General Schema Theory for Genetic Programming with Subtree-Swapping Crossover: Part I

Riccardo Poli rpoli@essex.ac.uk
Department Computer Science, University of Essex, Colchester, CO4 3SQ, UK

Nicholas Freitag McPhee mcphee@mrs.umn.edu
Division of Science and Mathematics, University of Minnesota, Morris, Morris, MN, USA

Abstract

This is the first part of a two-part paper which introduces a general schema theory for genetic programming (GP) with subtree-swapping crossover. The theory is based on a Cartesian node reference system which makes it possible to describe programs as functions over the space $\mathbb{N}^2$ and allows one to model the process of selection of the crossover points of subtree-swapping crossovers as a probability distribution over $\mathbb{N}^4$. In Part I, we present these notions and models and show how they can be used to calculate useful quantities. In Part II we will show how this machinery, when integrated with other definitions, such as that of variable-arity hyperschema, can be used to construct a general and exact schema theory for the most commonly used types of GP.

Keywords

Genetic Programming, Node Reference Systems, Models of Crossover, Schema Theory

1 Introduction

One of the most difficult tasks one has to face when developing theoretical models of a complex system, such as genetic programming (GP), is choosing a suitable language in which to express the theory. This involves identifying the crucial quantities on which to base the theory and expressing them in a good representation, the outcome of this process being a set of useful definitions. When these choices are correct, finding relationships between the quantities which describe the system is a much easier task and the resulting theory is simpler and more compact, easier to communicate, etc. How can one choose these theoretical building blocks well when attacking a new unchartered domain? There is no simple and ready recipe here: intuition, luck, reuse of concepts developed for other domains and a substantial amount of trial and error often are key ingredients.

This paper is the first part of a two-part paper which introduces a general schema theory for genetic programming (GP) with subtree-swapping crossover. In Part I we present the results of our efforts to develop a suitable language in which we can express the theory. In particular we will focus on the following ingredients: (1) a Cartesian node reference system which makes it possible to describe the position of any node in any tree in a population univocally, (2) the notion that programs (and various features of programs) are functions over the coordinates in this reference system, and (3) the notion that the key elements of subtree-swapping crossover and mutation operators can be modelled as probability distributions over $\mathbb{N}^4$ or $\mathbb{N}^2$, respectively. Other key notions, such as the definitions of schema and hyperschema, and the schema theorem

©2003 by the Massachusetts Institute of Technology

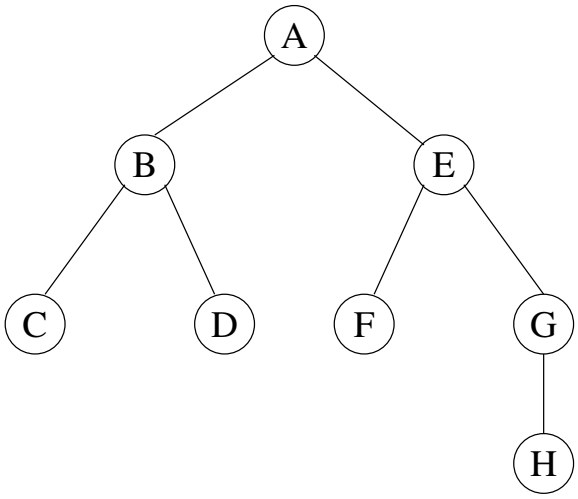

Figure 1: A sample syntax tree.

for subtree swapping crossovers will be introduced in Part II.[1]

The paper is organised as follows. Firstly, in Section 2, we introduce the notion of node reference systems. We use it to define the concepts of functions and probability distributions over such reference systems in Section 3. Then, we show how these can be used to build probabilistic models of different crossover operators (Section 4). Some conclusions are drawn in Section 5 where we also briefly summarise how the theory developed here will be used in Part II.

2 Node Reference Systems

Given a syntax tree like, for example, the one in Figure 1, representing the S-expression `(A (B C D) (E F (G H)))`, there are a variety of methods to indicate unambiguously the position of one particular node in the tree.

One method is to use the path from the root node (D'haeseleer, 1994), which corresponds to using a variable-dimensionality relative coordinate system. For example, as illustrated in Figure 2, it would be possible to indicate node `H` in the syntax tree in Figure 1 using the list of coordinates [2 2 1] which means "select the second argument of node `A`, then the second of node `E`, then the first of node `G`". This method, by definition, allows one to determine exactly the path followed to reach a particular node in the reference system. However, it has the disadvantage of not corresponding to our typical notion of a Cartesian reference system because the number of coordinates necessary to locate a node grows with the depth of the node in the tree.

A better alternative from this point of view is to use a coordinate system in which the nodes in the tree are organised into layers of increasing depth (as is frequently done when trees are drawn). The nodes are then aligned to the left (as is illustrated in Figure 3) and an index is assigned to each node in a layer. The layer number d and the index i can then be used to define a Cartesian coordinate system. So, for example, the position of node `G` in Figure 1 could be represented with the coordinates (2,3), meaning that it belongs to layer 2 (node `A` being in layer 0) and that it is the fourth node in the layer (assuming that the index starts from 0). This reference system, however, presents the problem that it is not possible to infer the structure of a

[1] Early versions of some of this work were presented in (Poli, 2001) and, to a lesser degree, (Poli and McPhee, 2001). This two-part paper is much more detailed, however, and includes a number of new results and examples.

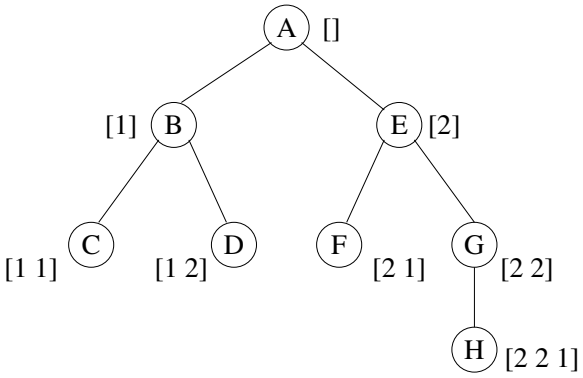

Figure 2: Variable-dimensionality relative node reference system.

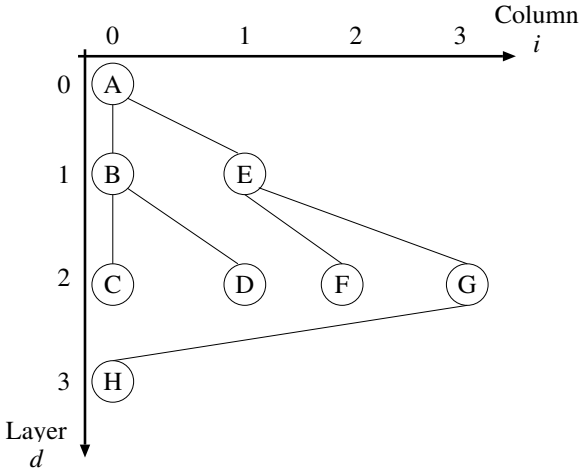

Figure 3: A tree-dependent Cartesian node reference system.

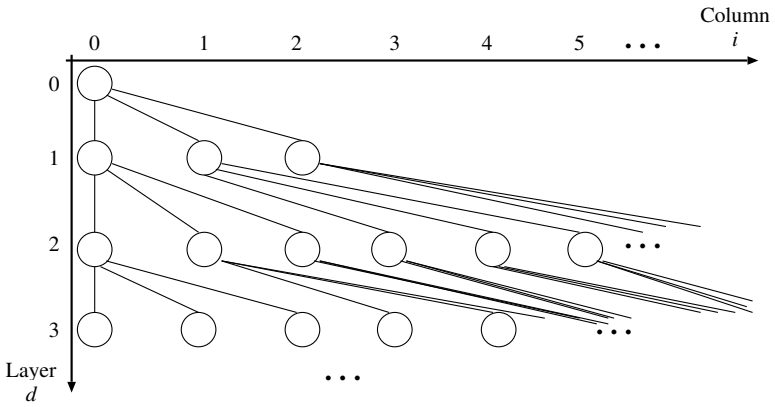

Figure 4: Maximal tree obtainable with nodes of arity 3 within a Cartesian reference system. Only four layers are shown.

tree from the coordinates of its nodes. For example, if one knows that node H is at coordinates (3,0), it is impossible to determine the coordinates of its parent node (in this case (2,3)).

A similar coordinate system but without this problem can be defined by assuming that the trees in question have a predefined maximum arity $a_{\max}$. Then one can define a node-reference system like the previous one for the largest possible tree that can be created with nodes of arity $a_{\max}$. This maximal tree would include 1 node of arity $a_{\max}$ at depth 0, $a_{\max}$ nodes of arity $a_{\max}$ at depth 1, $a_{\max}^2$ nodes of arity $a_{\max}$ at depth 2, etc. (see the example in Figure 4). Finally one can use this maximal system to also locate the nodes of non-maximal trees using a subset of the nodes and links in the maximal tree. This is illustrated in Figure 5, where we assume $a_{\max} = 3$. So, for example, node G in Figure 1 would have coordinates (2,4) while node H would have coordinates (3,12). In this reference system it is always possible to find a node's parent, and consequently find the route to the root node from any given valid pair of coordinates. This reference system has features similar to those of the one shown in Figure 2. However, it corresponds more closely to the standard notion of a Cartesian reference system. If one chooses $a_{\max}$ to be the maximum arity of the functions in the function set, it is possible to use this reference system to represent the structure of any program in any population. Because of these properties, we will use this reference system in the rest of the paper.

Finally, it should be noted that in the Cartesian reference system in Figure 5 it is possible to transform pairs of coordinates into integers by counting the nodes in the reference system in breadth-first order. So, node (d, i) would correspond to the integer $\sum_{x=0}^{d-1}(a_{\max})^x + i$. For example, if $a_{\max} = 3$, (2,1) corresponds to 5. Clearly, it is also possible to map integers into node coordinates unambiguously, although not all the coordinates will refer to existing nodes in non-maximal trees. We will use these properties to simplify the notation in some parts of this paper.

3 Functions and Probability Distributions over Node Reference Systems

Given a node reference system it is possible to define functions over it. An example of such functions is a function which represents a particular computer program. Given a pair of coordinates, this function returns the primitive stored at these coordinates in a given syntax tree. For a given program h one could, for example, define the function $N(d, i, h)$ which returns the node in h stored at position (d, i) if (d, i) is a valid pair of coordinates. If (d, i) is not in h then a

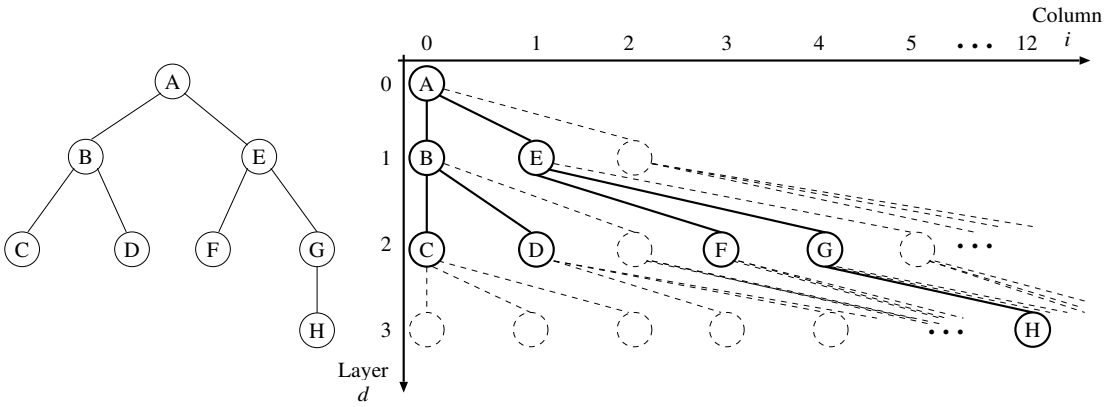

Figure 5: Tree-independent Cartesian node reference system (right) representing the non-maximal tree (left). Nodes and links of the maximal tree are drawn with dashed lines.

conventional default value, $\emptyset$, is returned to represent the absence of a node. For example, the program h=`(IF (AND x1 x2) (OR x1 x3) x1)` represented in Figure 6 would induce the following function: $N(0,0,h) =$ `IF`, $N(1,0,h) =$ `AND`, $N(1,1,h) =$ `OR`, $N(1,2,h) =$ `x1`, $N(2,0,h) =$ `x1`, $N(2,1,h) =$ `x2`, $N(2,2,h) = \emptyset$, $N(2,3,h) =$ `x1`, $N(2,4,h) =$ `x3`, $N(2,5,h) = \emptyset$, $N(2,6,h) = \emptyset$, etc.. Obviously, this function could be represented as the following table:

$N(d,i,h)$

	i					
d	0	1	2	3	4	$\cdots$
0	IF	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	
1	AND	OR	x1	$\emptyset$	$\emptyset$	
2	x1	x2	$\emptyset$	x1	x3	$\cdots$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	
$\vdots$			$\vdots$			

So, programs can be seen as functions over the space $\mathbb{N}^2$. Below we will refer to $N(d,i,h)$ as the *name function* for h.

Another useful function is the *size function* $S(d,i,h)$ which represents the number of nodes present in the subtree rooted at coordinates (d,i) in tree h, with the convention that $S(d,i,h) = 0$ if (d,i) indicates a nonexistent node. For example if $a_{\max} = 3$ the tree in Figure 1 has the following size function:

$S(d,i,h)$

	i								
d	0	1	2	3	4	5	$\cdots$	12	$\cdots$
0	8	0	0	0	0	0		0	
1	3	4	0	0	0	0		0	
2	1	1	0	1	2	0	$\cdots$	0	$\cdots$
3	0	0	0	0	0	0		1	
4	0	0	0	0	0	0		0	
$\vdots$			$\vdots$					$\vdots$	

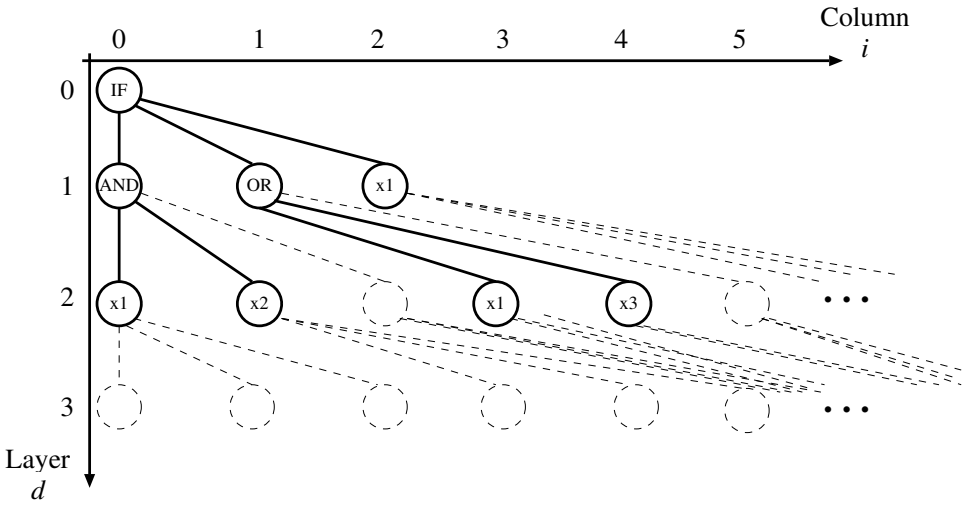

Figure 6: Syntax tree for the program `(IF (AND x1 x2) (OR x1 x3) x1)` represented in a node reference system for nodes with maximum arity 3.

The size function can be defined using the name function:

$$S(d,i,h) = \sum_{x \geq d} \sum_{y=i\cdot(a_{\max})^{x-d}}^{(i+1)\cdot(a_{\max})^{x-d}-1} \delta(N(x,y,h) \neq \emptyset),$$

where $\delta(a)$ is a function which returns 1 if a is true, 0 otherwise.

Other examples of useful node functions include:

- The *arity function* $A(d,i,h)$ which returns the arity of the node at coordinates (d,i) in program h, and returns $\emptyset$ or -1 by convention if (d,i) indicates a nonexistent node. For example, for the tree in Figure 5, $A(0,0,h) = 2$, $A(1,0,h) = 2$, $A(2,1,h) = 0$ and $A(2,4,h) = 1$. Clearly this function could be defined as the composition of the node function $N(d,i,h)$ mentioned above with a function $a(x)$ which returns the arity of primitive x: $A(d,i,h) = a(N(d,i,h))$, where $a(\emptyset)$ could be defined to be $\emptyset$ or -1 by convention.

- The *type function* $T(d,i,h)$ which returns the *data type* of the node at coordinates (d,i) in h. This could be defined as $T(d,i,h) = t(N(d,i,h))$ where the function $t(x)$ returns the type of primitive x (with $t(\emptyset) = \emptyset$).

- The *function-node function* $F(d,i,h)$ which returns 1 if the node at coordinates (d,i) is in h and it is a function, 0 otherwise. The number of internal nodes in h is therefore given by $\sum_{d,i} F(d,i,h)$.

Similarly, it is possible to define functions on multiple trees, each with their own node-coordinate system. An interesting function of this kind is the *crossover-offspring size function* $S_o(d_1,i_1,d_2,i_2,h_1,h_2)$ which returns the size of the offspring produced by replacing the subtree rooted at coordinates (d_1,i_1) in parent h_1 with the subtree rooted at coordinates (d_2,i_2) in parent h_2. Using the size function defined above we can write:

$$S_o(d_1,i_1,d_2,i_2,h_1,h_2) = S(0,0,h_1) - S(d_1,i_1,h_1) + S(d_2,i_2,h_2).$$

Other useful functions of this kind will be introduced and used in the next section to model different types of subtree crossover. Before we do that we need to introduce the notion of probability distribution over a node reference system.

Most genetic operators used in GP require the selection of a node where a transformation is performed which leads to the creation of an offspring (e.g. the insertion of a random subtree, or of a subtree taken from another parent). In most cases the selection of the node is performed using a stochastic process of some sort, and it is possible to model this process by assuming that a probability distribution is defined over the nodes of each individual. If we use the node-reference system introduced in the previous section, this can be expressed as a function:

$$p(d, i|h) = \Pr\{\text{A node at depth } d \text{ and column } i \text{ is selected in program } h\}, \tag{1}$$

where we assume that $p(d, i|h)$ is zero for all the coordinates (d, i) which represent nonexistent nodes in h.[2] For example, if we consider the tree in Figure 5 and we select nodes with uniform probability, then $p(d, i|h) = \frac{1}{8}$ if there is a node at (d, i), $p(d, i|h) = 0$ otherwise. If instead we follow (Koza, 1992) and select functions with a probability 0.9 and any node with a probability 0.1, then $p(0, 0|h) = p(1, 0|h) = p(1, 1|h) = p(2, 4|h) = 0.2375$, $p(2, 0|h) = p(2, 1|h) = p(2, 3|h) = p(3, 12|h) = 0.0125$, and $p(d, i|h) = 0$ for all other coordinate pairs.

There are many possible uses for this type of probability distribution. For example it is possible to compute the probability $p(d|h)$ that a node at depth d is selected in h:

$$p(d|h) = \sum_i p(d, i|h).$$

For instance, for the tree in Figure 5 and for the function $p(d, i|h)$ given above, we would obtain $p(0|h) = 0.2375$, $p(1|h) = 0.4750$, $p(2|h) = 0.2750$, $p(3|h) = 0.0125$, and $p(d|h) = 0$ for $d > 3$.

Once $p(d|h)$ is defined it is possible to compute the expected depth of the nodes being selected:

$$E[D|h] = \sum_d p(d|h) \cdot d,$$

where D is a stochastic variable representing the depth of the node selected. For example, if we consider the function $p(d|h)$ given above we obtain:

$$E[D|h] = 0.2375 \cdot 0 + 0.4750 \cdot 1 + 0.2750 \cdot 2 + 0.0125 \cdot 3 + 0.0000 \cdot 4 + \cdots = 1.0625$$

The probability distribution $p(d, i|h)$ can also be used to compute the expected value of numeric node functions, like for example the expected value of the size function for a particular tree h

$$E[S(D, I, h)|h] = \sum_d \sum_i S(d, i, h) p(d, i|h)$$

D and I being random variables representing the depth and column of the node being selected, or other descriptors like the variance of the size function, i.e. $E\left[\left(S(D, I, h) - E[S(D, I, h)|h]\right)^2 \middle| h\right]$.

For the purpose of developing a general schema theory for GP, an important use of probability distributions over node reference systems is for modelling crossover operators, as discussed in the following section.

[2] For this probability distribution we use the notation $p(d, i|h)$ rather than $p(d, i, h)$ to emphasise the fact that $p(d, i|h)$ can be seen as the conditional probability of selecting node (d, i) if (or given that) the program being considered is h. In the rest of the paper, we will do the same for other probabilities distributions.

4 Modelling Subtree-swapping Crossovers

It is quite easy to extend the previous ideas and model subtree-swapping crossover operators as conditional probability distribution functions over the space $\mathbb{N}^4$.

We can do that by considering the probability distribution[3]

$$p(d_1, i_1, d_2, i_2 | h_1, h_2) = \Pr \left\{ \begin{array}{l} \text{A node at depth } d_1 \text{ and column } i_1 \text{ is selected in parent } h_1 \\ \text{and a node at depth } d_2 \text{ and column } i_2 \text{ is selected in parent } h_2 \end{array} \right\},$$

with the convention $p(d_1, i_1, d_2, i_2 | h_1, h_2) = 0$ if (d_1, i_1) is a nonexistent node in h_1 or (d_2, i_2) is an nonexistent node in h_2. It is then possible to compute the marginal probabilities:

$$p_1(d_1, i_1 | h_1, h_2) = \sum_{d_2} \sum_{i_2} p(d_1, i_1, d_2, i_2 | h_1, h_2)$$

and

$$p_2(d_2, i_2 | h_1, h_2) = \sum_{d_1} \sum_{i_1} p(d_1, i_1, d_2, i_2 | h_1, h_2).$$

The first one of these gives the probability that the crossover point in the first parent will be (d_1, i_1) when the parents are h_1 and h_2. The second one does the same for the crossover points in the second parent.[4]

These probability distributions can be used to compute important statistical descriptors like the expected value (or the variance) of the crossover-offspring size function for parents h_1 and h_2:

$$\begin{aligned} & E[S_o(D_1, I_1, D_2, I_2, h_1, h_2) | h_1, h_2] \\ &= \sum_{d_1} \sum_{i_1} \sum_{d_2} \sum_{i_2} S_o(d_1, i_1, d_2, i_2, h_1, h_2) p(d_1, i_1, d_2, i_2 | h_1, h_2) \\ &= S(0, 0, h_1) - \sum_{d_1} \sum_{i_1} \sum_{d_2} \sum_{i_2} S(d_1, i_1, h_1) p(d_1, i_1, d_2, i_2 | h_1, h_2) \\ &\quad + \sum_{d_1} \sum_{i_1} \sum_{d_2} \sum_{i_2} S(d_2, i_2, h_2) p(d_1, i_1, d_2, i_2 | h_1, h_2) \\ &= S(0, 0, h_1) - \sum_{d_1} \sum_{i_1} S(d_1, i_1, h_1) p_1(d_1, i_1 | h_1, h_2) \\ &\quad + \sum_{d_2} \sum_{i_2} S(d_2, i_2, h_2) p_2(d_2, i_2 | h_1, h_2), \end{aligned} \tag{2}$$

where D_i and I_i are random variables representing the depth and column of the node being selected in parent h_i.

Not surprisingly, this result indicates that $E[S_o(D_1, I_1, D_2, I_2, h_1, h_2) | h_1, h_2]$ is given by the size of the first parent minus the mean size of the subtree removed from h_1 plus the mean size of the subtree removed from h_2. However, note that the mean size of the subtrees *removed* from h_1 is not the same quantity as the mean size of the subtrees in h_1: the former is calculated using the probability distribution $p_1(d_1, i_1 | h_1, h_2)$, the latter using the probability distribution

[3] In this paper we assume that, for any given pair of parents, the stochastic algorithm that selects the crossover points does not change its behaviour over time. However, all of the results in the paper can trivially be generalised to the case of time-varying (e.g., adaptive) operators, by simply adding t to the list of conditioning variables, e.g., using the conditional probability distribution $p(d_1, i_1, d_2, i_2 | h_1, h_2, t)$ instead of $p(d_1, i_1, d_2, i_2 | h_1, h_2)$ as a model of crossover.

[4] In general, $p_1(d_1, i_1 | h_1, h_2) \neq p(d_1, i_1 | h_1)$ and $p_2(d_2, i_2 | h_1, h_2) \neq p(d_2, i_2 | h_2)$, as will become clear later in this section.

$p(d_1, i_1|h_1)$. A similar argument applies to the mean size of the subtrees removed from h_2 for insertion.

If the selection of the crossover points is performed independently in the two parents, then

$$p(d_1, i_1, d_2, i_2|h_1, h_2) = p(d_1, i_1|h_1) \cdot p(d_2, i_2|h_2),$$

where $p(d, i|h)$ is defined in Equation 1. We will call crossover operators with this property *separable*.

With functions and probability density functions over node reference systems in hand we can probabilistically model a huge variety of subtree-swapping crossover operators. A few important examples on how to do this are given in the next subsections. Thanks to these probabilistic models of crossover, it is possible to develop a general schema theory for GP as described in Part II.

4.1 Standard Crossover

The most well-known and frequently used type of subtree-swapping crossover operator is the one defined in (Koza, 1992). In this operator the offspring is created by removing a random subtree from one parent and replacing it with a random subtree taken from the other parent. We will call this type of operator *standard crossover*.

Standard crossover is a separable operator. Indeed, assuming uniform selection of the crossover points,

$$p_{\mathrm{StdUnif}}(d_1, i_1, d_2, i_2|h_1, h_2) = p_{\mathrm{StdUnif}}(d_1, i_1|h_1) \cdot p_{\mathrm{StdUnif}}(d_2, i_2|h_2)$$

with

$$p_{\mathrm{StdUnif}}(d, i|h) = \frac{\delta(N(d, i, h) \neq \emptyset)}{S(h)}$$

where $S(h) = S(0, 0, h)$ is the number of nodes in h and $N(d, i, h)$ is the name function defined in Section 3. So,

$$p_{\mathrm{StdUnif}}(d_1, i_1, d_2, i_2|h_1, h_2) = \frac{\delta(N(d_1, i_1, h_1) \neq \emptyset)\delta(N(d_2, i_2, h_2) \neq \emptyset)}{S(h_1)S(h_2)}. \tag{3}$$

For standard crossover with a 90%-function/10%-any-node selection policy (see (Koza, 1992)), it is easy to show that

$$p_{\mathrm{Std90/10}}(d, i|h) = 0.9\frac{F(d, i, h)}{\sum_d \sum_i F(d, i, h)} + 0.1\frac{\delta(N(d, i, h) \neq \emptyset)}{S(h)},$$

where $F(d, i, h)$ is the function-node function defined in Section 3. Thus,

$$\begin{aligned} &p_{\mathrm{Std90/10}}(d_1, i_1, d_2, i_2|h_1, h_2) = \\ &\left(0.9\frac{F(d_1, i_1, h_1)}{\sum_d \sum_i F(d, i, h_1)} + 0.1\frac{\delta(N(d_1, i_1, h_1) \neq \emptyset)}{S(h_1)}\right) \times \\ &\left(0.9\frac{F(d_2, i_2, h_2)}{\sum_d \sum_i F(d, i, h_2)} + 0.1\frac{\delta(N(d_2, i_2, h_2) \neq \emptyset)}{S(h_2)}\right). \end{aligned} \tag{4}$$

Note that for separable operators, the marginal distributions are $p(d_1, i_1|h_1, h_2) = p(d_1, i_1|h_1)$ and $p(d_2, i_2|h_1, h_2) = p(d_2, i_2|h_2)$, whereby Equation 2 becomes

$$E[S_o(d_1, i_1, d_2, i_2, h_1, h_2)|h_1, h_2] = S(0, 0, h_1) - E[S(d_1, i_1, h_1)|h_1] + E[S(d_2, i_2, h_2)|h_2].$$

4.2 One-point Crossover

One-point crossover (Poli and Langdon, 1997b; Poli and Langdon, 1998; Langdon and Poli, 2002) works by selecting a common crossover point in the parent programs and then swapping the corresponding subtrees, like standard crossover. To account for the possible structural diversity of the two parents, one-point crossover analyses the two trees from the root nodes and considers for the selection of the crossover point only the parts of the two trees (common region) which have the same topology (i.e. the same arity in the nodes encountered traversing the trees from the root node).

So, in one-point crossover the selection of the crossover points in the two parents is not performed independently. Indeed, the first and second crossover points must have the same coordinates. To model this operator we define the *common region membership function* $\mathcal{C}(d,i,h_1,h_2)$ which returns 1 when (d,i) is part of the common region of h_1 and h_2. Formally, $\mathcal{C}(d,i,h_1,h_2)=1$ when either $(d,i)=(0,0)$ or

$$\begin{array}{c} A\left(\text{parent}(d,i),h_1\right)=A\left(\text{parent}(d,i),h_2\right)\neq 0,\\ A\left(d,i,h_1\right)\geq 0,\ A\left(d,i,h_2\right)\geq 0,\\ \textbf{and } \mathcal{C}\left(\text{parent}(d,i),h_1,h_2\right)=1, \end{array}$$

where $\text{parent}(d,i)=(d-1,\lfloor i/a_{\max}\rfloor)$, $\lfloor\cdot\rfloor$ is the integer-part function and A is the arity function defined in Section 3. This allows us to formalise the notion of *common region*:

$$C(h_1,h_2)=\{(d,i)\mid \mathcal{C}(d,i,h_1,h_2)=1\}. \tag{5}$$

If we use a uniform probability of node selection, then

$$p_{1\text{pt}}(d_1,i_1,d_2,i_2|h_1,h_2)=\begin{cases} 1/\mathbf{NC}(h_1,h_2) & \text{if } (d_1,i_1)=(d_2,i_2) \text{ and } (d_1,i_1)\in C(h_1,h_2),\\ 0 & \text{otherwise,}\end{cases} \tag{6}$$

where $\mathbf{NC}(h_1,h_2)$ is the number of nodes in the common region $C(h_1,h_2)$. This can also be expressed as:

$$p_{1\text{pt}}(d_1,i_1,d_2,i_2|h_1,h_2) \quad = \quad \frac{\mathcal{C}(d_1,i_1,h_1,h_2)\mathcal{C}(d_2,i_2,h_1,h_2)\delta((d_1,i_1)=(d_2,i_2))}{\mathbf{NC}(h_1,h_2)}.$$

Incidentally, given this definition it is easy to show that

$$p_1(d_1,i_1|h_1,h_2)=\frac{\mathcal{C}(d_1,i_1,h_1,h_2)}{\mathbf{NC}(h_1,h_2)}$$

and

$$p_2(d_2,i_2|h_1,h_2)=\frac{\mathcal{C}(d_2,i_2,h_1,h_2)}{\mathbf{NC}(h_1,h_2)}.$$

Thus for one-point crossover,

$$\begin{array}{rcl} E[S_o(d_1,i_1,d_2,i_2,h_1,h_2)|h_1,h_2] & = & S(0,0,h_1)- \\ & & \frac{\sum_{d_1}\sum_{i_1}S(d_1,i_1,h_1)\mathcal{C}(d_1,i_1,h_1,h_2)}{\mathbf{NC}(h_1,h_2)}+ \\ & & \frac{\sum_{d_2}\sum_{i_2}S(d_2,i_2,h_2)\mathcal{C}(d_2,i_2,h_1,h_2)}{\mathbf{NC}(h_1,h_2)}. \end{array}$$

4.3 Strict One-point Crossover

Similarly, it is possible to model strict-one point crossover (Poli and Langdon, 1997a). In strict one-point crossover, the crossover points in the two parents are constrained not only to be at the same coordinates, but also to belong to a region, the strict common region, which is defined below. Let us start by defining the *strict common region membership function* $\mathcal{SC}(d,i,h_1,h_2)$ which returns 1 when either $(d,i)=(0,0)$ or

$$\begin{gathered} N\left(\text{parent}(d,i),h_1\right)=N\left(\text{parent}(d,i),h_2\right)\neq\emptyset, \\ N\left(d,i,h_1\right)\neq\emptyset,\ N\left(d,i,h_2\right)\neq\emptyset, \\ \textbf{and } \mathcal{SC}\left(\text{parent}(d,i),h_1,h_2\right)=1, \end{gathered}$$

where, again, $\text{parent}(d,i)=(d-1,\lfloor i/a_{\max}\rfloor)$ while N is the name function defined in Section 3. This allows us to define the notion of *strict common region*:

$$SC(h_1,h_2)=\{(d,i)\mid \mathcal{SC}(d,i,h_1,h_2)=1\}. \tag{7}$$

If we use a uniform probability of node selection, then

$$p_{\text{strict 1pt}}(d_1,i_1,d_2,i_2|h_1,h_2)=\begin{cases}1/\mathbf{NSC}(h_1,h_2) & \text{if } (d_1,i_1)=(d_2,i_2) \text{ and} \\ & (d_1,i_1)\in SC(h_1,h_2), \\ 0 & \text{otherwise,}\end{cases} \tag{8}$$

where $\mathbf{NSC}(h_1,h_2)$ is the number of nodes in $SC(h_1,h_2)$.

4.4 Context-preserving Crossover

In context-preserving crossover (D'haeseleer, 1994), the crossover points are constrained to have the same coordinates, like in one-point crossover. However, in this case no other constraint is imposed on their selection (i.e., they are not limited to the common region). In order to model context-preserving crossover, we define the *position match function*

$$\text{PM}(d_1,i_1,d_2,i_2,h_1,h_2)=\begin{cases}1 & \text{if } (d_1,i_1)=(d_2,i_2), N(d_1,i_1,h_1)\neq\emptyset \\ & \text{and } N(d_2,i_2,h_2)\neq\emptyset, \\ 0 & \text{otherwise.}\end{cases}$$

Then, we can write:

$$\begin{aligned} p_{\text{context}}(d_1,i_1,d_2,i_2|h_1,h_2) &= \frac{\text{PM}(d_1,i_1,d_2,i_2,h_1,h_2)}{\sum_{d_1}\sum_{i_1}\sum_{d_2}\sum_{i_2}\text{PM}(d_1,i_1,d_2,i_2,h_1,h_2)} \\ &= \frac{\text{PM}(d_1,i_1,d_2,i_2,h_1,h_2)}{\sum_{d}\sum_{i}\text{PM}(d,i,d,i,h_1,h_2)}. \end{aligned}$$

4.5 Strongly Typed GP Crossover

In strongly typed GP crossover (Montana, 1995), the crossover point in the first parent is selected randomly. The crossover point in the second parent is selected only among the nodes of the same type as the first crossover point. If no nodes of that type exist in the second parent then two alternatives are available: one of the parents can be returned, or nothing is returned. In this second case crossover has to be attempted again. Here we model a version of this second type of crossover in which a new attempt is made using the same two parents. Since the root node is of the same type in all individuals, a pair of valid crossover points always exists.

We start by introducing the *type match function*

$$\text{TM}(d_1,i_1,d_2,i_2,h_1,h_2)=\begin{cases}1 & \text{if } T(d_1,i_1,h_1)=T(d_2,i_2,h_2)\neq\emptyset, \\ 0 & \text{otherwise,}\end{cases}$$

where $T(d,i,h)$ is the type function defined in Section 3. Using this, we can then write:

$$p_{\mathrm{stgp}}(d_1,i_1,d_2,i_2|h_1,h_2) = \frac{\mathrm{TM}(d_1,i_1,d_2,i_2,h_1,h_2)}{\sum_{d_1}\sum_{i_1}\sum_{d_2}\sum_{i_2}\mathrm{TM}(d_1,i_1,d_2,i_2,h_1,h_2)}.$$

4.6 Size-fair Crossover

In size-fair crossover (Langdon, 1999; Langdon, 2000) the first crossover point is selected randomly like in standard crossover (choosing functions with a probability 0.9 and any node with a probability 0.1). Then the size of the subtree to be excised from the first parent is calculated. This is used to constrain the choice of the second crossover point so as to guarantee that the subtree excised from the second parent will not be "unfairly" big.

An approximate model for size-fair crossover (Langdon, 1999; Langdon, 2000) is the following:

$$p_{\mathrm{sizeFair}}(d_1,i_1,d_2,i_2|h_1,h_2) = p_{\mathrm{Std90/10}}(d_1,i_1|h_1)p_{\mathrm{sizeFair}}(d_2,i_2|h_2,S(d_1,i_1,h_1)),$$

where

$$p_{\mathrm{sizeFair}}(d,i|h,s) = \begin{cases} f(h,s) & \text{if } 1 \le S(d,i,h) \le 2s+1, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The function $f(h,s)$ is a relatively complicated function of the following quantities:

$$\begin{aligned} n_+ &= \sum_{(d,i)\in h} \delta(S(d,i,h) > s) \\ n_- &= \sum_{(d,i)\in h} \delta(S(d,i,h) < s) \\ n_0 &= \sum_{(d,i)\in h} \delta(S(d,i,h) = s) \\ mean_+ &= \sum_d \sum_i (S(d,i,h) - s)\delta(S(d,i,h) > s) \\ mean_- &= \sum_d \sum_i (s - S(d,i,h))\delta(S(d,i,h) < s) \end{aligned}$$

(see (Langdon, 1999; Langdon, 2000) for more details).

The expression of $p_{\mathrm{sizeFair}}(d_1,i_1,d_2,i_2|h_1,h_2)$ is only an approximate description of the behaviour of size-fair crossover because it does not model the atypical events in which $n_+ = n_0 = 0$ or $n_- = n_0 = 0$. In these cases the selection of the crossover point in the first parent is repeated, which means that this is chosen according to a probability distribution slightly different from $p_{\mathrm{Std90/10}}(d_1,i_1|h_1)$. This then makes the choice of the crossover points in the two parents dependent, like for one-point crossover. It is possible to derive an exact model of size-fair crossover in terms of conditional probability distributions, but this is beyond the scope of this paper.

5 Conclusions

This paper is Part I of a two-part paper in which a general schema theory for genetic programming with subtree swapping crossover is presented. In this part, we have presented the most important components of a mathematical language that we have developed to express the theory.

The first step has been the definition of a reference system which makes it possible to describe the position of any node in any tree in a population unambiguously. This has then made it possible to represent programs as functions over the coordinates in this reference system, or more generally over $\mathbb{N}^2$. This has allowed us to formalise the definition of a variety of other functions that describe important program features and that are instrumental in simplifying the theoretic results in Part I and II. Among these are some probability distributions over single node reference systems or pairs of reference systems (or more generally over $\mathbb{N}^2$ or $\mathbb{N}^4$), which describe the selection of mutation or crossover points. In this paper, we have provided precise probabilistic models of this type for a variety of crossover operators, including, for example, standard crossover, one-point crossover, and strongly-typed-GP crossover among others. These models can be used to compute a variety of quantities, like for example the mean size of the subtrees present in a tree, or the expected size of offspring produced by two specific parent trees.

In Part II we will show how this machinery, when integrated with other notions, such as the GP schema and the variable-arity hyperschema, can be used to construct a general and exact schema theory for the most commonly used types of GP. The theory will include two main results describing the propagation of GP schemata: a microscopic and a macroscopic schema theorem. The microscopic version is applicable to any crossover operator which replaces a subtree in one parent with a subtree from the other parent to produce the offspring. The macroscopic version is valid for subtree-swapping crossover operators in which the probability of selecting any two crossover points in the parents depends only on their size and shape. Consequently, these theorems can be applied to model most GP systems used in practice.

In Part II, we will also show how this theory can be used to obtain other general results, such as a size-evolution equation for GP with subtree-swapping crossover, and we will provide a variety of examples and the results of numerically integrating schema-evolution equations.

Acknowledgements

The authors would like to thank Jon Rowe, Julian Miller, Xin Yao, and W. B. Langdon for useful discussions and comments on various parts of the work reported in this paper. Nic thanks The University of Birmingham School of Computer Science for graciously hosting him during his sabbatical, and various offices and individuals at the University of Minnesota, Morris, for making that sabbatical possible. Riccardo would like to thank the members of the NEC (Natural and Evolutionary Computation) group at Essex for helpful comments and discussion.

References

D'haeseleer, P. (1994). Context preserving crossover in genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, volume 1, pages 256–261, Orlando, Florida, USA. IEEE Press.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Langdon, W. B. (1999). Size fair and homologous tree genetic programming crossovers. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1092–1097, Orlando, Florida, USA. Morgan Kaufmann.

Langdon, W. B. (2000). Size fair and homologous tree genetic programming crossovers. *Genetic Programming and Evolvable Machines*, 1(1/2):95–119.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer-Verlag.

Montana, D. J. (1995). Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230.

Poli, R. (2001). General schema theory for genetic programming with subtree-swapping crossover. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS 2038, pages 143–159, Milan. Springer-Verlag.

Poli, R. and Langdon, W. B. (1997a). Genetic programming with one-point crossover. In Chawdhry, P. K., Roy, R., and Pant, R. K., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 180–189. Springer-Verlag London.

Poli, R. and Langdon, W. B. (1997b). A new schema theory for genetic programming with one-point crossover and point mutation. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 278–285, Stanford University, CA, USA. Morgan Kaufmann.

Poli, R. and Langdon, W. B. (1998). Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6(3):231–252.

Poli, R. and McPhee, N. F. (2001). Exact schema theorems for GP with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS 2038, pages 126–142, Milan. Springer-Verlag.