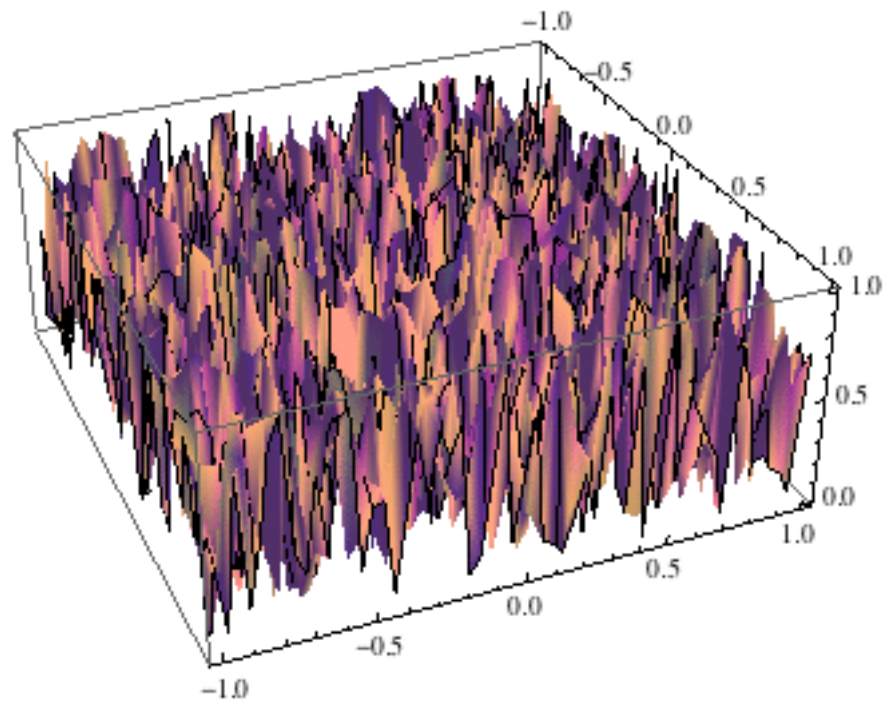
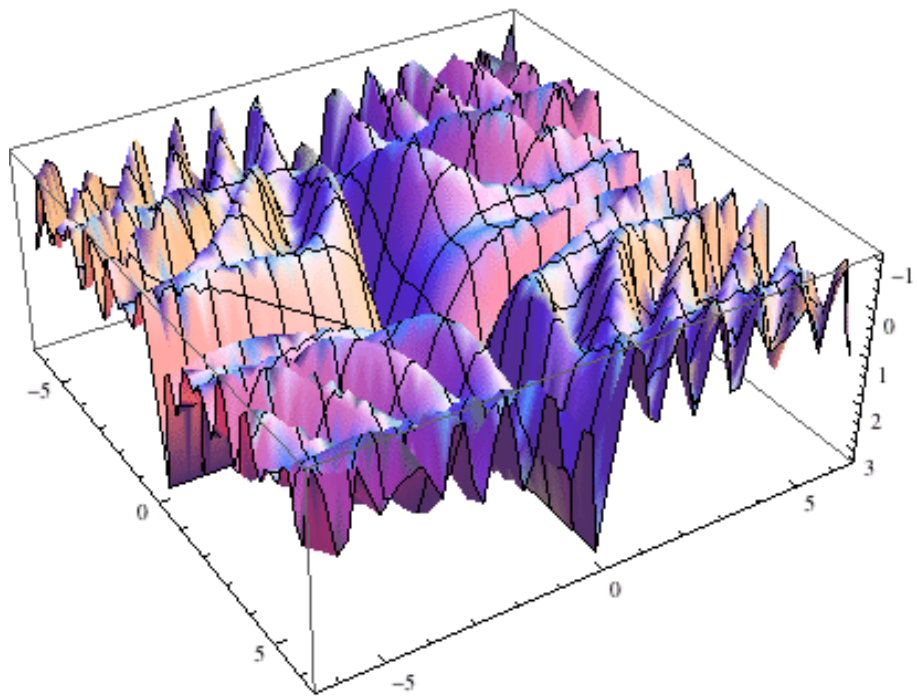
# *Fitness Landscape*

- Gráfico que ilustra as  $n$  dimensões do seu problema, e a *qualidade da sua solução naquele ponto do espaço*.



$$F(x,y) = \sin(x) + \cos(y)$$

# *Fitness Landscape*

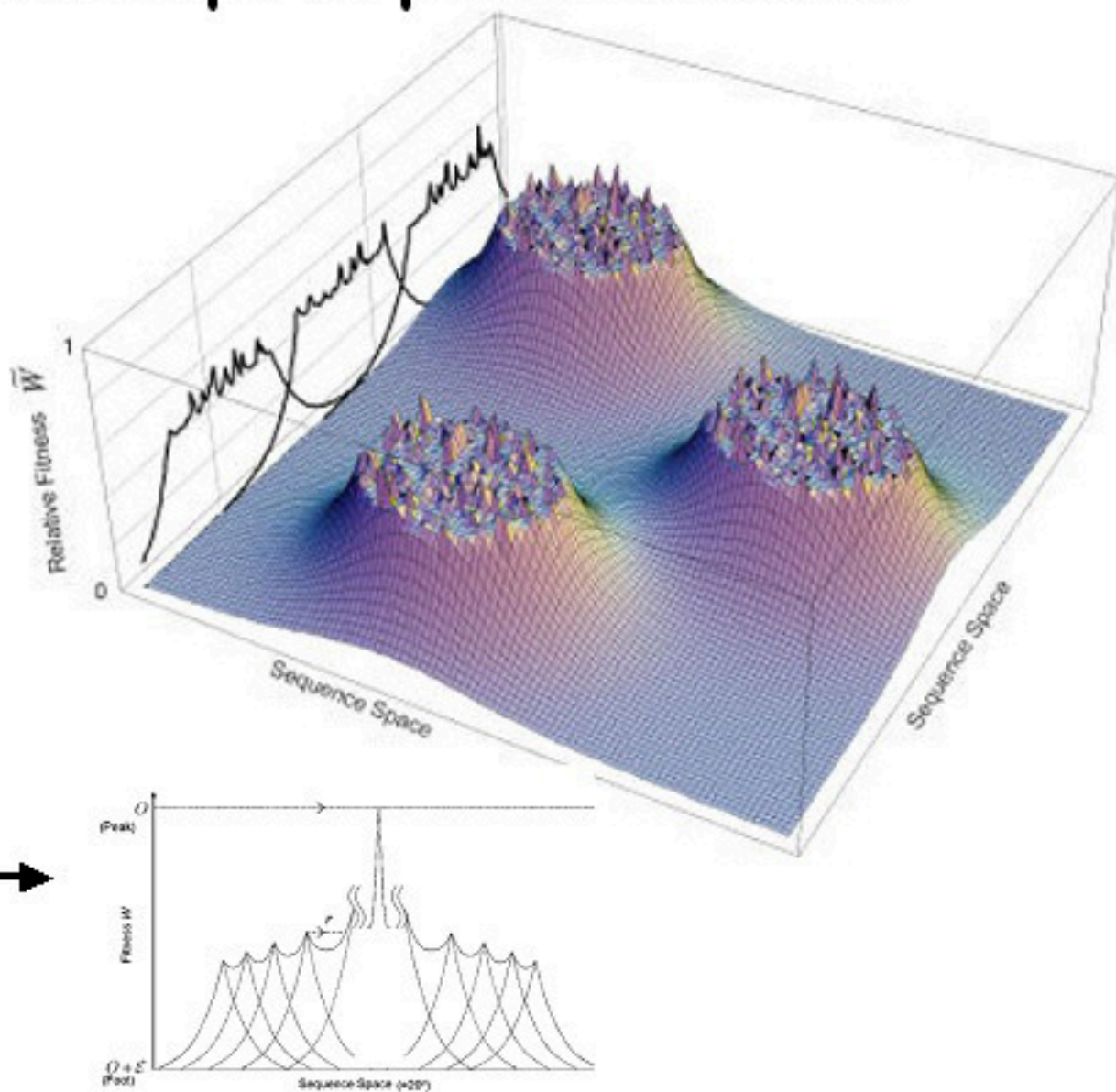


# A real fitness landscape for protein evolution

Random substitutions easily climb to 40% of the original function.

Enormous diversity is required to climb within the rugged surface to 100%.

2D plot of a hill →



<http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0000096>



# Localidade (*locality*)

- Diz respeito ao quanto que genótipos vizinhos correspondem a fenótipos vizinhos
- É um bom indicador da dificuldade de se resolver um problema
- Localidade
  - Alta – todos os genótipos vizinhos correspondem a fenótipos vizinhos
  - Baixa – maioria dos genótipos vizinhos não corresponde aos fenótipos vizinhos

# Localidade (*locality*)

- Representações com alta localidade são necessárias para se ter uma busca eficiente
  - Operadores tem o mesmo efeito no espaço de genótipo e fenótipo
- Operadores genéticos podem ser aproveitar do conhecimento que se tem do espaço de fitness para guiar a busca

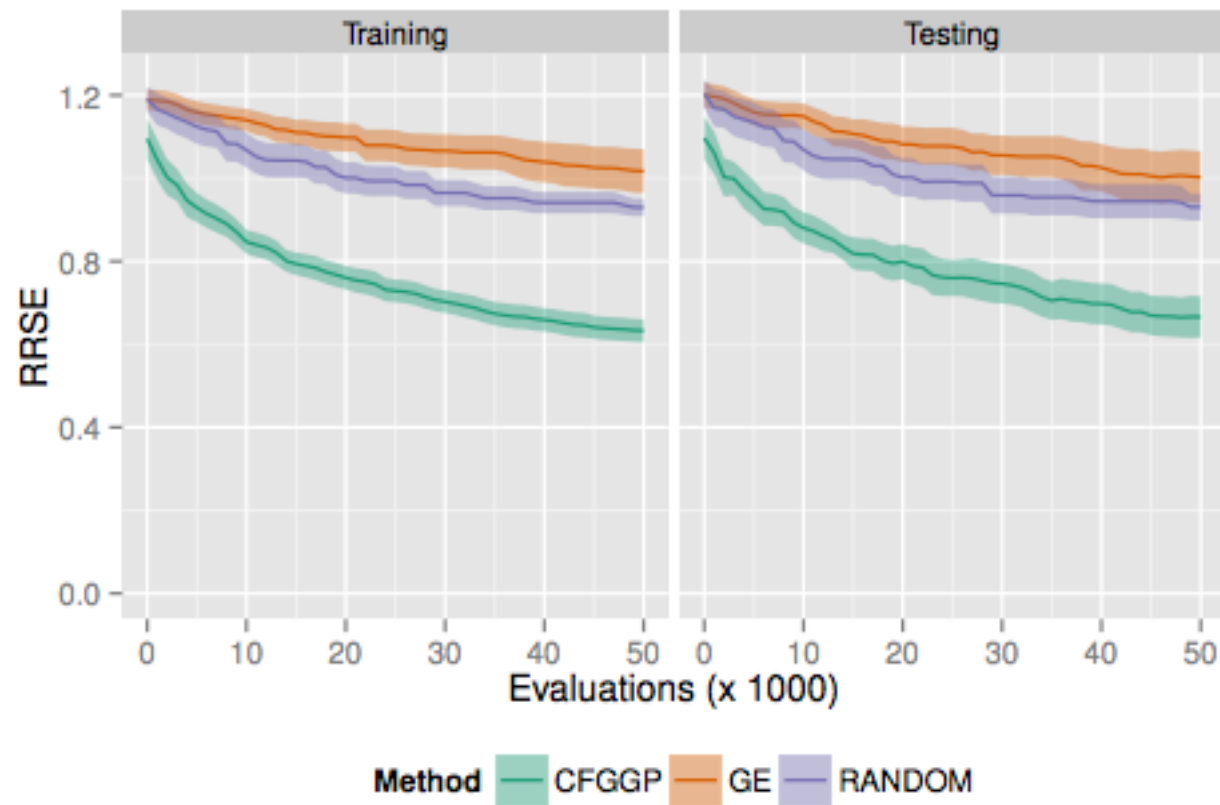
# **RESULTADOS: TIPO 1 VERSUS TIPO 2**

# Gramática – Regressão Simbólica

```
<prog> ::= <expr>
<expr> ::= <expr> <op> <expr>
          | ( <expr> <op> <expr> )
          | <pre-op> ( <expr> )
          | <protected-op> |<var>
<op> ::= + | * | -
<protected-op> ::= div( <expr>, <expr>)
<pre-op> ::= sin | cos | exp | inv | log
<var> ::= X | 1.0
```

(b) Symbolic Regression

# Comparações



**Figure 6: Evolution of training and testing fitness on the Boston Housing symbolic regression problem.**

## Leitura Recomendada

- O'Neill M., Ryan C. *Automatic Generation of Programs with Grammatical Evolution*. In Proceedings of AICS 1999, pages 72-78.
- P. A. Whigham, Grammatically-based Genetic Programming, Proc. of the Workshop on Genetic Programming: From Theory to Real-World Applications, 1995, pages 33-41.

### Mais informações

- <http://www.grammatical-evolution.org/>

# Artigos de referência da aula

- Hayashi, Yuuki, et al. "Experimental rugged fitness landscape in protein sequence space." *PLoS One* 1.1 (2006): e96.
- P. A. Whigham, G. Dick, J. Maclaurin, and C. A. Owen. 2015. Examining the "Best of Both Worlds" of Grammatical Evolution. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*