

Exact Schema Theory and Markov Chain Models for Genetic Programming and Variable-length Genetic Algorithms with Homologous Crossover

RICCARDO POLI

Department Computer Science, University of Essex, Colchester, CO4 3SQ, UK

rpoli@essex.ac.uk

NICHOLAS FREITAG MCPHEE

Division of Science and Mathematics, University of Minnesota, Morris, Morris, MN, USA

mcphee@mrs.umn.edu

JONATHAN E. ROWE

School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, UK

j.e.rowe@cs.bham.ac.uk

Abstract. Genetic Programming (GP) homologous crossovers are a group of operators, including GP one-point crossover and GP uniform crossover, where the offspring are created preserving the position of the genetic material taken from the parents. In this paper we present an exact schema theory for GP and variable-length Genetic Algorithms (GAs) which is applicable to this class of operators. The theory is based on the concepts of GP crossover masks and GP recombination distributions that are generalisations of the corresponding notions used in GA theory and in population genetics, as well as the notions of hyperschema and node reference systems, which are specifically required when dealing with variable size representations.

In this paper we also present a Markov chain model for GP and variable-length GAs with homologous crossover. We obtain this result by using the core of Vose's model for GAs in conjunction with the GP schema theory just described. The model is then specialised for the case of GP operating on 0/1 trees: a tree-like generalisation of the concept of binary string. For these, symmetries exist that can be exploited to obtain further simplifications.

In the absence of mutation, the Markov chain model presented here generalises Vose's GA model to GP and variable-length GAs. Likewise, our schema theory generalises and refines a variety of previous results in GP and GA theory.

Keywords: Genetic programming, Variable-length genetic algorithms, Schema theory, Markov chain, Vose model, Mixing Matrices.

1. Introduction

Genetic programming (GP) theory has had a difficult childhood. After some excellent early efforts leading to different approximate schema theorems [8, 1, 15, 44, 29, 18], only very recently have schema theories become available which give exact formulations (rather than lower bounds) for the expected number of instances of a schema at the next generation. These exact theories are applicable to GP with one-point crossover [16], standard crossover and other subtree-swapping crossovers [21, 22], and different types of subtree mutation and headless chicken crossover [19, 11].

In Genetic Algorithms (GAs) theory, unlike GP, after a strong initial interest in schemata [7, 27], in the 1990s interest shifted towards exact microscopic Markov chain models [14, 42, 4, 35, 33, 32, 34] possibly with aggregated states [30, 36]. This shift of interest was probably due to the fact that, when first proposed, Vose's model was the only

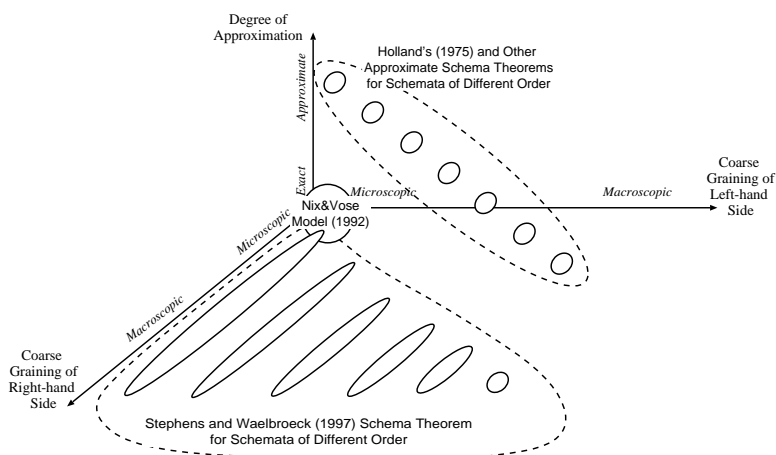


Figure 1. Representation for Vose, Stephens and Holland's GA models.

exact probabilistic model of a GA. However, after the work of Stephens and collaborators in the late 1990s (e.g., see [40, 41, 39]) on exact schema theories based on the notion of dynamic building blocks and the connection highlighted by Vose between his model and a different type of exact schema-based model [42, Chapter 19], it is now clear that Markov-chain and schema-based models, when exact, are just different representations of the same thing.

To understand the relationships between these models, let us focus our attention on the following three properties of GA and GP models: a) whether the models are approximate or exact, b) the level of coarse graining of the predicted quantities (i.e. the left-hand side of a model's equations), and c) the level of coarse graining of the quantities used to make the prediction (i.e. the variables on the right-hand side of a model's equations). We can then represent the space of possible models as a reference system defined by these three quantities. (See [16] for a more extensive discussion on this idea.)

It is easy then to place various models in this space (see Figure 1). Vose-like models, being exact and focusing on microscopic quantities, occupy a spot near the origin. Stephens' schema theory, being exact and focusing on coarse-grained quantities, occupies different portions of the horizontal plane depending on the order of the schema of interest [16]. Holland's schema theorem, being approximate, is represented by various "clouds floating in mid-air" (their exact position depending on the order of the schema of interest).

What's the point in populating the space of GA/GP models? That is, why do we need multiple models? There are at least two reasons. Firstly, different models use different representations for the object under study (in our case a GA or GP system). Each representation puts certain phenomena in the foreground and others in the background. So, in practice each model can answer satisfactorily only certain questions about the system under study and these questions may be very different for different models. Secondly, different

models may have different aims. Because GA and GP systems have so many degrees of freedom and present non-linearity/non-decomposability, exact models are typically huge and totally unmanageable from a computational point of view. However, for specific applications (e.g. designing competent GAs/GP systems) it is possible to dramatically reduce the complexity of such models if one can accept some level of approximation.

A first contribution of this paper¹ is to extend the scope of the schema theory by presenting an exact schema theory for GP which is applicable to a very general class of operators which we call homologous crossovers. This group of operators is important because it generalises most common GA crossovers and includes GP one-point crossover and GP uniform crossover [17]. These operators differ from the standard subtree-swapping crossover [8] in that they require that the offspring being created preserve the position of the genetic material taken from the parents.

While in the last few years the theory of schemata has made considerable progress, both for GAs and GP, this has not corresponded to an equivalent development of Markov chain models for GP and variable-length GAs. As a second contribution, in this paper we start filling this theoretical gap and present a Vose-like Markov-chain model for GP with homologous crossovers. We obtain this result by using the core of Vose’s theory in conjunction with a specialisation of the schema theory for such operators. This formally links GP schema theory and Markov chain models, two worlds believed by many people to be quite separate.

The paper is organised as follows. Firstly, we provide a review of earlier relevant work on GP schemata and cover key definitions and terms in Section 2. Then, in Section 3 we show how these ideas can be used to define the class of homologous crossover operators and build probabilistic models for them. In Section 4 we use these to derive schema theory results for GP with homologous crossover. In Section 5, we summarise Vose’s model for GAs. Then, in Section 6 we present our Markov chain model for GP and variable-length GAs with homologous crossover. In Section 7 we indicate how the theory can be simplified thanks to symmetries which exist when we restrict ourselves to 0/1 trees: a tree-like generalisation of the concept of binary string. We discuss the results presented in this paper in Section 8. Finally, some conclusions are drawn in Section 9. We provide illustrative examples on how to apply the theory in Appendix A.

2. Background and Key Definitions

In this section we provide some background for the research presented in this paper. In so doing we will also introduce some notions which are key to the understanding of the new material presented in later sections. We will, however, postpone the introduction of background information on Vose’s model until later. Let us start with an important notion: the schema.

A *schema* is a set of points of the search space sharing some syntactic feature. For example, in the context of GAs operating on binary strings, the syntactic representation of a schema is usually a string of symbols from the alphabet $\{0,1,*\}$, where the character $*$ is interpreted as a “don’t care” symbol. Typically, schema theorems are descriptions of how the number of members of the population belonging to a schema varies over time. Let $\alpha(H, t)$ denote the probability at time t that a newly created individual samples (or

matches) the schema H , which we term the *total transmission probability* of H . Then an exact schema theorem for a generational system is simply

$$E[m(H, t + 1)] = M\alpha(H, t), \quad (1)$$

where M is the population size, $m(H, t + 1)$ is the number of individuals sampling H at generation $t + 1$ and $E[\cdot]$ is the expectation operator. Holland's [7] and other worst-case-scenario schema theories normally provide a lower bound for $\alpha(H, t)$ or, equivalently, for $E[m(H, t + 1)]$.

One of the difficulties in obtaining theoretical results on GP using the idea of schema is that finding a workable definition of a schema is much less straightforward than for GAs. Several alternative definitions have been proposed in the literature [8, 1, 15, 44, 18, 29]. For brevity here we will describe only the definition introduced in [18], since this is what is used in this paper.

Definition 1. Syntactically a GP schema is a tree composed of functions from the set $\mathcal{F} \cup \{=\}$ and terminals from the set $\mathcal{T} \cup \{=\}$, where \mathcal{F} and \mathcal{T} are the function and terminal sets used in a GP run. The primitive $=$ is a "don't care" symbol that stands for a single terminal or function. Semantically a schema H is the set of all programs having the same shape and the same labels for the non- $=$ nodes as H 's tree-based syntactic representation.

For example, if $\mathcal{F}=\{+, *\}$, $\mathcal{T}=\{x, y\}$ and we represent trees using prefix notation expressions, the schema $(+ x (= y =))$ represents the four programs $(+ x (+ y x))$, $(+ x (+ y y))$, $(+ x (* y x))$ and $(+ x (* y y))$.

In [18] a worst-case-scenario schema theorem was derived for GP with point mutation and one-point crossover.² *One-point crossover* works by using *the same* crossover point in both parent programs, and then swapping the corresponding subtrees like standard crossover. To account for the possible structural diversity of the two parents, the selection of the crossover point is restricted to the *common region*. The common region is the largest rooted region where the two parent trees have the same topology (see Figure 2). The common region will be defined formally in Section 3.

One-point crossover can be considered to be an instance of a much broader class of operators that can be defined through the notion of the common region. For example, in [17] we defined and studied a GP operator, called *uniform crossover* (based on uniform crossover in GAs). In uniform crossover the offspring is created by independently swapping the nodes in the common region with a uniform probability (see Figure 3). If a node belongs to the boundary of the common region and is a function then also the nodes below it are swapped, otherwise only the node label is swapped.³ Many other operators of this kind are possible. We will call them *homologous crossovers*, noting that our definition is more restrictive than that in [9]. A formal description of these operators will be given in Section 3.

The approximate schema theorem in [18] was improved in [16], where an exact schema theory for GP with one-point crossover was derived which was based on the notion of hyperschema.

Definition 2. Syntactically a GP hyperschema is a rooted tree composed of internal nodes from $\mathcal{F} \cup \{=\}$ and leaves from $\mathcal{T} \cup \{=, \#\}$. Semantically, $=$ is interpreted as a "don't

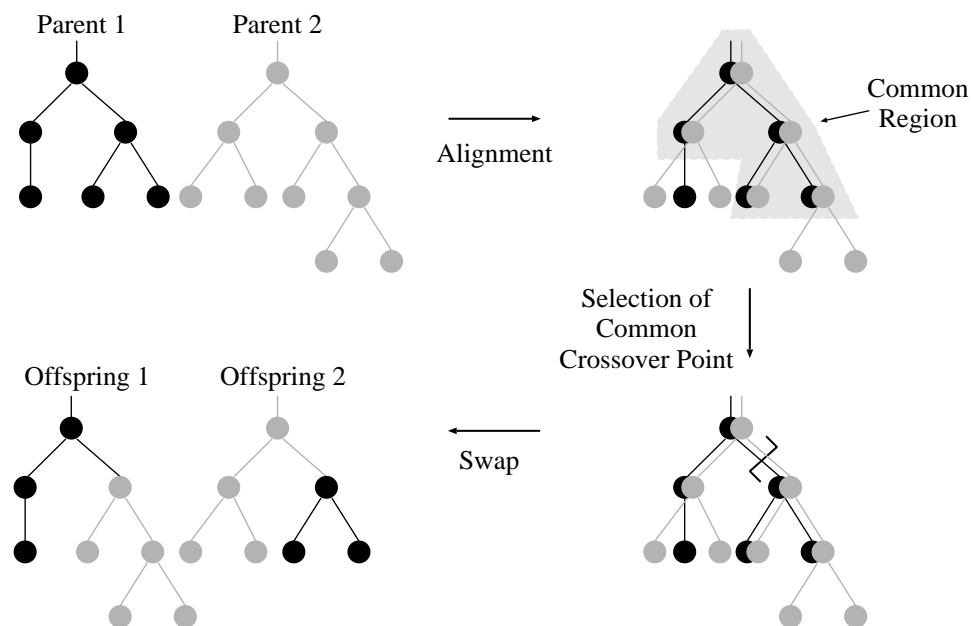


Figure 2. Example of GP one-point crossover. (Note, here we return two offspring in each crossover event. However, if not otherwise stated, normally we will assume that only the first offspring is returned.)

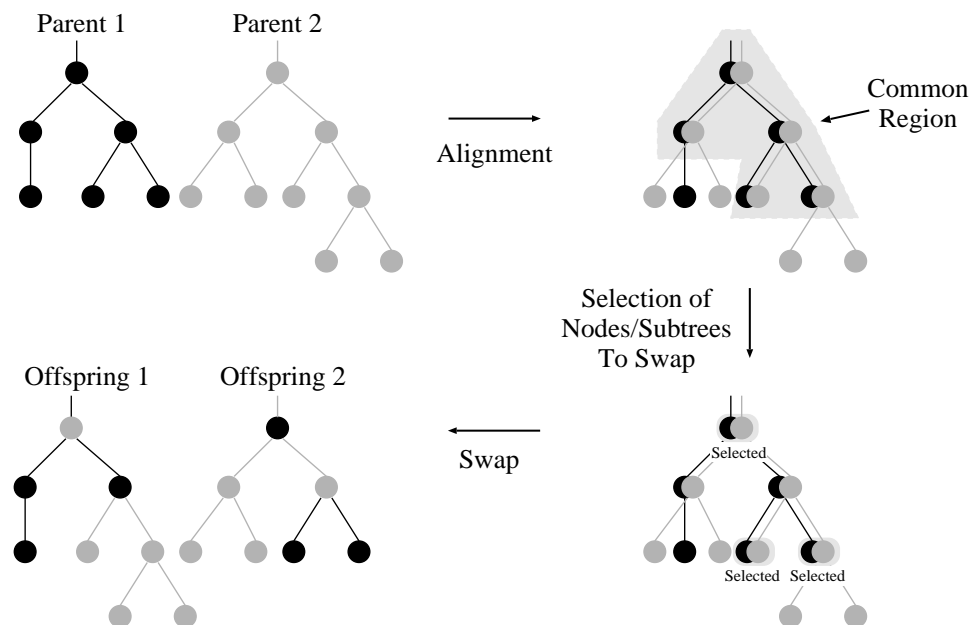


Figure 3. Example of GP uniform crossover.

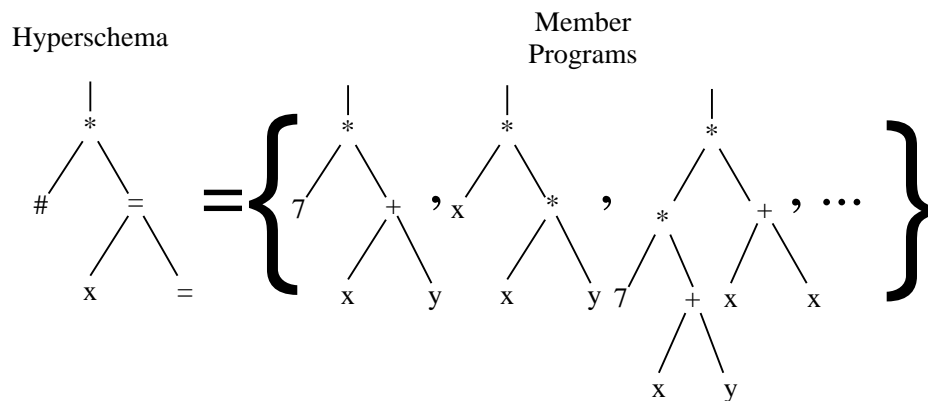


Figure 4. Sample hyperschema: its syntax (the tree on the left) and its semantics (the set of programs on the right).

care” symbols which stands for exactly one node, while # stands for any valid subtree. So, a hyperschema H is the set of all programs matching the syntax tree representing H .

For example, the hyperschema $(* \# (= x =))$ (see Figure 4) represents all the programs with the following characteristics: a) the root node is a product, b) the first argument of the root node is any valid subtree, c) the second argument of the root node is any function of arity two, d) the first argument of this function is the variable x , e) the second argument of the function is any valid node in the terminal set.

Hyperschemata are important because they make it possible to express in mathematical form the properties that two parent programs need to possess in order for them to produce offspring belonging to a schema of interest. These properties are expressed by the following two hyperschemata:

Definition 3. Given a schema H and a crossover point i , the lower building block, $L(H, i)$, is the hyperschema obtained by replacing all the nodes on the path between crossover point i and the root node with $=$ nodes, and all the subtrees connected to those nodes with # nodes. The upper building block, $U(H, i)$, is the hyperschema obtained by replacing the subtree below crossover point i with a # node. If a crossover point i is outside the schema H , then $L(H, i)$ and $U(H, i)$ are defined to be \emptyset (the empty set).

The steps involved in the construction of $L(H, i)$ and $U(H, i)$ for the schema $H = (* = (+ x =))$ are illustrated in Figure 5. The hyperschemata $L(H, i)$ and $U(H, i)$ are important because, if one crosses over at point i any individual in $L(H, i)$ with any individual in $U(H, i)$, the resulting offspring is always an instance of H .

The main result in [16] is the following:

THEOREM 1 For a population undergoing GP one-point crossover

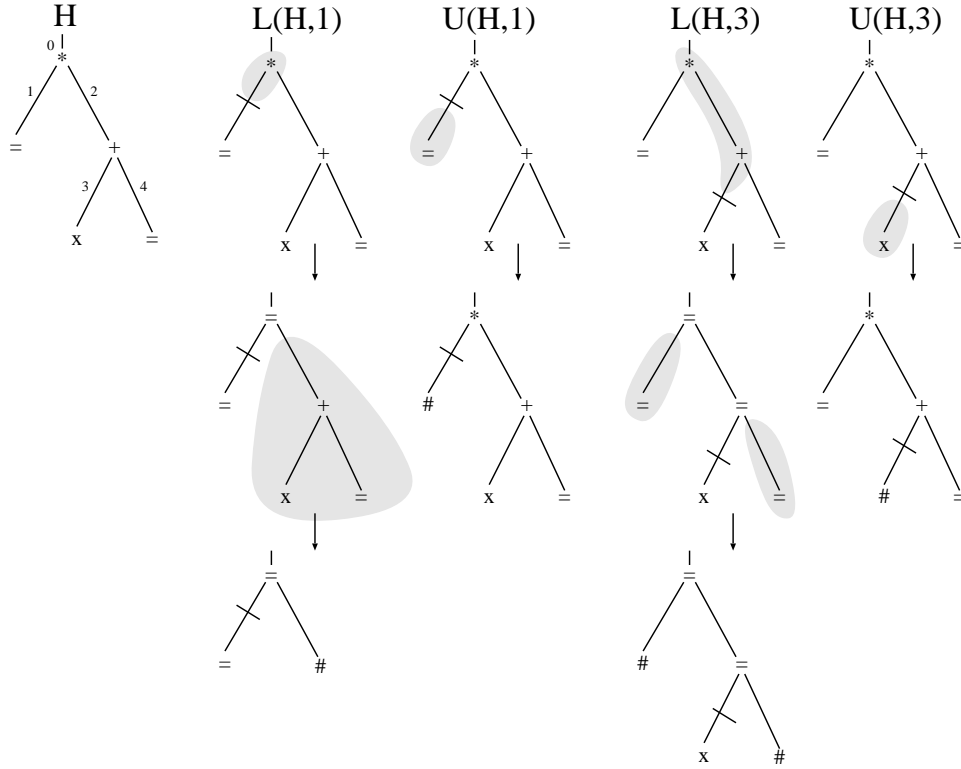


Figure 5. Example of a schema (H) and illustration of the construction of some of its potential hyperschema building blocks ($L(H, 1)$, $U(H, 1)$, $L(H, 3)$ and $U(H, 3)$). The crossover points in H are numbered as shown in the top left. The building block construction follows the procedure in Definition 3. For example, the hyperschema $L(H, 3)$ (at the bottom of the fourth column) is constructed by first identifying the path between crossover point 3 and the root node in a copy of H (see tree at the top of the fourth column), by replacing all the nodes on the path with “=” signs while at the same time identifying their other arguments (see tree in the middle of the fourth column) and finally replacing the subtrees connected to such nodes with “#” signs. The hyperschema $U(H, 3)$ (at the bottom of the fifth column) is constructed by first identifying the subtree rooted below crossover point 3 in H (see tree at the top of the fifth column) and then replacing such a subtree with a “#” sign. The hyperschemata at the bottom of columns two and three are constructed in similar ways.

$$\alpha(H, t) = (1 - p_{xo})p(H, t) + p_{xo}\alpha_{xo}(H, t) \quad (2)$$

and

$$\alpha_{xo}(H, t) = \sum_{k,l} \frac{1}{|C(G_k, G_l)|} \times \sum_{i \in C(G_k, G_l)} p(U(H, i) \cap G_k, t)p(L(H, i) \cap G_l, t) \quad (3)$$

where: p_{xo} is the crossover probability; $p(H, t)$ is the selection probability of the schema H ,⁴ G_1, G_2, \dots are an enumeration of all the possible schemata containing “=” signs only

(we will call these sets program shapes or just shapes hereafter); $|C(G_k, G_l)|$ is the number of nodes in the common region between shape G_k and shape G_l ; $C(G_k, G_l)$ is the set of indices of the crossover points in such a common region; \cap represents the intersection between sets.⁵

As discussed in [16, 10], it is possible to show that, in the absence of mutation, Equations 2 and 3 generalise and refine not only the GP schema theorem in [18] but also the version of Holland's schema theorem [7] presented in [6] and [46], as well as more recent exact GA schema theory [40, 41].

In Section 4 we will generalise Theorem 1 to uniform crossover and, in fact, to the whole class of GP homologous crossovers. However, before we can do so, we will need to formalise the notion of homologous crossover. We will do that in Section 3. This formalisation makes use of a reference system introduced in [21] where a general, exact schema theory for GP with subtree swapping crossover was presented. That theory is based on a generalisation of the notion of hyperschema (not described here being unnecessary for what follows) and on a Cartesian node reference system, which we describe below.

The *Cartesian node reference system* is obtained by considering the ideal infinite tree consisting entirely of nodes of some fixed maximum 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, and generally $(a_{max})^d$ nodes at depth d . Then one could imagine organising the nodes in the tree into layers of increasing depth and assigning an index to each node in a layer. The layer number d and the index i can then be used to define a Cartesian coordinate system. Clearly, one could also use this reference system to locate the nodes of non-maximal trees. This is possible because a non-maximal tree can always be described using a subset of the nodes and links in the maximal tree. For example, assuming $a_{max} = 2$, the root node of the schema $(* = (+ x =))$ (see top left corner of Figure 6) would have coordinates $(0,0)$, the $+$ would have coordinates $(1,1)$, etc. In this reference system it is always possible to find the route to the root node from any valid coordinate. Also, 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 that can be constructed with that function set. In the next section we will make ample use of this Cartesian reference systems.

The theory developed in [21, 22] using this reference system is very general. For example, it is applicable to standard GP crossover [8] with and without uniform selection of the crossover points, size-fair crossover [9], strongly-typed GP crossover [13], and many others including one-point crossover [18] which is a member of both the class of subtree-swapping crossovers and the class of homologous crossovers. The theory does not, however, cover the whole class of homologous operators and the first goal of this paper is to fill that theoretical gap.

3. Modelling Homologous Crossovers

In the previous section we have provided only an intuitive description of the class of homologous crossovers and of the concept of common region. In this section we will define these concepts rigorously.

Let us start with the common region. Given a node reference system it is possible to define functions over it. An important such function is the *arity function* $A(d, i, h)$ that returns the arity of the node at coordinates (d, i) in tree h . If applied to a pair of coordinates not in h , the function $A(d, i, h)$ returns -1 by convention to indicate the absence of a node at that position. For example, for the schema at the top left corner of Figure 6, $A(0, 0, h) = 2$, $A(1, 0, h) = 0$ and $A(2, 1, h) = -1$. Once the arity function is available, it is possible to define a *common region membership function* $\mathcal{C}(d, i, h_1, h_2)$ which returns **true** when (d, i) is part of the common region of trees h_1 and h_2 . We can do this by using the following recursive definition:

$$\mathcal{C}(d, i, h_1, h_2) = \begin{cases} \mathbf{true} & \text{if } (d, i) = \mathbf{root}, \\ \mathbf{true} & \text{if } A(\mathbf{parent}(d, i), h_1) = A(\mathbf{parent}(d, i), h_2) \neq 0, \\ & A(d, i, h_1) \geq 0, A(d, i, h_2) \geq 0, \text{ and} \\ & \mathcal{C}(\mathbf{parent}(d, i), h_1, h_2) = \mathbf{true}, \\ \mathbf{false} & \text{otherwise,} \end{cases}$$

where $\mathbf{root} = (0, 0)$, $\mathbf{parent}(d, i) = (d - 1, \lfloor i/a_{\max} \rfloor)$ and $\lfloor \cdot \rfloor$ is the integer-part function. This allows us to formally define the *common region* as the set:

$$C(h_1, h_2) = \{(d, i) \mid \mathcal{C}(d, i, h_1, h_2) = \mathbf{true}\}. \quad (4)$$

This is the notion of common region used in Theorem 1.

Because of the recursive nature of the common region membership function the nodes in the common region are always arranged so as to form a tree, and so it is possible to represent the common region as a tree or as an equivalent expression in prefix notation. For example, if the common region is the set of node coordinates $\{(0,0),(1,0),(1,1)\}$, which are organised as a small tree with just the root node and two leaves, then we might represent it using the prefix-notation expression $(? ? ?)$ (where we don't care about the semantics and arity of $?$ since only the shape and number of nodes in the tree is relevant).

As indicated before, one-point crossover selects the same crossover point in both parents by randomly choosing a node (or a link) in the common region. In the one-offspring version of this operator, the offspring is created by replacing the subtree rooted at such a node in the first parent with the subtree rooted at the same node, but in the second parent. An alternative way to interpret the action of one-point crossover is to imagine that we split $C(h_1, h_2)$ into two subsets, $C'(h_1, h_2)$ and $C''(h_1, h_2)$, such that $C'(h_1, h_2) \cap C''(h_1, h_2) = \emptyset$ and $C'(h_1, h_2) \cup C''(h_1, h_2) = C(h_1, h_2)$ and where $C'(h_1, h_2)$ is made up of the nodes in $C(h_1, h_2)$ below the chosen crossover point⁶, while $C''(h_1, h_2)$ includes all the remaining nodes of $C(h_1, h_2)$. Then, the nodes in $C'(h_1, h_2)$ are transferred from parent h_2 into an empty coordinate system, while the nodes in $C''(h_1, h_2)$ are taken from parent h_1 . Clearly, nodes representing the leaves of the common region (again interpreted as a tree) should be transferred together with their subtrees, if any. Other homologous crossovers can simply be defined by splitting C into C' and C'' differently. For example, uniform crossover builds C' and C'' by picking up the elements in C one at a time and flipping a coin to decide in which subset they should go.

While this description of homologous crossover is in the right direction for a formal definition of the class of these operators, it is not precise enough. This is because it does not state which are the possible ways in which each operator can split C into C' and C''

and how likely each (C', C'') pair is. A good way to address these problems is to extend the notions of crossover masks and recombination distributions used in genetics (e.g., [5]) and in the GA literature [3, 2, 37].

In a GA operating on fixed-length strings a crossover mask is simply a binary string. When crossover is executed, the bits of the offspring corresponding to the 1's in the mask will be taken from one parent, those corresponding to 0's from the other parent. For example, if the parents are the strings `aaaaaa` and `bbbbbb` and the crossover mask is `110100`, one offspring would be `aababb`. For operators returning two offspring it is easy to show that the second offspring can be obtained by simply complementing, bit by bit, the crossover mask. For example, the complement of the mask `110100`, `001011`, gives the offspring `bbabaa`. If the GA operates on strings of length N , then 2^N different crossover masks are possible. If, for each mask i , one defines a probability, p_i , that the mask is selected for crossover, then it is easy to see that many crossover operators can simply be interpreted as different ways of choosing the probability distribution p_i . For example, for strings of length $N = 4$ the probability distribution for one-point crossover would be $p_i = 1/3$ for the crossover masks $i = 1000, 1100, 1110$ and $p_i = 0$ otherwise, while for uniform crossover $p_i = 1/16$ for all 16 i 's. The probability distribution p_i is called a *recombination distribution*.

Let us now extend the notion of recombination distributions to genetic programming with homologous crossover. For any given shape and size of the common region we can define a set of *GP crossover masks* which correspond to all possible ways in which a recombination event can take place within the given common region. As noted above, the nodes in the common region are always arranged as a tree. So, GP crossover masks can be thought of as trees that have the same size and shape as the common region. These are obtained by labelling the nodes in the common region with 0's and 1's in all possible ways. Naturally, these trees can also be represented as a prefix-notation expressions. For example, if the common region is the set of node coordinates $\{(0,0),(1,0),(1,1)\}$ which, like we did earlier, we represent using the expression $(? ? ?)$, there are eight valid GP crossover masks: $(0 0 0)$, $(0 0 1)$, $(0 1 0)$, $(0 1 1)$, $(1 0 0)$, $(1 0 1)$, $(1 1 0)$ and $(1 1 1)$. Effectively, one can obtain these by replacing the ?'s in the expression representing the common region with 0's and 1's in all possible ways. The complement of a GP crossover mask is an obvious extension, where the complement \bar{i} has the same structure as mask i but with the 0's and 1's swapped. In the following we will use χ_c to denote the set of the $2^{N(c)}$ crossover masks associated with the common region c , where $N(c)$ is the number of nodes in c . Since we are typically interested in the common region defined by two trees, we'll use $\chi(h_1, h_2)$ as shorthand for $\chi_{C(h_1, h_2)}$. Note that each member of χ_c corresponds to a different way of splitting c into the two disjoint subsets C' and C'' mentioned earlier in this section.

Once χ_c is defined we can define a *fixed-size-and-shape recombination distribution* \hat{p} which gives the probability that crossover mask $i \in \chi_c$ will be chosen for crossover between individuals having common region c . Then the set $\{p_i^c \mid \forall c\}$, which we call a *GP recombination distribution*, completely defines the behaviour of a GP homologous crossover operator, different operators being characterised by different assignments for the p_i^c . For example, the GP recombination distribution for uniform GP crossover with 50% probabil-

ity of exchanging nodes is $p_i^c = (0.5)^{N(c)}$ for all masks $i \in \chi_c$. Each p_i^c represents the probability that a given (C', C'') pair will occur in a specific crossover operator.

GP crossover masks and GP recombination distributions generalise the corresponding GA notions, which are used extensively in [42]. Indeed, as also discussed in [16, 10], GAs operating on fixed-length strings are simply a special case of GP with homologous crossover. This can be shown by considering the case of function sets including only unary functions and initialising the population with programs of the same length. Since in a linear GP system with fixed length programs every individual has exactly the same size and (linear) shape, only one common region c is possible. Therefore, only one fixed-size-and-shape recombination distribution p_i^c is required to characterise crossover. In variable length GAs and GP, multiple fixed-size-and-shape recombination distributions are necessary, one for every possible common region c .

4. Exact GP Schema Theory for Homologous Crossovers

We start by defining a function that will help us describe the features the parents need to possess for them to produce offspring that are instances of a given schema.

Definition 4. Given a schema H and a crossover mask i , the building block generating function $\Gamma(H, i)$ returns the empty set if i contains any node not in H . Otherwise it returns the hyperschema obtained by replacing certain nodes in H with either = or # nodes: If a node in H corresponds to (i.e., has the same coordinates as) a non-leaf node in i that is labelled with a 0, then that node in H is replaced with a =; If a node in H corresponds to a leaf node in i that is labelled with a 0, then it is replaced with a #; All other nodes in H are left unchanged.

If, for example, $H = (* = (+ x =))$, as indicated in Figure 6(a), then $\Gamma(H, (0 1 0))$ is obtained by first replacing the root node with a = symbol (because the crossover mask has a function node 0 at coordinates (0,0)) and then replacing the subtree rooted at coordinates (1,1) with a # symbol (because the crossover mask has a terminal node 0 at coordinates (1,1)) obtaining $(= = \#)$. The hyperschema $\Gamma(H, (1 0 1))$ is instead obtained by replacing the subtree rooted at coordinates (1,0) with a # symbol obtaining $(* \# (+ x =))$, as illustrated in Figure 6(b).

Complementary pairs of hyperschemata $\Gamma(H, i)$ and $\Gamma(H, \bar{i})$, like the one in Figure 6, are generalisations of the schemata $L(H, i)$ and $U(H, i)$ used in Equation 2 (compare Figures 5 and 6). They are important because of the following

LEMMA 1 *If one crosses over using crossover mask i any individual in $\Gamma(H, i) \neq \emptyset$ with any individual in $\Gamma(H, \bar{i}) \neq \emptyset$, the resulting offspring is always an instance of H . Conversely, if two individuals cross using mask i to form an element of H then one of them must have come from $\Gamma(H, i) \neq \emptyset$ and the other from $\Gamma(H, \bar{i}) \neq \emptyset$.*

Proof: Because $\Gamma(H, i) \neq \emptyset$ and $\Gamma(H, \bar{i}) \neq \emptyset$, then i cannot include nodes not in H . Also, the non-leaf nodes in i must have the same arity as the corresponding nodes in H . Let us analyse the structure of the offspring, node by node. There are four possible situations:

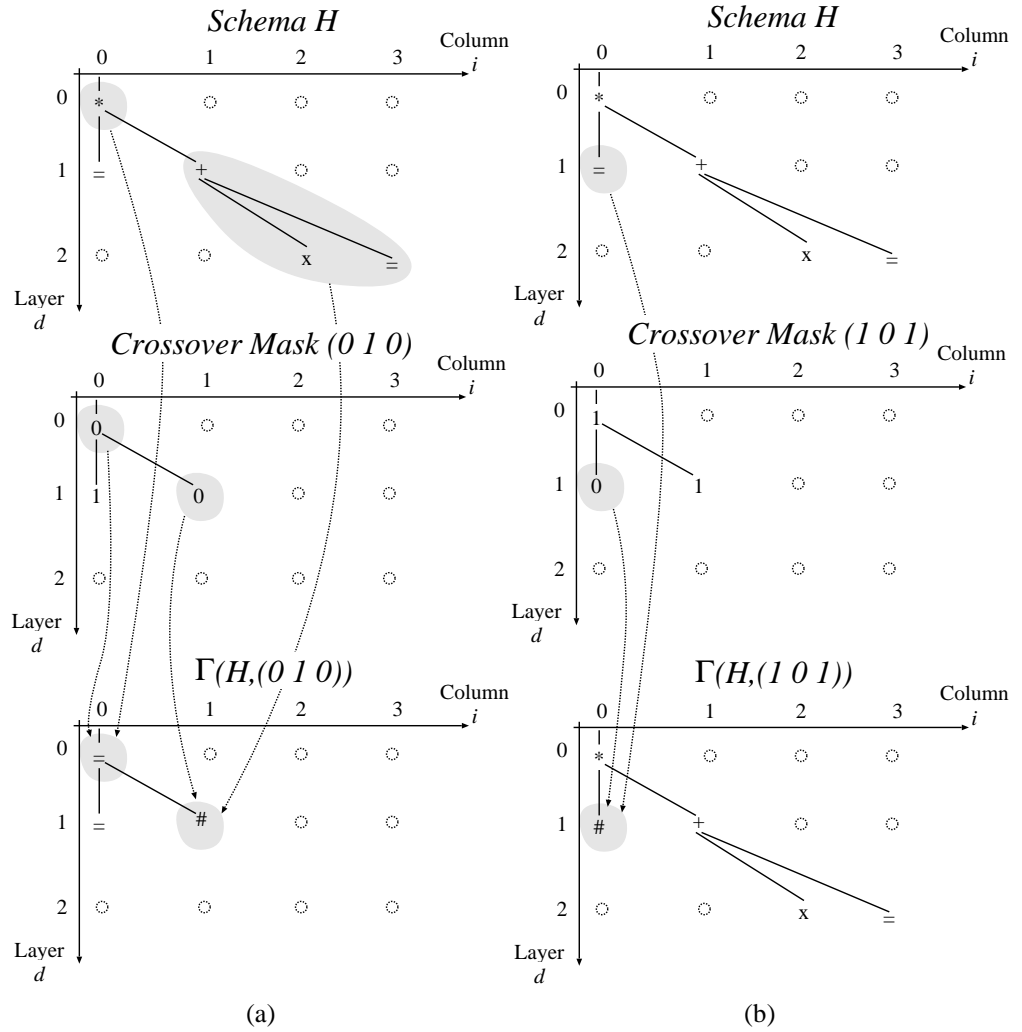


Figure 6. A complementary pair of hyperschemata $\Gamma(H, i)$ for the schema $H = (* = (+ x =))$. (Note that in the node reference systems shown here we have assumed that primitives could have at most an arity $a_{\max} = 2$.)

1. Firstly, let us consider the nodes in the offspring corresponding to non-leaf nodes in i labelled with 0's. When building the offspring, those nodes will be taken from h_2 which, as we know, belongs to $\Gamma(H, \bar{i})$. But by definition of $\Gamma(H, \bar{i})$ those nodes must match the corresponding symbols in H . So, the nodes of the offspring corresponding to non-leaf 0's in i will match the corresponding nodes in H .
2. Secondly, let us consider the nodes in the offspring corresponding to non-leaf nodes in i labelled with 1's. When building the offspring, those nodes will be taken from h_1 which, as we know, belongs to $\Gamma(H, i)$. But by definition of $\Gamma(H, i)$ those nodes must

match the corresponding symbols in H . So, the nodes of the offspring corresponding to non-leaf 1's in i will also match the corresponding nodes in H .

3. Thirdly, let us consider the nodes in the offspring belonging to subtrees rooted at the coordinates of leaf nodes in i labelled with 0's. When building the offspring, the subtree rooted at those nodes will be taken from h_2 which, as we know, belongs to $\Gamma(H, \bar{i})$. But by definition of $\Gamma(H, \bar{i})$ the nodes in that subtree must match the corresponding symbols in H . So, the subtrees of the offspring rooted at the same coordinates as terminal 0's in i will match corresponding subtrees in H .
4. Fourthly and finally, let us consider the nodes in the offspring belonging to subtrees rooted at the coordinates of leaf nodes in i labelled with 1's. The subtree rooted at those nodes will be taken from h_1 which, as we know, belongs to $\Gamma(H, i)$. But by definition of $\Gamma(H, i)$ the nodes in that subtree must match the corresponding symbols in H . So, the subtrees of the offspring rooted at the same coordinates as terminal 1's in i will match the corresponding subtrees in H .

To sum up, all the components (nodes or subtrees) of the offspring match corresponding components in the tree representing the schema H . So, the offspring belongs to H . Which proves the “if” part of the lemma. Let us now prove the “and only if” part.

Let us assume that we have found two individuals h_1 and h_2 which, when recombined using crossover mask i , produce an instance of H , but either $h_1 \notin \Gamma(H, i)$ or $h_2 \notin \Gamma(H, \bar{i})$ or both. Let us suppose, without loss of generality, that it is $h_1 \notin \Gamma(H, i)$. Then there must be at least one node in h_1 which does not match one of the symbols in $\Gamma(H, i)$. Let us denote its coordinates with (\hat{x}, \hat{y}) .

Let us define the function

$$\mathbf{ancestor}(d, i, h) = \begin{cases} \mathbf{root} & \text{if } (d, i) = \mathbf{root}, \\ \mathbf{parent}(d, i) & \text{if } h(\mathbf{parent}(d, i)) \neq \emptyset, \\ \mathbf{ancestor}(\mathbf{parent}(d, i), h) & \text{otherwise,} \end{cases}$$

where the notation $h(x, y)$ is meant to represent a function which returns the label of the node at coordinates (x, y) in tree h , with $h(x, y) = \emptyset$ if (x, y) is not in h .

We have three possibilities:

1. The node at coordinates (\hat{x}, \hat{y}) in $\Gamma(H, i)$ is empty and $\mathbf{ancestor}(\hat{x}, \hat{y}, \Gamma(H, i))$ is a non-leaf node of $\Gamma(H, i)$. Whether this is one of the nodes replaced with ‘=’ or it is one of the nodes originally in H , namely $H(\mathbf{ancestor}(\hat{x}, \hat{y}))$, it must have the same arity as $H(\mathbf{ancestor}(\hat{x}, \hat{y}))$, otherwise h_1 could not belong to H , having a different shape from H . But if this is the case, either that node is not $\mathbf{ancestor}(\hat{x}, \hat{y}, \Gamma(H, i))$ or h_1 has a shape different from H . Absurd, in either case.
2. The node at coordinates (\hat{x}, \hat{y}) in $\Gamma(H, i)$ is empty and $\mathbf{ancestor}(\hat{x}, \hat{y}, \Gamma(H, i))$ is a leaf node of $\Gamma(H, i)$. This node cannot be a # symbol, because, otherwise $h_1(\hat{x}, \hat{y})$ would match it, which goes against our assumption. So, $\Gamma(H, i)(\mathbf{ancestor}(\hat{x}, \hat{y}, \Gamma(H, i)))$ can only be a terminal symbol of H . But if this is the case, then crossover mask i must have a “1” leaf at $\mathbf{ancestor}(\hat{x}, \hat{y}, \Gamma(H, i))$, meaning that when building the offspring crossover will pick up the subtree rooted at

that node from parent h_1 . This subtree is not a single-node subtree since it must include also the node at coordinates (\hat{x}, \hat{y}) . Therefore, this subtree cannot match any terminal of H , and so any tree having it rooted at position $\mathbf{ancestor}(\hat{x}, \hat{y}, \Gamma(H, i))$ cannot match H . So, $h_1 \notin H$. Absurd.

3. The node at coordinates (\hat{x}, \hat{y}) in $\Gamma(H, i)$ is non-empty. Whether this is one of the nodes replaced with '=' or it is one of the nodes originally in H , namely $H(\hat{x}, \hat{y})$, it must have the same arity as $H(\hat{x}, \hat{y})$, otherwise h_1 could not belong to H , having a different shape from H . If $\Gamma(H, i)(\hat{x}, \hat{y})$ was one of the nodes replaced with '=' or $\Gamma(H, i)(\hat{x}, \hat{y}) = H(\hat{x}, \hat{y}) = '='$, then $h_1(\hat{x}, \hat{y})$ would match it. So, this cannot be the reason why $h_1(\hat{x}, \hat{y})$ does not match $\Gamma(H, i)(\hat{x}, \hat{y})$ and we are only left with one option: that $\Gamma(H, i)(\hat{x}, \hat{y}) = H(\hat{x}, \hat{y}) \neq '='$. But $h_1 \in H$, so $H(\hat{x}, \hat{y}) = h_1(\hat{x}, \hat{y})$. Therefore, also $\Gamma(H, i)(\hat{x}, \hat{y}) = h_1(\hat{x}, \hat{y})$, i.e. the node at coordinates (\hat{x}, \hat{y}) is not a mismatch. Absurd.

So, our assumption that $h_1 \notin \Gamma(H, i)$ cannot be true. Likewise for $h_2 \notin \Gamma(H, \bar{i})$. ■

Thanks to the notions of GP recombination distribution and building block generating function just proposed, we are now in a position to prove the following:

THEOREM 2 *The total transmission probability for a GP schema H under homologous crossover is given by Equation 2 with*

$$\alpha_{\text{so}}(H, t) = \sum_{h_1} \sum_{h_2} p(h_1, t)p(h_2, t) \sum_{i \in \chi(h_1, h_2)} p_i^{C(h_1, h_2)} \times \delta(h_1 \in \Gamma(H, i))\delta(h_2 \in \Gamma(H, \bar{i})) \quad (5)$$

where: the first two summations are over all the individuals in the population; $C(h_1, h_2)$ is the common region between program h_1 and program h_2 ; $\chi(h_1, h_2)$ is the set of crossover masks associated with $C(h_1, h_2)$; $\delta(x)$ is a function which returns 1 if x is true, 0 otherwise; \bar{i} is the complement of crossover mask i .

Proof: Let $p(h_1, h_2, i, t)$ be the probability that, at generation t , the selection-crossover process will choose parents h_1 and h_2 and crossover mask i . Then, let us consider the function

$$g(h_1, h_2, i, H) = \delta(h_1 \in \Gamma(H, i))\delta(h_2 \in \Gamma(H, \bar{i})).$$

From the previous lemma it follows that, given two parent programs, h_1 and h_2 , and a schema of interest H , this function returns the value 1 if and only if crossing over h_1 and h_2 with crossover mask i yields an offspring in H . It returns 0 otherwise. This function can be considered as a measurement function (see [2]) that we want to apply to the probability distribution of parents and crossover masks at time t , $p(h_1, h_2, i, t)$. If h_1 , h_2 and i are stochastic variables with joint probability distribution $p(h_1, h_2, i, t)$, the function $g(h_1, h_2, i, H)$ can be used to define a stochastic variable $\gamma = g(h_1, h_2, i, H)$. The expected value of γ is:

$$E[\gamma] = \sum_{h_1} \sum_{h_2} \sum_i g(h_1, h_2, i, H) p(h_1, h_2, i, t). \quad (6)$$

Since γ is a binary stochastic variable, its expected value also represents the proportion of times γ takes the value 1. This corresponds to the proportion of times the offspring of h_1 and h_2 are in H .

Because parent programs are selected independently and the probability of choosing a particular crossover mask does not vary over time for any two given parents, we can write

$$p(h_1, h_2, i, t) = p(i|h_1, h_2)p(h_1, t)p(h_2, t),$$

where $p(i|h_1, h_2)$ is the conditional probability that crossover mask i will be selected when the parents are h_1 and h_2 , while $p(h_1, t)$ and $p(h_2, t)$ are the selection probabilities for the parents. In homologous crossover $p(i|h_1, h_2) = p_i^{C(h_1, h_2)} \delta(i \in \chi(h_1, h_2))$, so

$$p(h_1, h_2, i, t) = p(h_1, t)p(h_2, t)p_i^{C(h_1, h_2)} \delta(i \in \chi(h_1, h_2)).$$

Substituting this into Equation 6 with minor simplifications leads to the expression of α_w in Equation 5. \blacksquare

Equations 2 and 5 allow one to compute the exact total transmission probability of a GP schema in terms of microscopic quantities. It is possible, however, to transform this model into the following exact macroscopic model of schema propagation

THEOREM 3 *The total transmission probability for a GP schema H under homologous crossover is given by Equation 2 with*

$$\alpha_w(H, t) = \sum_j \sum_k \sum_{i \in \chi(G_j, G_k)} p_i^{C(G_j, G_k)} \times p(\Gamma(H, i) \cap G_j, t) p(\Gamma(H, \bar{i}) \cap G_k, t), \quad (7)$$

where G_1, G_2, \dots are an enumeration of all the possible schemata containing = signs only.

Proof: Let us start by considering all the possible program shapes G_1, G_2, \dots . These schemata represent disjoint sets of programs. Their union represents the whole search space, so $\sum_j \delta(h_1 \in G_j) = 1$. We insert the l.h.s. of this expression and of an analogous expression for $\delta(h_2 \in G_k)$ in Equation 5 and reorder the terms obtaining:⁷

$$\begin{aligned} \alpha_w(H, t) &= \sum_j \sum_k \sum_{h_1} \sum_{h_2} p(h_1, t)p(h_2, t) \\ &\quad \sum_{i \in \chi(h_1, h_2)} p_i^{C(h_1, h_2)} \delta(h_1 \in \Gamma(H, i)) \delta(h_1 \in G_j) \delta(h_2 \in \Gamma(H, \bar{i})) \delta(h_2 \in G_k) \\ &= \sum_j \sum_k \sum_{h_1 \in G_j} \sum_{h_2 \in G_k} p(h_1, t)p(h_2, t) \\ &\quad \sum_{i \in \chi(h_1, h_2)} p_i^{C(h_1, h_2)} \delta(h_1 \in \Gamma(H, i)) \delta(h_2 \in \Gamma(H, \bar{i})) \end{aligned}$$

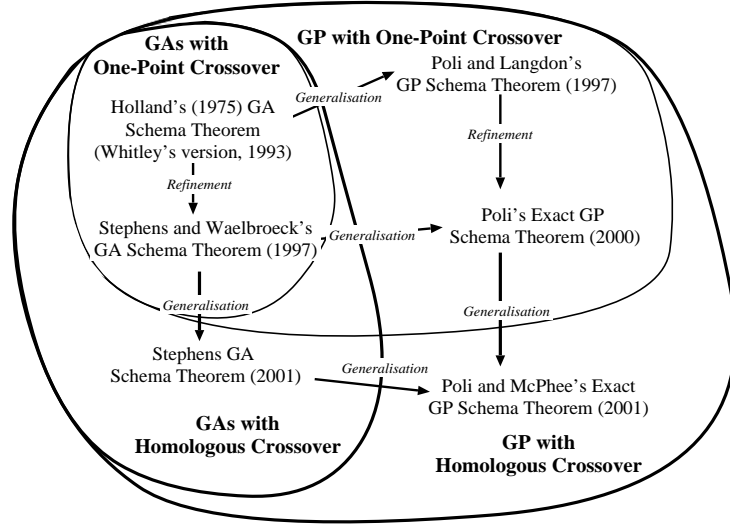


Figure 7. Relation between approximate and exact schema theorems for different representations and different forms of homologous crossover (in the absence of mutation).

$$\begin{aligned}
&= \sum_j \sum_k \sum_{h_1 \in G_j} \sum_{h_2 \in G_k} p(h_1, t) p(h_2, t) \\
&\quad \sum_{i \in \mathcal{X}(G_j, G_k)} p_i^{C(G_j, G_k)} \delta(h_1 \in \Gamma(H, i)) \delta(h_2 \in \Gamma(H, \bar{i})) \\
&= \sum_j \sum_k \sum_{i \in \mathcal{X}(G_j, G_k)} p_i^{C(G_j, G_k)} \sum_{h_1 \in G_j} p(h_1, t) \delta(h_1 \in \Gamma(H, i)) \\
&\quad \sum_{h_2 \in G_k} p(h_2, t) \delta(h_2 \in \Gamma(H, \bar{i})).
\end{aligned}$$

Since $\sum_{h_1 \in G_j} p(h_1, t) \delta(h_1 \in \Gamma(H, i)) = p(\Gamma(H, i) \cap G_j, t)$ (and similarly for $p(\Gamma(H, \bar{i}) \cap G_k, t)$), this equation completes the proof of the theorem. ■

This theorem is a generalisation of Theorem 1. This, as indicated in Section 2, is a generalisation of a recent GA schema theorem for one-point crossover [40, 41] and a refinement (in the absence of mutation) of both the GP schema theorem in [18] and Goldberg's version [6] of Holland's schema theory [7]. The schema theorems in this paper also generalise other GA results (such as those summarised in [46]), as well as the result in [2, appendix], since they can be applied to linear schemata and even fixed-length binary strings. So, in the absence of mutation, *the schema theory in this paper generalises and refines not only earlier GP schema theorems but also old and modern GA schema theories for one- and*

multi-point crossover, uniform crossover and all other homologous crossovers. The relations between various schema theoretic results are illustrated in Figure 7.

5. Background on Nix and Vose's Markov Chain Model of GAs

Now that we have developed an exact schema-based model for GP with homologous crossover, we turn our attention to a different kind of model: Vose's GA model. In this section we provide some necessary background on the this model, while in the next we extend it to GP using the schema theory developed earlier in the paper. The description of Vose's model provided here is largely based on [14, 43, 42]. See [12, 28] for gentler introductions to this topic.

Let Ω be the set of all possible strings of length l , i.e. $\Omega = \{0, 1\}^l$. Let $r = |\Omega| = 2^l$ be the number of elements of such a space. Let P be a population represented as a multiset of elements from Ω , let $n = |P|$ be the population size, and let $N = \binom{n+r-1}{r-1}$ be the number of possible populations. Let Z be an $r \times N$ matrix whose columns represent the possible populations of size n . The i th column $\Phi_i = \langle z_{0,i}, \dots, z_{r-1,i} \rangle^T$ of Z is the incidence vector for the i th population P_i . That is $z_{y,i}$ is the number of occurrences of string y in P_i (where y is unambiguously interpreted as an integer or as its binary representation depending on the context).

Once this state representation is available, one can model a GA with a Markov chain in which the N columns of Z represent the states of the model. The transition matrix for the model, Q , is an $N \times N$ matrix where the entry Q_{ij} represents the conditional probability that the next generation will be P_j assuming that the current generation is P_i .

In order to determine the values Q_{ij} let us assume that we know the probability $p_i(y)$ of producing individual y in the next generation given that the current generation is P_i . To produce population P_j we need to get exactly $z_{y,j}$ copies of string y for $y = 0, \dots, r-1$. The probability of this joint event is given by a multinomial distribution with success probabilities $p_i(y)$ for $y = 0, \dots, r-1$. So [38]

$$Q_{i,j} = \frac{n!}{z_{0,j}! z_{1,j}! \dots z_{r-1,j}!} \prod_{y=0}^{r-1} (p_i(y))^{z_{y,j}}. \quad (8)$$

The calculations necessary to compute the probabilities $p_i(y)$ depend crucially on the representation and the operators chosen. In [42] results for various GA crossover operators were reported. As noted in [14], it is possible to decompose the calculations using ideas firstly introduced in [43] as follows.

Assuming that the current generation is P_i , we can write

$$p_i(y) = \sum_{m,n=0}^{r-1} s_{m,i} s_{n,i} r_{m,n}(y) \quad (9)$$

where $r_{m,n}(y)$ is the probability that crossing over strings m and n yields string y and $s_{x,i}$ is the probability of selecting x from P_i . Assuming that fitness proportionate selection,

$$s_{x,i} = \frac{z_{x,i}f(x)}{\sum_{j=0}^{r-1} z_{j,i}f(j)}, \quad (10)$$

where $f(x)$ is the fitness of string x .

We can map these results into a more recent formulation of Vose's model [42] by making use of matrices and operators. We start by treating the fitness function as a vector f of components $f_k = f(k)$. Then, if x is the incidence vector representing a particular population, we define an operator \mathcal{F} , called the *selection scheme*,⁸ which computes the selection probabilities $s_{x,i}$ for all the members of Ω . For proportional selection $\mathcal{F}(x) = \text{diag}(f)x/f^T x$. Then we organise the probabilities $r_{m,n}(y)$ into r arrays M_y of size $r \times r$, called *mixing matrices*, the elements of which are $(M_y)_{m,n} = r_{m,n}(y)$. We finally define an operator \mathcal{M} , called the *mixing scheme*,

$$\mathcal{M}(x) = \langle x^T M_0 x, x^T M_1 x, \dots, x^T M_{r-1} x \rangle,$$

that returns a vector whose components are the expected proportion of individuals of each type assuming that individuals are selected from the population x randomly (with replacement) and crossed over.

Finally we introduce the operator $\mathcal{G} = \mathcal{M} \circ \mathcal{F}$, which provides a compact way of expressing the probabilities $p_i(y)$ since (for fitness proportionate selection)

$$p_i(y) = \{\mathcal{G}(\Phi_i)\}_y = \left\{ \mathcal{M} \left(\frac{\text{diag}(f)\Phi_i}{f^T \Phi_i} \right) \right\}_y$$

where the notation $\{\cdot\}_y$ is used to represent the y th component of a vector. So, the entries of the transition matrix for the Markov chain model of a GA can concisely be written as

$$Q_{i,j} = n! \prod_{y=0}^{r-1} \frac{(\{\mathcal{G}(\Phi_i)\}_y)^{z_{y,j}}}{z_{y,j}!}. \quad (11)$$

In [43, 14, 42] it is shown how, for fixed-length binary GAs, the operator \mathcal{M} can be calculated as a function of the mixing matrix M_0 only. This is done by using a set of permutation operators that permute the components of any generic vector $x \in \mathbb{R}^r$:

$$\sigma_j \langle x_0, \dots, x_{r-1} \rangle^T = \langle x_{j \oplus 0}, \dots, x_{j \oplus r-1} \rangle^T, \quad (12)$$

where \oplus is a bitwise XOR.⁹ Then one can write

$$\mathcal{M}(x) = \langle (\sigma_0 x)^T M_0 \sigma_0 x, \dots, (\sigma_{r-1} x)^T M_0 \sigma_{r-1} x \rangle^T. \quad (13)$$

6. Vose-like Model for GP

In order to extend Vose's model to GP and variable-length GAs with homologous crossover we define Ω to be an indexed set of all possible trees of maximum depth ℓ that can be constructed with a given function set \mathcal{F} and a given terminal set \mathcal{T} . Assuming that the initialisation algorithm selects programs in Ω , GP with homologous crossover cannot produce

programs outside Ω , and Ω is therefore a finite search space. Again, $r = |\Omega|$ is the number of elements in the search space; this time, however, r is not 2^l . All other quantities defined in Section 5 can be redefined by simply replacing the word “string” with the word “program”, provided that the elements of Ω are indexed appropriately. With these extensions, all the equations in that section are also valid for GP, except Equations 12 and 13.

These are all minor changes. A major change is instead required to compute the probabilities $p_i(y)$ of generating the y th program in Ω when the population is P_i . Fortunately, these probabilities can be computed by applying the schema theory developed in Section 4. Since schema equations are applicable to individual programs as well as to schemata, it is clear that:

$$p_i(y) = \alpha(y, t) \quad (14)$$

where α is calculated for population P_i . This can be done by specialising Equations 2 and 7. Doing this allows one to instantiate the transition matrix for the model using Equation 8. However, it is possible to express $p_i(y)$ in terms of more primitive quantities:

THEOREM 4 *For GP homologous crossover*

$$p_i(y) = \sum_{m,n=0}^{r-1} s_{m,i} s_{n,i} r_{m,n}(y)$$

where

$$r_{m,n}(y) = \left[(1 - p_{x_o}) \delta(m = y) + p_{x_o} \sum_{l \in \mathcal{X}(m,n)} p_l^{C(m,n)} \delta(m \in \Gamma(y, l)) \delta(n \in \Gamma(y, \bar{l})) \right] \quad (15)$$

Proof: Let us specialise Equations 2 and 7 for the y th program in Ω :

$$\begin{aligned} p_i(y) &= (1 - p_{x_o}) p(y, t) + \\ & p_{x_o} \sum_j \sum_k \sum_{l \in \mathcal{X}(G_j, G_k)} p_l^{C(G_j, G_k)} \times p(\Gamma(y, l) \cap G_j, t) p(\Gamma(y, \bar{l}) \cap G_k, t) \\ &= (1 - p_{x_o}) \sum_{m \in \Omega} \delta(m = y) p(m, t) \times \underbrace{\sum_{n \in \Omega} p(n, t)}_{=1} \\ & + p_{x_o} \sum_j \sum_k \sum_{l \in \mathcal{X}(G_j, G_k)} p_l^{C(G_j, G_k)} \times \\ & \sum_{m \in \Omega} p(m, t) \delta(m \in \Gamma(y, l)) \delta(m \in G_j) \times \sum_{n \in \Omega} p(n, t) \delta(n \in \Gamma(y, \bar{l})) \delta(n \in G_k) \\ &= \sum_{m,n \in \Omega} p(m, t) p(n, t) \times \\ & \left[(1 - p_{x_o}) \delta(m = y) + p_{x_o} \sum_{l \in \mathcal{X}(m,n)} p_l^{C(m,n)} \times \delta(m \in \Gamma(y, l)) \delta(n \in \Gamma(y, \bar{l})) \right], \end{aligned}$$

where we used the fact that $\sum_w \delta(x \in G_w) = 1$.

Assuming the current population at generation t is P_t , we have that $p(h, t) = s_{h,i}$. ■

Note that Equation 15 could have been obtained by direct calculation, rather than through the specialisation of a schema theorem. However, this would still have required the definition and use of the building block generating function Γ and of the concepts of GP crossover masks and GP recombination distributions. Also, notice that the set of GP crossover masks also includes masks containing all ones. These correspond to cloning the first parent. Therefore, by suitable readjustment of the probabilities $p_i^{C(m,n)}$, we can rewrite Equation 15 as

$$r_{m,n}(y) = \sum_{l \in \mathcal{X}(m,n)} p_l^{C(m,n)} \delta(m \in \Gamma(y, l)) \delta(n \in \Gamma(y, \bar{l})). \quad (16)$$

This formula is analogous to the case of crossover defined by masks for fixed-length binary strings [42].

We are now in a position to show the mathematical equivalence between exact schema-based models and Vose-like models. Here, we have effectively instantiated a Vose-like model for GP and variable-length GAs by using the exact schema theory for homologous crossover. However, had we known the values of the probabilities $p_i(y)$ by some type of direct calculation and assuming the population at generation t is P_t , we could have easily obtained the total transition probability for any schema of interest H for that population just by adding together the appropriate $p_i(y)$'s, i.e. $\alpha(H, t) = \sum_{y \in H} p_i(y)$.

7. Mixing Matrices for 0/1 Trees

As has already been stated in Section 5, for the case of fixed-length binary strings, the mixing operator \mathcal{M} can be written in terms of a single mixing matrix M_0 and a set of permutation matrices. This works because the permutation matrices are a representation of a group (in the mathematical sense of the word) that acts transitively on the search space. This group action describes the symmetries that are inherent in the definition of crossover for fixed-length strings [42]. This idea can be generalised to other finite search spaces (see [31] for the detailed theory). However, in the case of GP, where the search space is a set of trees (up to some depth), the symmetry is more complex and does not seem to give rise to a single mixing matrix.

In this section we will look at what symmetry does exist and the simplifications of the mixing operator it produces when we restrict ourselves to the space of *0/1 trees*. These are trees constructed using primitives from a terminal set $\mathcal{T} = \{0_0, 1_0\}$ and from a function set $\mathcal{F} = \bigcup_{i \in \iota} \mathcal{F}_i$ where $\mathcal{F}_i = \{0_i, 1_i\}$, ι is a finite subset of \mathbb{N} , and the subscripts 0 and i represent the arity of a 0/1 primitive.¹⁰ It should be noted that the semantics of the primitives in 0/1 trees is unimportant for the theory. So, 0/1 trees can actually represent a variety of program search spaces, if the elements of \mathcal{F} and \mathcal{T} are interpreted as instructions. For example, we could interpret 0_0 as the variable x , 1_0 as the variable y , 0_2 as $+$ and 1_2 as $*$. Then, if we represent 0/1 trees in prefix notation, the 0/1 tree $(1 (0 0 1) 0)$ would

in fact correspond to the program $(* (+ x y) x)$. More generally 0/1 trees represent all the GP search spaces where the primitive set includes always *exactly* two primitives for any give arity. Note also that 0/1 trees are a generalisation of the notion of binary strings.¹¹

Let Ω be the set of 0/1 trees of depth at most ℓ (where a program containing only a terminal has depth 1). Let $L(\Omega)$ be the set of full trees of exactly depth ℓ obtained by using the primitive set $\mathcal{T} \cup \mathcal{F}_{i_m}$ where i_m is the maximum element in ι . We term *node-wise XOR* the operation which, given two trees a and b in $L(\Omega)$, returns the 0/1 tree whose nodes are labelled with the result of the addition (modulo 2) of the binary labels of the nodes in a and b having corresponding coordinates; this operator is denoted $a \oplus b$. For example,

$$(1 (1 0 1) (0 0 1)) \oplus (0 (1 0 0) (0 1 1)) = (1 (0 0 1) (0 1 0)).$$

$L(\Omega)$ is a group under node-wise XOR. Notice that the definition of \oplus extends naturally to pairs of trees with identical size and shape.

For each tree $k \in \Omega$ we define a truncation function

$$\pi_k : L(\Omega) \longrightarrow \Omega$$

as follows. Given any tree $a \in L(\Omega)$ we match up the nodes in k with the nodes in a , recursively:

1. The root nodes are matched.
2. The children of a matched node in k are matched to children of the corresponding node in a from the left. Recall that each node in a has the maximum possible arity, and that a has the maximum possible depth. Note that the arity of nodes in a will be reduced (if necessary) to that of the matching nodes in k .

This procedure corresponds to matching by node co-ordinates. The effect of the operator π_k on a tree $a \in L(\Omega)$ is to throw away all nodes that are not matched against nodes in k . The remaining tree $\pi_k(a)$ will then be of the same size and shape as k .

For example, suppose the maximum depth is $\ell = 3$ and the maximum arity is also 3. Let $a \in L(\Omega)$ be the tree $(1 (0 1 1 0) (1 0 1 1) (1 1 1 0))$ and let $k = (0 (1 1 0) (0 1))$. Then matching nodes and truncating a produces $\pi_k(a) = (1 (0 1 1) (1 0))$.

The group $L(\Omega)$ acts on the elements of Ω as follows. Let $a \in L(\Omega)$ and $k \in \Omega$. Then define $a(k) = \pi_k(a) \oplus k$ which means we apply addition modulo 2 on each matched pair of nodes. We have used the extended definition of \oplus since $\pi_k(a)$ and k are guaranteed to have the same size and shape. In our previous example we would have $a(k) = (1 (1 0 1) (1 1))$.

We can extend the definition of \oplus further by setting $a \oplus k = a(k)$ for any $k \in \Omega$ and $a \in L(\Omega)$. The effect of this is essentially a relabelling of the nodes of the tree k in accordance with the pattern of ones and zeros found in a .

For each $a \in L(\Omega)$ we define a corresponding $r \times r$ permutation matrix σ_a with

$$(\sigma_a)_{i,j} = \delta((a \oplus i) = j) \tag{17}$$

LEMMA 2 *Let $m, n, y \in \Omega$ and let $a \in L(\Omega)$. Then for homologous crossover*

$$r_{m,n}(y) = r_{a \oplus m, a \oplus n}(a \oplus y)$$

Proof: Interpreting Equation 15 for 0/1 trees m, n and y , the following hold:

$$\begin{aligned} a \oplus m &= a \oplus y \iff m = y \\ C(a \oplus m, a \oplus n) &= C(m, n) \\ (a \oplus m) \in \Gamma(a \oplus y, l) &\iff m \in \Gamma(y, l) \end{aligned}$$

and the result follows. The third assertion follows from the fact that we are relabelling the nodes in tree m according to the pattern of ones in a , and we relabel the nodes in the hyperschema $\Gamma(y, l)$ according to exactly the same pattern. ■

Let us consider the GP schema G consisting only of “=” nodes representing the shape of some of the programs in Ω . We denote with 0^G the element of Ω obtained by replacing the = nodes in G with 0 nodes.

THEOREM 5 *On the space of 0/1 trees with depth at most ℓ homologous crossover gives rise to a mixing operator*

$$\mathcal{M}(x) = \langle x^T M_0 x, x^T M_1 x, \dots \rangle$$

(where we are indexing vectors by the elements of Ω). Then for each fixed shape G of depth not bigger than ℓ there exists a mixing matrix $M = M_{0^G}$ such that if $y \in \Omega$ is of shape G then $M_y = \sigma_a^T M \sigma_a$ for some $a \in L(\Omega)$, where σ_a is defined in Equation 17.

Proof: Let $y \in \Omega$ be of shape G as required. Construct a maximal full tree a of depth not bigger than ℓ by appending a sufficient number of 0 nodes to the tree y so that each internal node in a has i_m children.¹²

Now suppose $m, n \in \Omega$ are trees which cross together to form y with probability $r_{m,n}(y)$. Because crossover is assumed to be homologous, the set of the coordinates on the nodes in m must be a superset of the set of node coordinates of G . Likewise for n .

The m, n th component of $\sigma_a^T M \sigma_a$ is

$$\begin{aligned} (\sigma_a^T M \sigma_a)_{m,n} &= \sum_v (\sigma_a^T M)_{m,v} (\sigma_a)_{v,n} \\ &= \sum_v \sum_w (\sigma_a)_{w,m} M_{w,v} (\sigma_a)_{v,n} \\ &= M_{a^{-1} \oplus m, a^{-1} \oplus n} \\ &= r_{a^{-1} \oplus m, a^{-1} \oplus n}(0^G) \\ &= r_{m,n}(a \oplus 0^G) \\ &= r_{m,n}(y \oplus 0^G) \\ &= r_{m,n}(y) \\ &= (M_y)_{m,n} \end{aligned}$$

where we have used the lemma to show $r_{a^{-1} \oplus m, a^{-1} \oplus n}(0^G) = r_{m,n}(a \oplus 0^G)$ and a^{-1} is the inverse of the group element a . For 0/1 trees $a^{-1} = a$ since $a \oplus a = 0^{G_m}$, where G_m is the schema representing the shape of the trees in $L(\Omega)$. ■

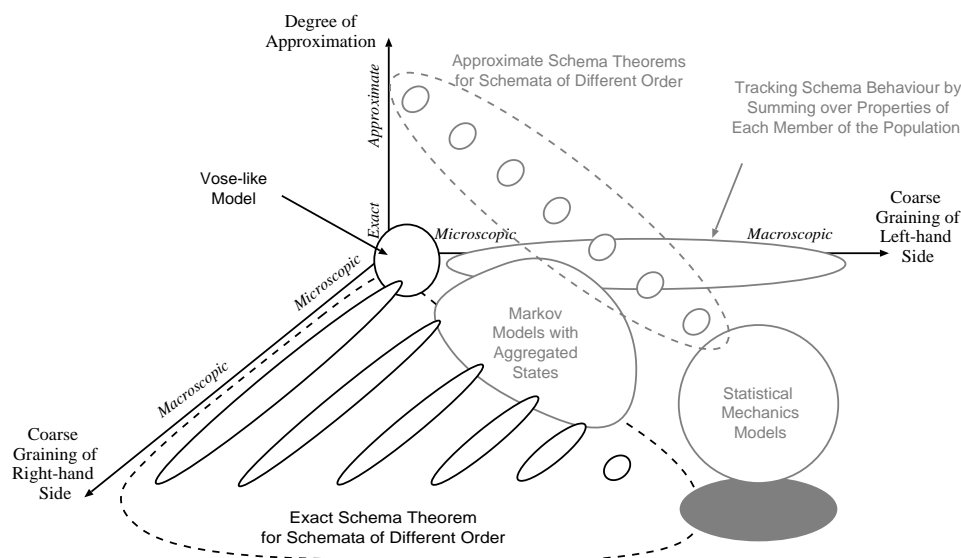


Figure 8. Existing models (black) and possible future models (grey) for GP and variable-length GAs with homologous crossover.

8. Discussion

As shown in Figure 8, with the two models presented in this paper, the space of possible models for GP and variable-length GAs using homologous crossover starts being populated. Following the theoretical developments seen for fixed-length GAs, hopefully in the future we will also be able to produce a variety of other kinds of models, e.g. approximate models like Holland's schema theorem or the models obtained using the statistical mechanics approach (e.g., see [26]).

The theory presented here doesn't just fill the space of GP models. It also provides the theoretical foundations for a complete unification of the world of fixed-length GAs, variable-length GAs and GP.¹³ In the absence of mutation the theory in this paper generalises perhaps the two most important results obtained to date in the theory for fixed-length GAs: Vose's model and Stephens' schema theorem. This is because if one uses only unary functions and the population is initialised with programs having a fixed common length, a GP system using homologous crossover is entirely equivalent to a GA acting on fixed-length strings. This unification is important because it opens up the possibility of generalising to GP and variable-length GAs a variety of previous results obtained in the last decade either by using Vose's model or some form of schema theory. For example, it should now become possible to see whether there are any conditions under which the transition matrix for our Markov chain model is ergodic (which would guarantee the convergence of the system towards the global optimum).¹⁴ Also, any new results proven using

the models presented here will have an immediate effect on the understanding of a variety of systems including fixed-length binary GAs, non-binary GAs, variable-length GAs, linear GP systems and tree-based GP systems.

The development of theory for evolutionary algorithms operating on structures with variable size and shape is complex, and simply getting to the point of *stating* exact models for these algorithms requires a lot of machinery. In this paper we have space only to state our exact models and illustrate (in Appendix A) how they can be used in simple examples. However, we are fully aware that modelling for the sake of modelling is not a good use of one's time: models of this kind are as good as the predictions and understanding they produce, and so in future research we will need to prove that our models are indeed highly predictive and explanatory. Because these results are so new, we don't want to make bold claims here. Naturally, the fact that these models are generalisations of highly regarded pre-existing GA models suggests that we will not be disappointed. Indeed, recently we have started using our exact schema evolution equations to understand the dynamics of GP or variable-length GAs with homologous crossover or to design competent GP/GA systems [24, 25]. In other recent work, we have specialised and applied the theory for other operators to understand phenomena such as operator biases and the evolution of size in variable length GAs [19, 11], and in the future we hope to be able to do the same and produce exciting new results with the theory presented here.

As regards our Markov chain model for GP, in the paper we analysed in detail the case of 0/1 trees (which include variable length binary strings and a variety of program spaces), where symmetries can be exploited to obtain further simplifications in the model. The similarity with Vose's GA model is very clear in this case. This is only a first step. In future research we intend to analyse in more depth the general case of tree-like structures to try to identify symmetries in the mixing matrices similar to those found for 0/1 trees. Also, we intend to study the characteristics of the transition matrices for the GP model, to gain insights into the dynamics of GP.

9. Conclusions

Unlike GA theory, which has made considerable progress in the 1990s, in the same period of time GP theory has typically been scarce, approximate and, as a rule, not terribly useful. This is not surprising given the complexities of building theories for variable size structures. In the last three years or so, however, significant breakthroughs have changed this situation radically. Today not only do we have exact schema theorems for GP with a variety of operators including subtree mutation, headless chicken crossover, standard crossover, one-point crossover, and all other sub-tree swapping crossovers, but this GP theory also generalises and refines a broad spectrum of GA theory.

We believe that this paper extends this series of breakthroughs. Here we have presented a schema theory applicable to GP and both variable- and fixed-length GAs with homologous crossovers: a set of operators where the offspring are created preserving the position of the genetic material taken from the parents. The theory is based on the concepts of GP crossover masks and GP recombination distributions, which are natural generalisations of the corresponding concepts in the theory for fixed-length GAs and in population genetics.

In this paper we have also presented a Vose-like model of GP and variable-length GAs. Obtaining this model has been possible thanks to the developments of the GP schema theory in Section 4, which has given us exact formulas for computing the probability that reproduction and recombination will create any specific program in the search space. A GP Markov chain model is then easily obtained by plugging this ingredient into a minor extension of Vose’s model of GAs. This theoretical approach provides an alternative framework for studying the dynamics of evolutionary algorithms (in terms of transient and long-term behaviour). It also makes explicit the relationship between the local action of genetic operators on individuals and the global behaviour of the population.

The theory presented in this paper generalises and/or refines a huge variety of results previously reported in the GA and GP literature, providing a framework for a complete unification of evolutionary computation theory. Important extensions and applications of this framework seem almost uncountable.

Acknowledgments

The authors would like to thank the anonymous reviewers and the Editor-in-Chief for their excellent suggestions for improving this manuscript. Nic McPhee 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 Poli would like to thank the members of the NEC (Natural and Evolutionary Computation) group at Essex for helpful comments and discussion.

Appendix A

A.1. Schema Theory Examples

In this section we provide two examples that show how to use the schema theory developed in this paper in practice and illustrate some of its benefits. To make the relationship between this work and our previous theory for one-point crossover clearer, we will use the same examples as in [16], this time using general homologous crossover operators instead of just one-point crossover.

A.1.1. Linear Trees

Since the calculations involved in applying exact GP schema theorems can become quite lengthy, we start with an extremely simple example.

Let us imagine that we have a function set $\{A_f, B_f, C_f, D_f, E_f\}$ including only unary functions, and the terminal set $\{A_t, B_t, C_t, D_t, E_t\}$. Since, all functions are unary, we can unambiguously represent expressions without parenthesis. In addition, since the only terminal in each expression is the rightmost node, we can remove the subscripts without generating any ambiguity. For example, $(A_f(B_f(A_f D_t))) \equiv A_f B_f A_f D_t \equiv ABAD$. Thus, every member of the search space can be seen as a variable-length string over the alphabet

$\{A, B, C, D, E\}$, and GP with homologous crossover is really a non-binary variable-length GA.

Let us now consider the schema AB=. We want to measure its total transmission probability (with $p_{xo} = 1$) under fitness proportionate selection and an arbitrary homologous crossover operator for the following population:

Population	Fitness
AB	2
BCD	2
ABC	4
ABCD	6

We will use Equation 7. In order to apply it we first need to number all the possible program shapes G_1, G_2 , etc.. Let G_1 be =, G_2 be ==, G_3 be === and G_4 be ==. We do not need to consider other, larger shapes because the population does not contain any larger programs. We then need to evaluate the shape of the common regions to determine $\chi(G_j, G_k)$ for all valid values of j and k . In this case the common regions can be naturally represented using integers that represent the length of the common region. Since the length of the common region is the length of the shorter parent, we know $C(G_j, G_k) = \min(j, k)$. Then, for each common region c we need to identify the hyperschemata $\Gamma(\text{AB=}, i)$ for all the meaningful crossover masks $i \in \chi_c$ and calculate $\Gamma(\text{AB=}, i) \cap G_j$ for all meaningful values of j . These calculations are shown in Table A.1. Using this table we can apply Equation 7, obtaining, after simplification and omitting t and the superscript c from p_i^c for brevity,

$$\begin{aligned}
\alpha(\text{AB=}) &= \alpha_{xo}(\text{AB=}) \\
&= \sum_{j,k=1}^4 \sum_{i \in \{0,1\}^{\min(j,k)}} p_i p(\Gamma(H, i) \cap G_j) p(\Gamma(H, \bar{i}) \cap G_k) \\
&= (p_0 + p_1) p(\text{AB=}) p(=) + \\
&\quad (p_{00} + p_{11}) p(\text{AB=}) p(==) + (p_{01} + p_{10}) p(= B=) p(A=) + \\
&\quad (p_{000} + p_{111}) p(\text{AB=}) (p(===) + p(====)) + \\
&\quad (p_{001} + p_{110}) p(===) (p(\text{AB=}) + p(\text{AB==})) + \\
&\quad (p_{010} + p_{101}) p(A==) (p(= B=) + p(= B==)) + \\
&\quad (p_{011} + p_{100}) p(= B=) (p(A==) + p(A===)).
\end{aligned}$$

This equation is valid for any homologous crossover operator, each of which is defined by a different set of p_i 's. Let us specialise it for GP uniform crossover, whose recombination distribution is $p_i = (0.5)^{N(i)}$, where $N(i)$ is the length of crossover mask i , as indicated in Section 3. We obtain:

$$\begin{aligned}
\alpha(\text{AB=}) &= p(\text{AB=}) p(=) + \\
&\quad \frac{1}{2} p(\text{AB=}) p(==) + \frac{1}{2} p(= B=) p(A=) +
\end{aligned}$$

Table A.1. Crossover masks and schemata necessary to calculate $\alpha_{wv}(A \ B \ =)$.

Mask i	$\Gamma(AB=, i)$	$\Gamma(AB=, i) \cap G_j$			
		$j = 1$	$j = 2$	$j = 3$	$j = 4$
0	#	=	==	===	====
1	AB=	\emptyset	\emptyset	AB=	\emptyset
00	=#	\emptyset	==	===	====
01	=B=	\emptyset	\emptyset	=B=	\emptyset
10	A#	\emptyset	A=	A==	A===
11	AB=	\emptyset	\emptyset	AB=	\emptyset
000	==#	\emptyset	\emptyset	===	====
001	===	\emptyset	\emptyset	===	\emptyset
010	=B#	\emptyset	\emptyset	=B=	=B==
011	=B=	\emptyset	\emptyset	=B=	\emptyset
100	A=#	\emptyset	\emptyset	A==	A===
101	A==	\emptyset	\emptyset	A==	\emptyset
110	AB#	\emptyset	\emptyset	AB=	AB==
111	AB=	\emptyset	\emptyset	AB=	\emptyset
0000	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

$$\begin{aligned} & \frac{1}{4}p(AB=)(p(===) + p(====)) + \\ & \frac{1}{4}p(===)(p(AB=) + p(AB==)) + \\ & \frac{1}{4}p(A==)(p(=B=) + p(=B==)) + \\ & \frac{1}{4}p(=B=)(p(A==) + p(A===)). \end{aligned}$$

If instead we specialise it for one-point crossover (using the recombination distribution¹⁵ $p_0 = 1, p_{00} = p_{10} = 1/2, p_{01} = p_{11} = 0, p_{000} = p_{100} = p_{110} = 1/3$, etc.) we obtain

$$\begin{aligned} \alpha(AB=) &= p(AB=)p(=) + \\ & \frac{1}{2}p(AB=)p(==) + \frac{1}{2}p(=B=)p(A=) + \\ & \frac{1}{3}p(AB=)(p(===) + p(====)) + \\ & \frac{1}{3}p(===)(p(AB=) + p(AB==)) + \\ & \frac{1}{3}p(=B=)(p(A==) + p(A===)), \end{aligned}$$

which is the same result as in [16, 10].

If we now calculate the quantities in the last two equations for the population considered here, we obtain $\alpha(AB=, t) \approx 0.2806$ for uniform crossover, while $\alpha(AB=, t) \approx 0.2925$ for one-point crossover, which indicates that uniform crossover is slightly less “friendly” towards the schema.

A.1.2. Analysis of Schema Equations for Binary Trees

Let us now consider a GP system using functions of arity 2, with individuals of up to depth 2 (considering the root node as being at depth 0). Then, the schema evolution equation for the tree $(A B C)$ (expressed in Lisp-like prefix notation) is:

$$\begin{aligned}
& \alpha((A B C)) \\
&= (p_1 + p_0) \mathcal{P}((A B C)) \mathcal{P}(=) + \\
&\quad (p_{(111)} + p_{(000)}) \mathcal{P}((A B C)) \mathcal{P}((= (= =) (= =))) + \\
&\quad (p_{(111)} + p_{(000)}) \mathcal{P}((A B C)) \mathcal{P}((= (= =) =)) + \\
&\quad (p_{(111)} + p_{(000)}) \mathcal{P}((A B C)) \mathcal{P}((= (= =))) + \\
&\quad (p_{(110)} + p_{(001)}) \mathcal{P}((= (= =) C)) \mathcal{P}((A B (= =))) + \\
&\quad (p_{(110)} + p_{(001)}) \mathcal{P}((= (= =) C)) \mathcal{P}((A B =)) + \\
&\quad (p_{(110)} + p_{(001)}) \mathcal{P}((= (= =) C)) \mathcal{P}((A B (= =))) + \\
&\quad (p_{(101)} + p_{(010)}) \mathcal{P}((= B (= =))) \mathcal{P}((A (= =) C)) + \\
&\quad (p_{(101)} + p_{(010)}) \mathcal{P}((= B (= =))) \mathcal{P}((A = C)) + \\
&\quad (p_{(101)} + p_{(010)}) \mathcal{P}((= B =)) \mathcal{P}((A (= =) C)) + \\
&\quad (p_{(101)} + p_{(010)}) \mathcal{P}((= B =)) \mathcal{P}((A = C)) + \\
&\quad (p_{(011)} + p_{(100)}) \mathcal{P}((= B C)) \mathcal{P}((A (= =) (= =))) + \\
&\quad (p_{(011)} + p_{(100)}) \mathcal{P}((= B C)) \mathcal{P}((A (= =) =)) + \\
&\quad (p_{(011)} + p_{(100)}) \mathcal{P}((= B C)) \mathcal{P}((A = (= =))) + \\
&\quad (p_{(011)} + p_{(100)}) \mathcal{P}((= B C)) \mathcal{P}((A = =)).
\end{aligned}$$

It is interesting to specialise this equation for the shape $(= = =)$ by substituting A, B and C with $=$. Noting that $p_1 + p_0 = 1 = p_{(000)} + p_{(001)} + p_{(010)} + p_{(011)} + p_{(100)} + p_{(101)} + p_{(110)} + p_{(111)}$ and that $\mathcal{P}(=) + \mathcal{P}((= = =)) + \mathcal{P}((= (= =))) + \mathcal{P}((= (= =) =)) + \mathcal{P}((= (= =) (= =))) = 1$ we obtain

$$\begin{aligned}
& \alpha((= = =)) = \\
&\quad \mathcal{P}((= = =)) + \\
&\quad (p_{(001)} + p_{(010)} + p_{(101)} + p_{(110)}) \times \\
&\quad \left(\mathcal{P}((= (= =) =)) \mathcal{P}((= (= =))) \right. \\
&\quad \quad \left. - \mathcal{P}((= = =)) \mathcal{P}((= (= =) (= =))) \right).
\end{aligned}$$

If we rewrite the equation using trees rather than a Lisp-like notation, we obtain

$$\alpha(\text{A}) = p(\text{A}) + (p_{(001)} + p_{(010)} + p_{(101)} + p_{(110)}) \times \left(p(\text{A})p(\text{B}) - p(\text{A})p(\text{C}) \right). \quad (\text{A.1})$$

In the case of one-point crossover this equation becomes

$$\alpha(\text{A}) = p(\text{A}) + \frac{2}{3} \left(p(\text{A})p(\text{B}) - p(\text{A})p(\text{C}) \right),$$

as reported in [16].

Equation A.1 is quite interesting. It states that shape A will evolve (on average) as if selection only was acting on it, but only if the product $p(\text{A})p(\text{B})$ is exactly the same as the product $p(\text{A})p(\text{C})$. Why?

When one crosses over two parents belonging to any of the four shapes A , B , C , and D , only offspring having one of these shapes can be produced. For example, when one crosses over an instance of the schema A with an instance of the schema B , there are four possible outcomes: an instance of B , an instance of C , an instance of A (if only the root node is exchanged and the parent donating the root is in A) or an instance of D (if only the root node is exchanged and the parent donating the root is in B). Likewise, if one crosses over an instance of the schema B with an instance of the schema A , there are exactly the same four possible outcomes. In all other possible combinations of parents only offspring with the same shapes as one of their parents are produced. So, there is a constant migration of individuals from one shape to another (see Figure A.1). Until the probabilities of creating all these instances are exactly balanced, there will have to be a net flow of individuals among the four schemata A , B , C , and D .

If we compare Equation A.1 with the schema evolution equations for the schemata B , C , and D , it is easy to convince ourselves that equilibrium is reached only when

$$p(\text{A})p(\text{B}) = p(\text{A})p(\text{C}),$$

at which point A will evolve as if selection only was acting on it. Indeed, the schema evolution equations for the schemata B and C have the form

$$\alpha(H) = p(H) + \beta \Delta p$$

while the schema evolution equations for the schemata D and A have the form

$$\alpha(H) = p(H) - \beta \Delta p$$

where

$$\Delta p = p(\text{B})p(\text{C}) - p(\text{A})p(\text{D})$$

and

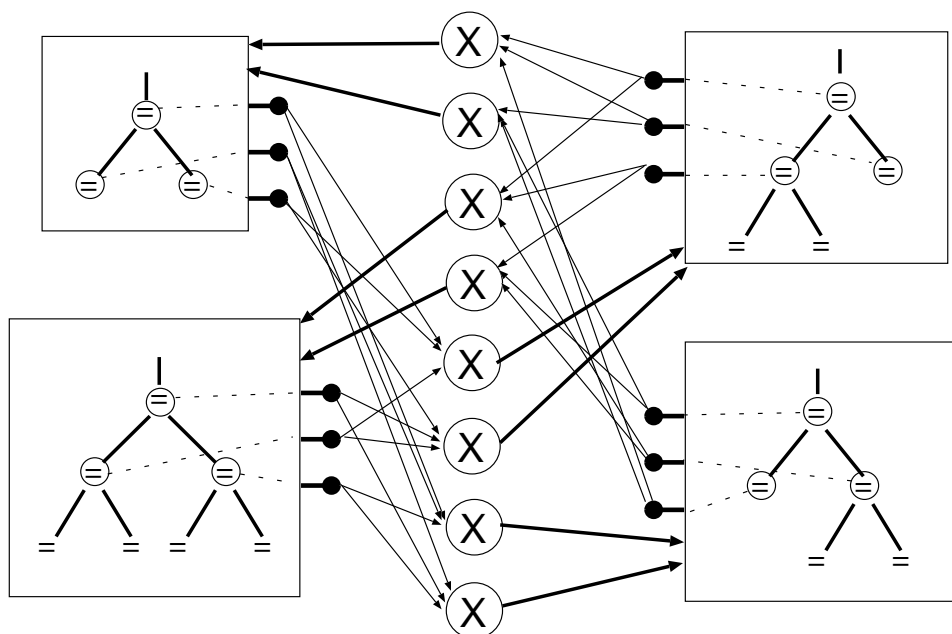
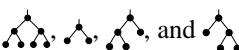



Figure A.1. Different ways in which homologous crossover can “exchange” individuals between shape classes. Each circled cross represents one way of performing crossover. The thin arrows indicate which nodes are transferred from which parent shape in each such event, while the thick arrows indicate the shape of the resulting offspring. Note that here we show only the subset of crossover events that produce offspring with shapes different from those of their parents.

$$\beta = p_{(001)} + p_{(010)} + p_{(101)} + p_{(110)}.$$

So, evolution will tend to make the value of Δp approach 0. Indeed, $\Delta p = 0$ is a necessary (but not sufficient) condition for evolution to reach a fixed point. This condition can be considered to be something like a *linkage equilibrium equation for shapes*. Until this form of equilibrium is reached evolution will continue *even on a flat landscape*. This is due to the bias imposed by homologous crossovers.

When $\Delta p = 0$ the evolution equations of the schemata  become:

$$\alpha(H) = p(H)$$

and evolution proceeds as if selection only was acting, as already noted above for the schema . So, at that point homologous crossovers become unbiased. It should be noted, however, that different homologous crossovers move towards the condition $\Delta p = 0$ at different speeds. For example, for one-point crossover $\beta = \frac{2}{3}$ while for uniform crossover $\beta = \frac{1}{2}$. So, GP one-point crossover is more aggressive than GP uniform crossover in pushing program shapes towards this form of shape equilibrium. This is surprising, since

we know from fixed-length-GA theory that uniform crossover pushes the *allele* distribution towards linkage equilibrium faster than one-point crossover. However, this appears not to be true generally in GP as far as program shapes are concerned (note that it may well be true as far as the *primitive* distribution is concerned).

The components $\beta_{\mathcal{P}}(\text{tree}_1)_{\mathcal{P}}(\text{tree}_2)$ and $\beta_{\mathcal{P}}(\text{tree}_1)_{\mathcal{P}}(\text{tree}_2)$ in the schema equations for tree_1 and tree_2 can be interpreted as *schema creation* and *schema disruption* probabilities (the role is reversed for tree_2 and tree_1). The component $p(H)$ in such equations could be interpreted as the probability of *schema survival* to crossover.

A.2. GP Markov Chain Model Examples

A.2.1. Calculation of the Markov Chain Transition Matrix

In this section we will provide a simple example to illustrate how to calculate the elements of the transition matrix for Markov chain the model introduced in Section 6.

Let us consider the function set $\mathcal{F}=\{+, *\}$ and the terminal set $\mathcal{T}=\{x, y\}$, and let us limit the maximum depth of programs to $\ell = 2$. With these constraints the search space Ω includes only the following $r = 10$ different programs: $x, y, (+ x x), (* x x), (+ y x), (* y x), (+ x y), (* x y), (+ y y), (* y y)$. For this example we will assume that the fitness of each of these programs equals the number of nodes in the program. So, y has fitness 1, while $(+ x x)$ has fitness 3. If we further assume that the population size is $n = 3$, then there are $N = \binom{3+10-1}{9-1} = \binom{12}{9} = 220$ different populations. So, the Markov chain's transition matrix Q will be a 220×220 matrix.¹⁶

Each element $Q_{i,j}$ of the matrix represents the probability that if the population in the current generation is population P_i the next generation's population will be P_j . Let us consider the case where $P_i = \{x, (+ x y), (* x x)\}$.

The incidence vector for P_i is $\Phi_i = \langle 1, 0, 0, 1, 0, 0, 1, 0, 0, 0 \rangle^T$. Using the non-zero elements $z_{l,i}$ of Φ_i and the fitnesses for the programs in the population we can calculate the selection probabilities $s_{x,i} = \frac{1}{7}$, $s_{(+ x y),i} = s_{(* x x),i} = \frac{3}{7}$. Naturally, the selection probabilities for all the other members of Ω can only be 0.

In order to calculate the probabilities $p_i(y)$ we would now need to calculate the probabilities $r_{m,n}(y)$ for all possible values of m, n and y . In general there are $10 \times 10 \times 10 = 1000$ of these. However, since only 5 different types of programs can be created via homologous crossover starting from population P_i and $s_{m,i} \times s_{n,i} \neq 0$ only for 9 (m, n) pairs, only a much smaller number of $r_{m,n}(y)$'s need to be calculated. Assuming $p_{x_0} = 1$, here are the values for the necessary $r_{m,n}(y)$'s:

$$\begin{aligned} r_{x,x}(x) &= 1 \\ r_{x,(+ x y)}(x) &= p_1 \\ r_{x,(* x x)}(x) &= p_1 \\ r_{(+ x y),x}(x) &= p_0 \\ r_{(* x x),x}(x) &= p_0 \\ r_{x,(+ x y)}((+ x y)) &= p_0 \\ r_{(+ x y),x}((+ x y)) &= p_1 \end{aligned}$$

$$\begin{aligned}
r_{(+ \times Y), (+ \times Y)}((+ \times Y)) &= 1 \\
r_{(+ \times Y), (* \times X)}((+ \times Y)) &= p_{(101)} + p_{(111)} \\
r_{(* \times X), (+ \times Y)}((+ \times Y)) &= p_{(000)} + p_{(010)} \\
r_{X, (* \times X)}((* \times X)) &= p_0 \\
r_{(* \times X), X}((* \times X)) &= p_1 \\
r_{(+ \times Y), (* \times X)}((* \times X)) &= p_{(000)} + p_{(010)} \\
r_{(* \times X), (+ \times Y)}((* \times X)) &= p_{(101)} + p_{(111)} \\
r_{(* \times X), (* \times X)}((* \times X)) &= 1 \\
r_{(* \times X), (+ \times Y)}((+ \times X)) &= p_{(001)} + p_{(011)} \\
r_{(+ \times Y), (* \times X)}((+ \times X)) &= p_{(100)} + p_{(110)} \\
r_{(* \times X), (+ \times Y)}((* \times Y)) &= p_{(100)} + p_{(110)} \\
r_{(+ \times Y), (* \times X)}((* \times Y)) &= p_{(001)} + p_{(011)}
\end{aligned}$$

By feeding these values and the selection probabilities calculated above into the expression for $p_i(y)$, we obtain (we expand the calculation only for $p_i(x)$):

$$\begin{aligned}
p_i(x) &= \frac{1}{7} \times \frac{1}{7} \times 1 + \frac{1}{7} \times \frac{3}{7} \times p_1 + \frac{1}{7} \times \frac{3}{7} \times p_1 + \\
&\quad \frac{3}{7} \times \frac{1}{7} \times p_0 + \frac{3}{7} \times \frac{1}{7} \times p_0 \\
&= \frac{1 + 6 \overbrace{(p_0 + p_1)}^{=1}}{49} = \frac{1}{7} \\
p_i((+ \times Y)) &= \frac{12 + 9(p_{(000)} + p_{(010)} + p_{(101)} + p_{(111)})}{49} \\
p_i((* \times X)) &= \frac{12 + 9(p_{(000)} + p_{(010)} + p_{(101)} + p_{(111)})}{49} \\
p_i((+ \times X)) &= \frac{9(p_{(001)} + p_{(011)} + p_{(100)} + p_{(110)})}{49} \\
p_i((* \times Y)) &= \frac{9(p_{(001)} + p_{(011)} + p_{(100)} + p_{(110)})}{49}
\end{aligned}$$

All other $p_i(y)$'s are 0 (indeed $p_i(x) + p_i((+ \times Y)) + p_i((* \times X)) + p_i((+ \times X)) + p_i((* \times Y)) = 1$, since $p_{(000)} + \dots + p_{(111)} = 1$). Let us specialise these equations for uniform crossover (by using the recombination distribution $p_{(000)} = \dots = p_{(111)} = 1/8$):

$$\begin{aligned}
p_i(x) &= \frac{14}{98} \\
p_i((+ \times Y)) &= \frac{33}{98} \\
p_i((* \times X)) &= \frac{33}{98} \\
p_i((+ \times X)) &= \frac{9}{98}
\end{aligned}$$

$$p_i((* \times Y)) = \frac{9}{98}$$

Let us further imagine that we want to calculate $Q_{i,j}$ when $\bar{P}_j = \{ (+ \times Y), (+ \times Y), (* \times Y) \}$. Using the probabilities $p_i(\cdot)$ just calculated and the incidence vector for population P_j , $\Phi_j = \langle 0, 0, 0, 0, 0, 0, 2, 1, 0, 0 \rangle^T$, the expression for $Q_{i,j}$ yields

$$\begin{aligned} Q_{i,j} &= \frac{3!}{0!0!0!0!0!2!1!0!0!} \left(\frac{14}{98}\right)^0 0^0 \left(\frac{9}{98}\right)^0 \left(\frac{33}{98}\right)^0 0^0 0^0 \left(\frac{33}{98}\right)^2 \left(\frac{9}{98}\right)^1 0^0 0^0 \\ &= 3 \frac{33^2 \times 9}{98^3} = \frac{29403}{941192} \approx 3.1\%. \end{aligned}$$

By considering all possible 220 incidence vectors Φ_j one can simply compute a whole row of Q . It is easy to see that the most likely successors of P_i are the populations $\{ (+ \times Y), (+ \times Y), (* \times X) \}$ and $\{ (+ \times Y), (* \times X), (* \times X) \}$ for which

$$Q_{i,j} = \frac{3!}{2!1!} \left(\frac{33}{98}\right)^3 = \frac{107811}{941192} \approx 11.5\%.$$

The probability of the population remaining the same is instead

$$Q_{i,i} = \frac{3!}{1!1!1!} \left(\frac{14}{98}\right) \left(\frac{33}{98}\right)^2 = \frac{91476}{941192} \approx 9.7\%.$$

As indicated at the end of the previous section, the quantities $p_i(y)$ can also be used to compute the expected proportion of any schema of interest in the next generation (under the assumption that the current generation is P_i). For example,

$$\alpha((= \times Y), t) = p_i((+ \times Y)) + p_i((* \times Y)) = \frac{42}{98} = \frac{3}{7}.$$

A.2.2. Mixing Matrices

In this section we will demonstrate the application of the mixing matrix theory for 0/1 trees (Section 7) to an example. To keep the presentation of the calculations manageable this example must perforce be quite simple, but should still be sufficient to illustrate the key concepts.

For this example we will assume that the function set contains only unary functions, with the possible labels for both functions and terminals being 0 and 1 (i.e., $\mathcal{F} = \mathcal{F}_1 = \mathcal{T} = \{0, 1\}$). As a result we can think of our structures as being variable length binary strings. We will let $\ell = 2$, which means that $r = 6$ and

$$\Omega = \{0, 1, 00, 01, 10, 11\}.$$

We will also limit ourselves here to the mixing matrices for GP one-point crossover and GP uniform crossover; we could however readily extend the example to any other homologous crossover operator.

A.2.3. GP one-point crossover

The key to applying this theory is to compute $r_{m,n}(y)$ as described in Equation 15. In other words, for each $y \in \Omega$ we need to construct a matrix $M_y = r_{m,n}(y)$ that contains the probabilities that GP one-point crossover with parents m and n will yield y . Since $r = |\Omega| = 6$, this will yield six 6×6 matrices. In the (fixed-length) GA case it would only be necessary to specify one mixing matrix, since symmetries would allow us to derive the others through permutations of the indices. As indicated in the previous section, the symmetries in 0/1 trees case are more complex, and one cannot reduce the situation down to just one case. In particular we find, as mentioned above, that the set of mixing matrices for our variable-length GA case splits into two different subsets, one for y of length 1, and one for y of length 2, and the necessary permutations are generated by the group $L(\Omega) = \{00, 01, 10, 11\}$.

To make this more concrete, let us consider M_0 and M_1 , each of which has exactly one non-zero column:¹⁷

$$M_0 = \begin{bmatrix} & | & 0 & 1 & 00 & \dots \\ 0 & | & 1 & 0 & 0 & \dots \\ 1 & | & 1 & 0 & 0 & \dots \\ 00 & | & 1/2 & 0 & 0 & \dots \\ 01 & | & 1/2 & 0 & 0 & \dots \\ 10 & | & 1/2 & 0 & 0 & \dots \\ 11 & | & 1/2 & 0 & 0 & \dots \end{bmatrix} \quad M_1 = \begin{bmatrix} & | & 0 & 1 & 00 & \dots \\ 0 & | & 0 & 1 & 0 & \dots \\ 1 & | & 0 & 1 & 0 & \dots \\ 00 & | & 0 & 1/2 & 0 & \dots \\ 01 & | & 0 & 1/2 & 0 & \dots \\ 10 & | & 0 & 1/2 & 0 & \dots \\ 11 & | & 0 & 1/2 & 0 & \dots \end{bmatrix}$$

Clearly M_1 is very similar to M_0 . Indeed, Theorem 5 shows that M_1 can be obtained by applying a permutation matrix to M_0 :

$$M_1 = \sigma_{10}^T M_0 \sigma_{10},$$

where

$$\sigma_{10}^T = \begin{bmatrix} & | & 0 & 1 & 00 & 01 & 10 & 11 \\ 0 & | & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & | & 1 & 0 & 0 & 0 & 0 & 0 \\ 00 & | & 0 & 0 & 0 & 0 & 1 & 0 \\ 01 & | & 0 & 0 & 0 & 0 & 0 & 1 \\ 10 & | & 0 & 0 & 1 & 0 & 0 & 0 \\ 11 & | & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The situation is more interesting for the mixing matrices for y of length 2:

$$M_{00} = \begin{bmatrix} & | & 0 & 1 & 00 & 01 & 10 & 11 \\ 0 & | & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & | & 0 & 0 & 1 & 0 & 0 & 0 \\ 00 & | & 0 & 0 & 1 & 0 & 1/2 & 0 \\ 01 & | & 0 & 0 & 1 & 0 & 1/2 & 0 \\ 10 & | & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 11 & | & 0 & 0 & 1/2 & 0 & 0 & 0 \end{bmatrix} \quad M_{01} = \begin{bmatrix} & | & 0 & 1 & 00 & 01 & 10 & 11 \\ 0 & | & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & | & 0 & 0 & 0 & 1 & 0 & 0 \\ 00 & | & 0 & 0 & 0 & 1 & 0 & 1/2 \\ 01 & | & 0 & 0 & 0 & 1 & 0 & 1/2 \\ 10 & | & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 11 & | & 0 & 0 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

$$M_{10} = \left[\begin{array}{c|cccccc} & 0 & 1 & 00 & 01 & 10 & 11 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 00 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 01 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 10 & 0 & 0 & 1/2 & 0 & 1 & 0 \\ 11 & 0 & 0 & 1/2 & 0 & 1 & 0 \end{array} \right] \quad M_{11} = \left[\begin{array}{c|cccccc} & 0 & 1 & 00 & 01 & 10 & 11 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 00 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 01 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 10 & 0 & 0 & 0 & 1/2 & 0 & 1 \\ 11 & 0 & 0 & 0 & 1/2 & 0 & 1 \end{array} \right]$$

Here again we can write these mixing matrices as permutations of M_{00} , i.e.,

$$M_s = \sigma_s^T M_{00} \sigma_s$$

for $s \in \{00, 01, 10, 11\}$. M_{01} , for example, can be written as

$$M_{01} = \sigma_{01}^T M_{00} \sigma_{01}$$

where σ_{01} is as above.

A.2.4. GP uniform crossover

Here will just show the mixing matrices M_0 and M_{00} since, as we have seen, the other four matrices can be readily obtained from these using the permutation matrices σ_s :

$$M_0 = \left[\begin{array}{c|cccccc} & 0 & 1 & 00 & 01 & 10 & 11 \\ \hline 0 & 1 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 00 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 01 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 10 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1/2 & 0 & 0 & 0 & 0 & 0 \end{array} \right]$$

$$M_{00} = \left[\begin{array}{c|cccccc} & 0 & 1 & 00 & 01 & 10 & 11 \\ \hline 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 00 & 1/2 & 1/2 & 1 & 1/2 & 1/2 & 1/4 \\ 01 & 0 & 0 & 1/2 & 0 & 1/4 & 0 \\ 10 & 0 & 0 & 1/2 & 1/4 & 0 & 0 \\ 11 & 0 & 0 & 1/4 & 0 & 0 & 0 \end{array} \right]$$

Comparing these matrices to those obtained for one-point crossover one can see that these are symmetric, where those for one-point crossover were not, pointing out that uniform crossover is symmetric with respect to the parents, where one-point crossover is not. The matrices for uniform crossover also have considerably more non-zero entries than those for one-point crossover, highlighting the fact that uniform crossover provides more ways to construct any given string.

Notes

1. This paper is the result of the integration and substantial extension of two conference papers [20, 23].
2. As discussed in [16, 10], this theorem is a generalisation of the version of Holland's schema theorem [7] presented in [6] and [46] to variable size structures.
3. When we say that a node is "below" another node (or a link) we mean that it belongs to the subtree rooted in such a node (or connected to such a link).
4. In fitness proportionate selection $p(H, t) = m(H, t)f(H, t)/(M\bar{f}(t))$, where $m(H, t)$ is the number of trees in the population belonging to schema H at time t , $f(H, t)$ is their mean fitness, and $\bar{f}(t)$ is the mean fitness of the trees in the population.
5. The intersection between a set representing a hyperschema and a set representing a program shape produces a set which is either empty or can be represented by a GP schema.
6. Here we imagine the common region as a tree.
7. Note that $h_1 \in G_j \wedge h_2 \in G_k \Rightarrow C(h_1, h_2) = C(G_j, G_k)$.
8. In this paper we have chosen to use the symbol \mathcal{F} to represent both the selection scheme of a GA and the function set used in GP, since this is the standard notation for both. This produces no ambiguity since the selection scheme is not used outside this section, and the function set is not referred to inside it.
9. The operators σ_j can also be interpreted as permutation matrices.
10. Subscripts will be dropped whenever it is possible to infer the arity of a primitive from the context.
11. The space of 0/1 trees obtained when $\mathcal{F} = \mathcal{F}_1$ is isomorphic to the space of binary strings of arbitrary length.
12. For example, if $\ell = 3$, $i_m = 3$, G is $(= (= =))$ and $y = (1\ 1\ (1\ 1\ 1))$, then

$$a = (1\ (1\ 0\ 0\ 0)\ (1\ 1\ 1\ 0)\ (0\ 0\ 0\ 0)).$$

13. Some attempts at this type of theoretical unification have been previously done. For example, in his work on Context-Free Grammar GP (CFG-GP) Whigham [45, pages 125–128] demonstrated that using an appropriate grammar CFG-GP could behave (almost) exactly like a fixed-length GA with one-point crossover, and that, then, his approximate schema theorem for CFG-GP would become (almost) exactly like Holland's schema theorem, the slight differences being only the result of the different mutation strategies used in GAs and CFG-GP. Poli and Langdon [18] produced an approximate schema theorem for GP with one-point crossover that was a proper generalisation of Holland's schema theorem and was applicable also to fixed-length GAs and variable-length GAs. In both cases, however, the theoretical unification between the GP world and the GA world was only at the level of approximate models.
14. Like in GAs, ergodicity will probably be achieved only if an appropriate form of mutation is present.
15. This recombination distribution allows choosing a crossover point before the first allele. This corresponds to the standard practice in GP, where one is allowed to select the root node as a crossover point. It is a simple matter to obtain an equation for the case where we don't allow this, following the much more common convention used in the GA literature.
16. Like for the original Vose's model, the size of the model equations becomes unmanageably large very quickly as the size of the search space grows, so much so that numerical integration of the model equations is only possible for the tiniest of examples.
17. Since these matrices are indexed by variable length binary strings instead of natural numbers, we have indicated the indices (0, 1, 00, 01, 10 and 11) along the top and left-hand side of each matrix. In M_0 , for example, the value in position (1, 0) is 1 and (01, 0) is $1/2$.

References

1. L. Altenberg. Emergent phenomena in genetic programming. In A. V. Sebald and L. J. Fogel, editors, *Evolutionary Programming — Proceedings of the Third Annual Conference*, pages 233–241, San Diego, CA, USA, 24–26 Feb. 1994. World Scientific Publishing.

2. L. Altenberg. The Schema Theorem and Price's Theorem. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 23–49, 1995. Morgan Kaufmann, San Francisco, CA, USA.
3. L. B. Booker. Recombination distributions for genetic algorithms. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 29–44, 1993. Morgan Kaufmann, San Francisco, CA.
4. T. E. Davis and J. C. Principe. A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, 1(3):269–288, 1993.
5. H. Geiringer. On the probability theory of linkage in Mendelian heredity. *Annals of Mathematical Statistics*, 15(1):25–57, March 1944.
6. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.
7. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.
8. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
9. W. B. Langdon. Size fair and homologous tree genetic programming crossovers. *Genetic Programming and Evolvable Machines*, 1(1/2):95–119, Apr. 2000.
10. W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.
11. N. F. McPhee, R. Poli, and J. E. Rowe. A schema theory analysis of mutation size biases in genetic programming with linear representations. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1078–1085, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27–30 May 2001. IEEE Press.
12. M. Mitchell. *An introduction to genetic algorithms*. Cambridge MA: MIT Press, 1996.
13. D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.
14. A. E. Nix and M. D. Vose. Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.
15. U.-M. O'Reilly and F. Oppacher. The troubling aspects of a building block hypothesis for genetic programming. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 73–88, 1995. Morgan Kaufmann, San Francisco, CA.
16. R. Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines*, 2(2):123–163, June 2001.
17. R. Poli and W. B. Langdon. On the search properties of different crossover operators in genetic programming. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 293–301, University of Wisconsin, Madison, Wisconsin, USA, 22–25 July 1998. Morgan Kaufmann.
18. R. Poli and W. B. Langdon. Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6(3):231–252, 1998.
19. R. Poli and N. F. McPhee. Exact GP schema theory for headless chicken crossover and subtree mutation. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1062–1069, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27–30 May 2001. IEEE Press.
20. R. Poli and N. F. McPhee. Exact schema theory for GP and variable-length GAs with homologous crossover. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 104–111, San Francisco, California, USA, 7–11 July 2001. Morgan Kaufmann.
21. R. Poli and N. F. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part I. *Evolutionary Computation*, 11(1):53–66, 2003.
22. R. Poli and N. F. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part II. *Evolutionary Computation*, 11(2), 2003.
23. R. Poli, J. E. Rowe, and N. F. McPhee. Markov chain models for GP and variable-length GAs with homologous crossover. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 112–119, San Francisco, California, USA, 7–11 July 2001. Morgan Kaufmann.
24. R. Poli, C. R. Stephens, A. H. Wright, and J. E. Rowe. On the search biases of homologous crossover in linear genetic programming and variable-length genetic algorithms. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A.

- Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 868–876, New York, 9–13 July 2002. Morgan Kaufmann Publishers.
25. R. Poli, C. R. Stephens, A. H. Wright, and J. E. Rowe. A schema-theory-based extension of Geiringer’s theorem for linear GP and variable-length GAs under homologous crossover. In K. D. Jong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithm 7*, pages 45–62, 2003. Morgan Kaufmann, San Francisco, CA.
 26. A. Prügel-Bennett and J. L. Shapiro. An analysis of genetic algorithms using statistical mechanics. *Physical Review Letters*, 72:1305–1309, 1994.
 27. N. J. Radcliffe. Schema processing. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.5–1–10. Oxford University Press, 1997.
 28. C. Reeves and J. Rowe. *Genetic algorithms: principles and perspectives*. Kluwer Academic Press, 2003.
 29. J. P. Rosca. Analysis of complexity drift in genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 286–294, Stanford University, CA, USA, 13–16 July 1997. Morgan Kaufmann.
 30. J. E. Rowe. Population fixed-points for functions of unitation. In W. Banzhaf and C. Reeves, editors, *Foundations of Genetic Algorithms 5*, pages 69–84, 1999. Morgan Kaufmann, San Francisco, CA.
 31. J. E. Rowe, M. D. Vose, and A. H. Wright. Group properties of crossover and mutation. *Evolutionary Computation*, 10(2):151–184, 2002.
 32. G. Rudolph. Convergence analysis of canonical genetic algorithm. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.
 33. G. Rudolph. Genetic algorithms. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.4–20–27. Oxford University Press, 1997.
 34. G. Rudolph. Models of stochastic convergence. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.3–1–3. Oxford University Press, 1997.
 35. G. Rudolph. Stochastic processes. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.2–1–8. Oxford University Press, 1997.
 36. W. M. Spears. Aggregating models of evolutionary algorithms. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzal, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 631–638, Mayflower Hotel, Washington D.C., USA, 6–9 July 1999. IEEE Press.
 37. W. M. Spears. The equilibrium and transient behaviour of mutation and recombination. In W. M. Spears and W. Martin, editors, *Foundations of Genetic Algorithms Workshop 6*, pages 241–260, 2001. Morgan Kaufmann, San Francisco, CA.
 38. M. R. Spiegel. *Probability and Statistics*. McGraw-Hill, New York, 1975.
 39. C. R. Stephens. Some exact results from a coarse grained formulation of genetic dynamics. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshek, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 631–638, San Francisco, California, USA, 7–11 July 2001. Morgan Kaufmann.
 40. C. R. Stephens and H. Waelbroeck. Effective degrees of freedom in genetic algorithms and the block hypothesis. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, pages 34–40, East Lansing, 1997. Morgan Kaufmann.
 41. C. R. Stephens and H. Waelbroeck. Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124, 1999.
 42. M. D. Vose. *The simple genetic algorithm: Foundations and theory*. MIT Press, Cambridge, MA, 1999.
 43. M. D. Vose and G. E. Liepins. Punctuated equilibria in genetic search. *Complex Systems*, 5(1):31–44, 1991.
 44. P. A. Whigham. A schema theorem for context-free grammars. In *1995 IEEE Conference on Evolutionary Computation*, volume 1, pages 178–181, Perth, Australia, 29 Nov. - 1 Dec. 1995. IEEE Press.
 45. P. A. Whigham. *Grammatical Bias for Evolutionary Learning*. PhD thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 14 Oct. 1996.
 46. D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Department of Computer Science, Colorado State University, Aug. 1993.