, , 1–42 ()

Exact Schema Theory for Genetic Programming and Variable-length Genetic Algorithms with One-Point Crossover

RICCARDO POLI r.poli@cs.bham.ac.uk
School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, UK

Communicated by:

Abstract. A few schema theorems for Genetic Programming (GP) have been proposed in the literature in the last few years. Since they consider schema survival and disruption only, they can only provide a *lower bound* for the expected value of the number of instances of a given schema at the next generation rather than an exact value. This paper presents theoretical results for GP with one-point crossover which overcome this problem. Firstly, we give an exact formulation for the expected number of instances of a schema at the next generation in terms of *microscopic* quantities. Thanks to this formulation we are then able to provide an improved version of an earlier GP schema theorem in which some (but not all) schema creation events are accounted for. Then, we extend this result to obtain an exact formulation in terms of *macroscopic* quantities which makes all the mechanisms of schema creation explicit. This theorem allows the exact formulation of the notion of effective fitness in GP and opens the way to future work on GP convergence, population sizing, operator biases, and bloat, to mention only some of the possibilities.

1. Introduction

Schema theorems are traditionally used to explain how Genetic Algorithms (GAs) and, more recently, Genetic Programming (GP) work [13, 11, 44, 33, 36, 35]. Typically, schema theorems state something about the properties of a population at the next generation, such as the expected proportion of individuals in a given schema (a subset of the search space), in terms of quantities measured at the current generation. These quantities are normally averages over the entire population or subsets of it, like schema fitnesses, population fitness, number of individuals in a schema, etc..

The usefulness of schemata and schema theorems have been widely criticised (see for example [5, 2, 8, 9]). While some criticisms are really not justified, as discussed in [43, 30, 14], others are reasonable and apply to many schema theories. Perhaps the main such criticism is that schema theorems provide only *lower bounds* for the expected value of the number of instances of a schema in the next generation. This makes it difficult to use schema theories to predict the future behaviour of a GA even for a single generation ahead (although it is possible to use them to find bounds for quantities like schema variance and signal-to-noise ratio [37]).

Because of correct and incorrect interpretations of the weaknesses of schema theories, many researchers nowadays incorrectly believe that schema theorems are nothing more than trivial tautologies of no use whatsoever (see for example [57, preface] and [3, Page xxxiv]).

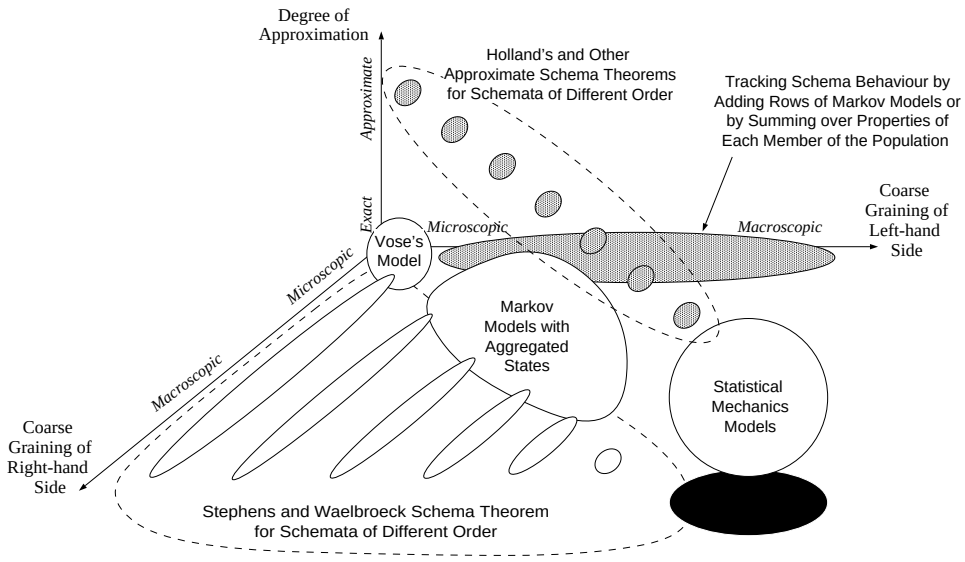

Figure 1. Space of GA/GP models.

Schema theorems are by no means the only models which can predict the behaviour of a GA in the next generation on the basis of what is known on the current generation. Indeed, in the last decade other approaches have become quite popular. We will review them below.

For the purposes of this paper it is useful to distinguish between the different models of GAs on the basis of three properties: 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). Figure 1 shows the space of possible models with a reference system defined by the three quantities mentioned above. In the following we will see where different GA/GP models can be placed in this reference system. In the figure and in the rest of the paper the term *microscopic* refers to properties of single strings/programs while the term *macroscopic* refers to properties (such as average fitness, cardinality, etc.) of larger sets of individuals.

Near the origin of the reference system are Vose's and other Markov chain models [21, 6, 46, 49, 47, 48, 57]. These are exact, *fully microscopic* models of the expected behaviour of a GA. These tend to produce equations with enormous numbers of degrees of freedom. To the other extreme of the spectrum are the statistical mechanics models of Prügel-Bennett and Shapiro [42] which are simpler, approximate, *fully macroscopic* models. These are represented by the large sphere in Figure 1. Between these two extremes other kinds of models can be placed in which both the predicted and the predicting quantities present the same level of coarse graining. This is, for example, one of the features of the approximate schema

theorems proposed in the past including Holland's [13, 11, 44, 33, 36, 35]. In Figure 1, these schema theorems have been represented as a collection of small blobs aligned on a straight line. This is to indicate that with these schema theorems one can specialise the schema equations to schemata of different order, and, therefore, one can vary the level of coarse graining.

Other examples of models in which both the predicted and the predicting quantities present the same level of coarse graining are exact [45] and approximate [50] Markov models in which some of the states of the system have been aggregated. These are represented by the large blob located half-way between Vose's and the statistical mechanics spheres in Figure 1.

The use of macroscopic quantities in the r.h.s. of GA models tends to lead to simpler equations which are relatively easy to study and understand. Sometimes this can only be done at a price: introducing approximations like in Holland's theorem, in the approximate Markov models with aggregate states and in the statistical mechanics models. Clearly, these approximations limit the predictions one can safely make using these models. So, the main weakness attributed to the schema theorems presented in the past, the inability to make accurate predictions over multiple generations, is in fact present also in more modern and fashionable groups of models, and it is simply the price to pay for simplicity.

However, it is not necessarily always the case that schema models based on macroscopic quantities are approximate as shown by the elongated ellipsoids within the triangular dashed spline in the lower part of Figure 1. These represent a schema theorem for binary GAs recently developed by Stephens and Waelbroeck's [54, 55] which gives an exact formulation (rather than a lower bound) for the expected number of instances of a schema in the next generation in terms of macroscopic quantities. Like with older schema theorems, one can specialise this schema theorem to schemata of different order, thus obtaining models of different level of coarse graining. An additional characteristic of this theorem is that the quantities on the r.h.s. present levels of coarse graining equal to or higher than that of the quantity on the l.h.s. (this is why one of the extremes of the ellipsoids representing such a theorem are aligned along the line bisecting the horizontal plane).

Other exact schema-based models are also possible. For example, it is possible to develop approximate schema models which share the features of Stephens and Waelbroeck's exact schema theorem but are approximate. For clarity, these are omitted from Figure 1. Also, it is possible to add up sets of equations of Vose's model to explicitly predict the number of instances of a given schema in the next generation using an approach such as the one in [7]. This is illustrated by the ellipsoid at the back of Figure 1. The same ellipsoid also represent the case in which macroscopic properties are predicted by adding up the contributions given by each possible member or each possible pair of members of the population, like in [2].

In the case of GP theory the space of models is significantly less densely populated than the space of GA models. Before the present work only approximate schema theorems [15, 24, 58, 33, 44], which we will review in the next section, and one exact

microscopic model presented in [1] (see Section 2.2) were available. These models are represented by the gray blobs in Figure 1.

This paper presents the first theoretical results on GP schemata which provide an exact macroscopic formulation (rather than a lower bound) for the expected number of instances of a schema in the next generation for GP with one-point crossover.[1] We achieve this because our results extend to GP the GA theory of Stephens and Waelbroeck, and make the effects and the mechanisms of schema creation explicit, unlike previous work which concentrated on schema survival and disruption.

The paper is organised as follows. Firstly, we provide a review of earlier relevant work on GP and GA schemata in Section 2. Then, in Section 3.1, we generalise the GA theory in [54, 55] to the case of trees of fixed size and shape. Then, we introduce the notion of hyperschema (Section 3.2) and we formulate an *exact microscopic* schema theorem (Section 3.3). Thanks to this formulation we are then able to provide a *macroscopic* GP schema theorem in which some (but not all) schema creation events are accounted for (Section 3.4), thus improving on the result presented in [33, 36].

One of the problems with the exact microscopic account mentioned above is that the r.h.s. is a function of microscopic quantities (i.e. properties of the individuals in the population, like their selection probability) rather than macroscopic quantities (i.e. properties of schemata, like their fitness or number of instances). For this reason, in Section 3.5 we extend the microscopic schema theorem to obtain an *exact* formulation in terms of *macroscopic* quantities. Using this we provide the exact formulation of the notion of effective fitness in GP (firstly introduced in GP in [22, 23] in approximate form) in Section 3.6. Then, Section 4 gives several detailed examples on how to use the theory in practice.

In the macroscopic schema theorem, the r.h.s. includes the selection probabilities of the schema itself and of a set of lower-order schemata which one-point crossover uses to build instances of the schema. As discussed in Section 5.1, this result supports the existence of building blocks in GP which, however, are not necessarily all short, low-order or highly fit. In Section 5.2, we discuss how schema theorems might be used to understand some phenomena in GP search which are not fully understood, such as bloat and code compression. Then (Section 5.3), we discuss the impact our schema theory can have on the study of the biases and the effects of crossover, and indicate how this may soon allow a formal mathematical comparison of different operators. We explain why in the near future we may expect new results on convergence and population sizing in GP based on schema theories in Section 5.4. Finally, some conclusions are drawn in Section 6.

2. Background

2.1. Holland's GA Schema Theory

Schemata are sets of points in the search space sharing some syntactic feature. Schema theorems are descriptions of how the number (or the fraction) of members of the population belonging to a schema vary over time.

In the context of GAs operating on binary strings, syntactically a schema (or similarity template) is a string of symbols taken from the alphabet {0,1,*}. The character * is interpreted as a "don't care" symbol, so that, semantically, a schema represents a set of bit strings. For example the schema *10*1 represents four strings: 01001, 01011, 11001 and 11011. The number of non-* symbols is called the *order* $\mathcal{O}(H)$ of a schema H. The distance between the furthest two non-* symbols is called the *defining length* $\mathcal{L}(H)$ of the schema. Holland obtained a result (often referred to as "the schema theorem") which predicts how the number of strings in a population matching (or belonging to) a schema is expected to vary from one generation to the next [13]. The theorem can be reformulated as follows:

$$E[m(H,t+1)] \geq Mp(H,t) \cdot (1-p_m)^{\mathcal{O}(H)} \cdot \left[1 - p_{xo}\frac{\mathcal{L}(H)}{N}\left(1-p(H,t)\right)\right] \quad (1)$$

where p_m is the probability of mutation per bit, p_{xo} is the probability of crossover, N is the number of bits in the strings, M is the number of strings in the population, $E[m(H,t+1)]$ is the expected number of strings matching the schema H at generation $t+1$, and $p(H,t)$ is the probability of selection of the schema H.[2] In fitness proportionate selection this is given by $p(H,t) = \frac{m(H,t)f(H,t)}{M\bar{f}(t)}$ where $m(H,t)$ is the number of strings matching the schema H at generation t, $f(H,t)$ is the mean fitness of the strings matching H, and $\bar{f}(t)$ is the mean fitness of the strings in the population.

2.2. GP Schema Theories

One of the difficulties in obtaining theoretical results on GP using the idea of schema is that the definition of schema is much less straightforward than for GAs and a few alternative definitions have been proposed in the literature. Syntactically all of them define schemata as composed of one or multiple trees or fragments of trees. In some definitions [15, 1, 24, 58, 59] schema components are *non-rooted* and a schema is seen as a set of components (subtrees) that can be present multiple times within the same program. The focus in these theories is to predict how the number or the frequency of such components vary over time. However, the variability of the size and shape of the programs matching the same schema and the fact that each component can be present a variable number of times in each program lead to considerable complications in the calculations necessary to formulate schema theorems for GP. For this reason, with the exception of one of the results in [1],[3] all past schema-as-component theorems provided only lower bounds.

In more recent definitions [33, 44] syntactically schemata are represented by *rooted* trees or tree fragments. These definitions make schema theorem calculations easier. We describe them below.

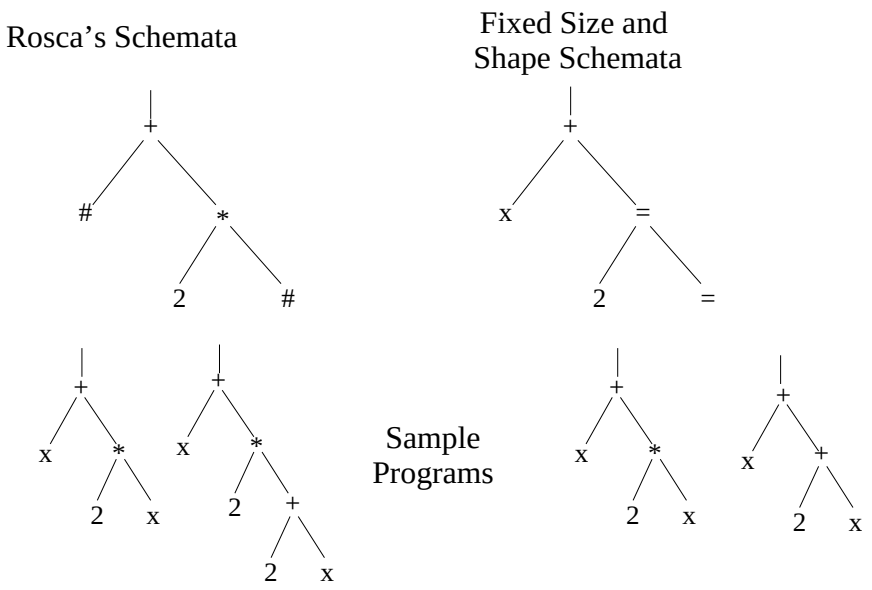

Figure 2. Examples of rooted schemata (top) and some instances of programs sampling them (bottom).

2.3. Rosca's GP Schema Theory for Standard Crossover

Rosca [44] proposed a definition of schema, called *rooted tree-schema*, in which, syntactically, a schema is a rooted contiguous tree fragment. If $\mathcal{F}$ and $\mathcal{T}$ are the function and the terminal sets used in a GP run, a tree fragment is a tree built using the function set $\mathcal{F}$ and the terminal set $\mathcal{T} \cup \{\#\}$. The primitive `#` is a "don't care" symbol which stands for any valid subtree. So, semantically a schema is the set of programs obtained by replacing the "don't care" symbols in it with valid subtrees in all possible ways. For example, the rooted tree-schema `(+ # x)` represents all the programs whose root node is a `+` the second argument of which is `x`. Another example of schema with some of its instances is shown in Figure 2 (left). With this definition, schemata divide the space of programs into subspaces containing programs of different sizes and shapes.

Rosca derived a schema theorem for GP with standard crossover which provided a lower bound for the expected number of instances of a schema in the next generation as a function of the schema order, and the fitness and size of its instances in the population.

2.4. GP Fixed-Size-and-Shape Schema Theory for One-point Crossover

In [33] a different definition of schema for GP was proposed in which a *schema* is a tree composed of functions from the set $\mathcal{F} \cup \{=\}$ and terminals from the set $\mathcal{T} \cup \{=\}$.

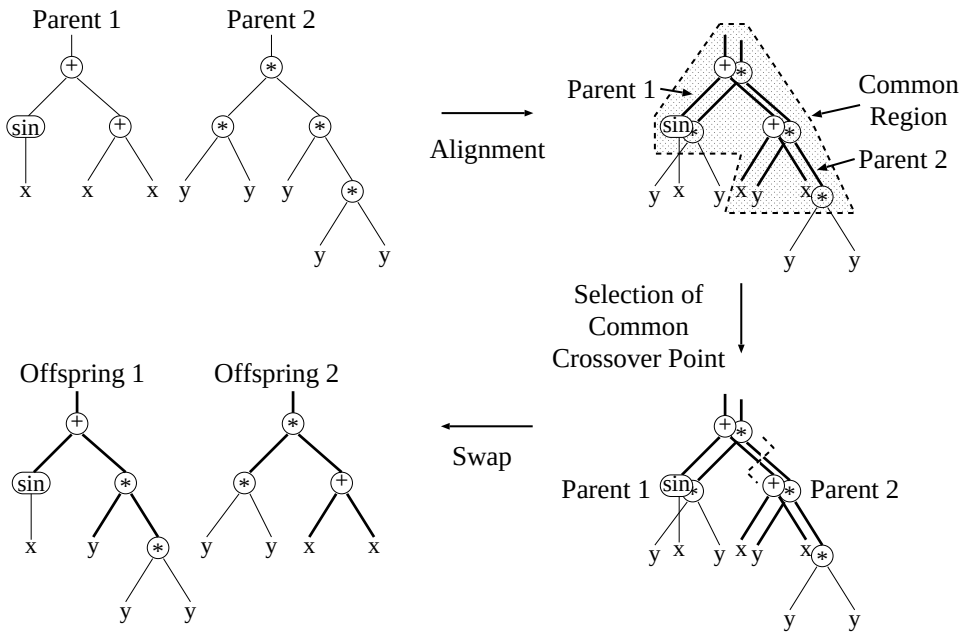

Figure 3. GP one-point crossover. The thick lines are links that can be selected as common crossover points.

The symbol = is a "don't care" symbol which stands for a *single* terminal or function. In line with the original definition of schema for GAs, a schema H represents programs having the same shape as H and the same labels for the non-= nodes. For example, if $\mathcal{F}$={`+, *`} and $\mathcal{T}$={`x, 2`} the schema `(+ x (= 2 =))` would represent the four programs `(+ x (+ 2 x))`, `(+ x (+ 2 2))`, `(+ x (* 2 x))` and `(+ x (* 2 2))`. This schema and some of its instances are shown in Figure 2 (right).[4] This definition of schema partitions the program space into subspaces of programs of fixed size and shape. For this reason in the following we will refer to these schemata as *fixed-size-and-shape schemata*.[5]

In order to derive a GP schema theorem for these schemata non-standard forms of mutation and crossover, namely point mutation and one-point crossover, were used. *Point mutation* is the substitution of a node in the tree with another node with the same arity. *One-point crossover* [33, 36] works by selecting a common crossover point in the parent programs and then swapping the corresponding subtrees, like standard crossover. To account for the possible structural diversity of the two parents, one-point crossover analyses the two trees from the root nodes and considers for the selection of the crossover point only the parts of the two trees (common region) which have the same topology (i.e. the same arity in the nodes encountered traversing the trees from the root node) as illustrated in Figure 3.

The resulting schema theorem is the following:

$$E[m(H,t+1)] \geq Mp(H,t)(1-p_m)^{\mathcal{O}(H)}. \qquad (2)$$
$$\left\{1 - p_{xo}\left[p_{\text{diff}}(t)\,(1-p(G(H),t)) + \frac{\mathcal{L}(H)}{N(H)}(p(G(H),t) - p(H,t))\right]\right\}$$

where:

- p_m is the mutation probability (per node),
- $\mathcal{O}(H)$ is the number of non-= symbols in the schema H which we term the *order of* H,
- $G(H)$ is the zero-th order schema with the same structure of H where all the defining nodes in H have been replaced with "don't care" symbols (so $G(H)$ represents all programs with the same shape and size as H),
- $N(H)$ (the *length* of the schema) is the total number of nodes in the schema,
- the *defining length* $\mathcal{L}(H)$ of the schema is the number of links in the minimum tree fragment including all the non-= symbols within the schema,[6]
- $p_{\text{diff}}(t)$ is the conditional probability that H is disrupted by crossover when the second parent has a different shape (i.e. does not sample $G(H)$),

and the other symbols have the same meaning as in Equation 1 (see [33, 36] for the proof).

The probability $p_{\text{diff}}(t)$ is hard to model mathematically. Its expected variations during a run were analysed in detail in [33], but in the absence of precise information one should assume the worst case scenario $p_{\text{diff}}(t) = 1$.

Equation 2 is a generalisation of Equation 1 which is a version of Holland's schema theorem. This can easily be checked by specialising Equation 2 to the case in which all the programs in the population have the same size and shape as the schema H, whereby $p(G(H),t) = 1$. In this case Equation 2 takes the same form as Equation 1. If, in addition, one assumes that all programs are linear structures (i.e. that the function set includes only unary functions) and that only two functions and two terminals are used, then the concept of GP schema corresponds exactly to the concept of a GA schema and GP one-point crossover corresponds exactly to GA one-point crossover (see Section 4.1 for an example). In these conditions the GP schema theorem reported above becomes the GA schema theorem in Equation 1.

In our work, a link between GP theory and GA theory was first emphasised in [36] where we used the slogan "$\lim_{t\to\infty} \text{GP}_{1\text{pt}}(t) = \text{GA}$" to summarise the asymptotic behaviour of GP with one-point crossover. However, an earlier, more formal link was described in Whigham's PhD thesis [59, Sections 6.7 and 6.8] where the concept of schema for his context-free grammar GP and the related approximate schema theorem were shown to be applicable both to fixed-length binary GAs under one-point crossover and to GP under standard crossover (thanks to the use of appropriate grammars).

2.5. Exact GA Schema Theory

As we noted in [37] the selection/crossover/mutation process can be seen as a Bernoulli trial (a newly created individual either samples or does not sample a schema H) and, therefore, the number of instances of H in the next generation, $m(H, t+1)$, is a binomial stochastic variable. So, if we denote with $\alpha(H, t)$ the success probability of each trial (i.e. the probability that a newly created individual samples H), which we term the *total transmission probability* of H, an exact schema theorem is simply

$$E[m(H, t+1)] = M\alpha(H, t). \tag{3}$$

In a binary GA in the absence of mutation the total transmission probability is given by the following equation (which can be obtained by simplifying the results in [54, 55] or, perhaps more simply, as described in the following paragraphs):

$$\alpha(H, t) = (1 - p_{xo})p(H, t) + \frac{p_{xo}}{N}\sum_{i=0}^{N-1} p(L(H, i), t)p(R(H, i), t), \tag{4}$$

where p_{xo} is the crossover probability, $p(K, t)$ is the selection probability of a generic schema K at generation t, $L(H, i)$ is the schema obtained by replacing all the elements of H from position $i+1$ to position N with "don't care" symbols (*), $R(H, i)$ is the schema obtained by replacing all the elements of H from position 1 to position i with "don't care" symbols, and i varies over the valid crossover points.[7] For example, if H =**1111, then $L(H, 1)$ =******, $R(H, 1)$ =**1111, $L(H, 3)$ =**1*** and $R(H, 3)$ =***111. If one, for example, wanted to calculate the total transmission probability of the schema *11, the previous equation would give:

$$\begin{aligned}\alpha(*11, t) &= (1 - p_{xo})p(*11, t) + \\ &\quad \frac{p_{xo}}{3}\Big(p(***, t)p(*11, t) + p(***, t)p(*11, t) + p(*1*, t)p(**1, t)\Big) \\ &= \Big(1 - \frac{2}{3}p_{xo}\Big)p(*11, t) + \frac{p_{xo}}{3}p(*1*, t)p(**1, t),\end{aligned}$$

since $p(***, t) = 1$.

It should be noted that Equation 4 looks considerably different from the results in [54, 55]. This is because it is expressed using our own notation and it is applicable only when mutation is absent. Also, we allow N, rather than $N-1$, crossover points in a string of length N. However, by substituting Equation 4 into Equation 3 and performing some minor additional calculations it is easy to prove that this is basically a crossover-only version of the GA schema theorem described in [54, 55].[8]

Stephens and Waelbroeck derived their schema theorem by properly reformulating their evolution equations for strings. However, here we want to provide a simpler direct proof of Equation 4. In our approach we assume that while producing each individual for a new generation one flips a biased coin to decide whether to apply selection followed by cloning (probability $1 - p_{xo}$) or selection followed by crossover

(probability p_{xo}). If selection plus cloning is applied, then there is a probability $p(H,t)$ that the new individual created samples H (hence the first term in Equation 4). If instead selection followed by crossover is selected, we use the unusual idea of first choosing the crossover point and then the parents. When selecting the crossover point, one has to choose randomly one of the N crossover points, each of which has a probability $1/N$ of being selected. Once this decision has been made, one has to select two parents. Then crossover is executed. This will result in an individual that samples H only if the first parent has the correct left-hand side (with respect to the crossover point) *and* the second parent has the correct right-hand side. These two events are independent because each parent is selected with an independent Bernoulli trial. So, the probability of the joint event is the product of the probabilities of the two events. Assuming that crossover point i has been selected, the first parent has the correct left-hand side if it belongs to $L(H,i)$ while the second parent has the correct right-hand side if it belongs to $R(H,i)$. The probabilities of these events are $p(L(H,i),t)$ and $p(R(H,i),t)$, respectively (whereby the terms in the summation in Equation 4, the summation being there because there are N possible crossover points). Combining the probabilities of all these events one obtains Equation 4.

Stephens and Waelbroeck [54, 55] used this result as a starting point for a number of other important results on the behaviour of a GA over multiple generations on the assumption of infinite populations. It should be noted that their theorem is a refinement of Holland's schema theorem. Indeed, assuming mutation is absent, Equations 3 and 4 can be transformed into Equation 1 as shown in the following derivation where $B(H)$ is the set of crossover points between the furthest defining bits in H:

$$
\begin{aligned}
&\frac{E[m(H,t+1)]}{M} \\
&= \alpha(H,t) \\
&= (1-p_{xo})p(H,t) + \frac{p_{xo}}{N}\sum_{i=0}^{N-1} p(L(H,i),t)p(R(H,i),t) \\
&= (1-p_{xo})p(H,t) + \\
&\quad \frac{p_{xo}}{N}\left(\sum_{i\in B(H)} p(L(H,i),t)p(R(H,i),t) + (N-\mathcal{L}(H))p(H,t)p(**\ldots*,t)\right) \\
&= p(H,t)\left(1-p_{xo}\frac{\mathcal{L}(H)}{N}\right) + \frac{p_{xo}}{N}\sum_{i\in B(H)} p(L(H,i),t)p(R(H,i),t) \qquad (5) \\
&\geq p(H,t)\left(1-p_{xo}\frac{\mathcal{L}(H)}{N}\right) + \frac{p_{xo}}{N}\sum_{i\in B(H)} (p(H,t))^2 \\
&= p(H,t)\left(1-p_{xo}\frac{\mathcal{L}(H)}{N}\right) + p_{xo}\frac{\mathcal{L}(H)}{N}(p(H,t))^2 \\
&= p(H,t)\cdot\left[1-p_{xo}\frac{\mathcal{L}(H)}{N}\left(1-p(H,t)\right)\right],
\end{aligned}
$$

where we used the following facts: $p(**\ldots*,t) = 1$, $p(L(H,i),t) \geq p(H,t)$ and $p(R(H,i),t) \geq p(H,t)$. These calculations can be used to evaluate in which circumstances Holland's schema theorem can be considered a good approximation to an exact schema theorem. Since the difference between the r.h.s. of the exact schema theorem and the r.h.s. of Holland's schema theorem is

$$\frac{p_{xo}}{N} \sum_{i \in B(H)} \Big(p(L(H,i),t) p(R(H,i),t) - (p(H,t))^2 \Big),$$

it is clear that, everything else being equal,[9] the approximation is better as the number of terms in the sum, $|B(H)| = \mathcal{L}(H)$, decreases.

2.6. Effective Fitness

The concept of effective fitness was introduced in GP in [22, 23] to explain the reasons for bloat and active-code compression. In that work the *effective fitness* of program j was defined as follows:

$$f_j^e = f_j \left(1 - p_c \frac{C_j^e}{C_j^a} p_j^d \right), \tag{6}$$

assuming fitness proportionate selection. In this equation C_j^a is the number of nodes in program j, C_j^e is the number of nodes in the active part (in contrast to the intron part) of program j, p_c is the crossover probability, p_j^d is the probability that crossover in an active block of program j leads to worse fitness for the offspring of j and f_j is the fitness of individual j. Then, letting P_j^t be the proportion of programs j at generation t, P_j^{t+1} the average proportion of offspring of j which behave like j at generation $t+1$, and $\bar{f}^t$ the average population fitness at generation t, one can write:

$$P_j^{t+1} \approx P_j^t \frac{f_j^e}{\bar{f}^t} \tag{7}$$

which describes "the proliferation of individuals from one generation to the next" [23]. The "$\approx$" sign in the equation should really be "$\geq$" but it was used with the justification that the reconstruction of individuals with the same behaviour as j (due to crossover applied to individuals different from j) was a rare event. Equation 7 clearly indicates that an alternative way of interpreting the effects of crossover is to imagine a GP system in which selection only is used, but in which each individual is given a fitness f_j^e rather than the original fitness f_j.

The concept of effective fitness is very similar to the concept of *operator-adjusted fitness* (not to be confused with Koza's adjusted fitness [15]) introduced for GAs a few years earlier by Goldberg in [10, page 155]. Goldberg defined the adjusted fitness of a schema as

$$f_{\text{adj}}(H,t) = f(H,t) \Big(1 - p_{xo} \frac{\mathcal{L}(H)}{N-1} - p_m \mathcal{O}(H) \Big) \tag{8}$$

which, in the case of fitness proportionate selection and assuming that only $N-1$ crossover points are available, allowed him to reformulate Holland's schema theorem as

$$E[m(H,t+1)] \geq \frac{m(H,t)}{\bar{f}(t)} f_{\text{adj}}(H,t). \tag{9}$$

This clearly indicates how Nordin and Banzhaf's notion of effective fitness and Goldberg's notion of operator-adjusted fitness are essentially the same idea, although specialised for different representations and operators. An interesting way of interpreting the operator-adjusted fitness is that a GA could be thought to be attracted towards the optima of f_{adj} rather than those of the fitness function f. If, for a particular problem, these optima do not coincide, then the problem can be said to be *deceptive* [10]. Like, Nordin and Banzhaf's effective fitness, the definitions of adjusted fitness and deception are only approximations since they are based on Holland's schema theorem. So, Equation 8 can only provide a lower bound for the true effective/adjusted fitness of a schema in a binary GA.

Stephens and Waelbroeck [54, 55] independently rediscovered the notion of effective fitness. Using the notation of Section 2.5, the *effective fitness of a schema* is implicitly defined through the equation

$$E\left[\frac{m(H,t+1)}{M}\right] = \frac{m(H,t)}{M} \cdot \frac{f_{\text{eff}}(H,t)}{\bar{f}(t)},$$

assuming that fitness proportionate selection is used. This has basically the same form as Equation 7. Indeed the two equations represent nearly the same idea, although in different domains. The equation is also very similar to Equation 9. Since $E\left[\frac{m(H,t+1)}{M}\right] = \alpha(H,t)$ and $\frac{m(H,t)}{M\bar{f}(t)} = \frac{p(H,t)}{f(H,t)}$, one obtains

$$\begin{aligned} f_{\text{eff}}(H,t) &= \frac{\alpha(H,t)}{p(H,t)} f(H,t) \qquad (10) \\ &= f(H,t)\Big[1 - p_{xo}\Big(1 - \sum_{i=0}^{N-1} \frac{p(L(H,i),t)p(R(H,i),t)}{Np(H,t)}\Big)\Big] \end{aligned}$$

where we used the value of $\alpha(H,t)$ in Equation 4. This can be rewritten as

$$f_{\text{eff}}(H,t) = f(H,t)\Big[1 - p_{xo} \sum_{i \in B(H)} \Big(1 - \frac{p(L(H,i),t)p(R(H,i),t)}{p(H,t)}\Big)\Big] \tag{11}$$

with calculations like those in Equation 5.

Equations 10 and 11 are similar to Equations 6 and 8, but there are important differences: f_j^e and $f_{\text{adj}}(H,t)$ are *approximations* (of unknown accuracy, being in fact lower bounds) of the true effective fitness of an *individual* in a standard GP system and of a schema in a GA, respectively, while $f_{\text{eff}}(H,t)$ is the *true* effective fitness for a *schema* in a binary GA. In addition, $f_{\text{eff}}(H,t)$ of a schema can be bigger than $f(H,t)$ if the building blocks for H are abundant and relatively fit. On the contrary

the estimates/bounds given by f_j^e and $f_{\text{adj}}(H,t)$ ignore the constructive effects of crossover and, therefore, are always smaller than f_j and $f(H,t)$, respectively.

In [55] Stephens and Waelbroeck proposed a new approach to defining *deception* based on their schema theorem and the notion of effective fitness just defined. The idea is that each of the terms in the sum in Equation 5 represents a different channel for creating instances of the schema H. The effect of these channels is counteracted by destruction effects. These are proportional to $p(H,t)$. So, if for a particular channel $p(L(H,i),t)p(R(H,i),t) < p(H,t)$, the channel could be said to be a *deceptive channel*. As shown by Equation 11, there is a relation between deceptiveness of creation channels and the effective fitness of a schema: if all channels leading to a schema H are deceptive (at least on average), then $f_{\text{eff}}(H,t) < f(H,t)$ and *vice versa*.

Finally, it is worth mentioning that in recent work [52, 53] the notion of effective fitness is used to propose the concept of effective fitness landscape as a general paradigm to understand population dynamics in GAs.

3. GP Hyperschema Theory

A question that immediately comes to mind is whether it is possible to extend Rosca's or the fixed-size-and-shape GP schema theories described in the previous section to obtain exact schema theorems for GP. This is important because it would make it possible to exploit recent GA results such as the ones reported in [37, 26, 29, 28] in GP, too. As described below we were able to extend the fixed-size-and-shape GP schema theory for one-point crossover, thanks to the introduction of a generalisation of the definition of GP schema: the hyperschema.

In the following sections, we start with a simple extension to GP operating on trees of fixed size and shape and then we introduce the general hyperschema theory.

3.1. Theory for Programs of Fixed Size and Shape

If one had a population of programs all having exactly the same size and shape, it would be possible to express the total transmission probability of a fixed-size-and-shape schema, in the presence of one-point crossover, in exactly the same way as in Equation 4, i.e.

$$\alpha(H,t) \;=\; (1-p_{xo})p(H,t) + \frac{p_{xo}}{N(H)} \sum_{i=0}^{N(H)-1} p(l(H,i),t)p(u(H,i),t), \qquad (12)$$

where: $N(H)$ is the number nodes in the schema H (which is assumed to have the same size and shape as the programs in the population); $u(H,i)$ is the schema obtained by replacing all the nodes in the subtree below crossover point i with = nodes; $l(H,i)$ is the schema obtained by replacing all the nodes *not* in the subtree below crossover point i with = nodes; i varies over the $N(H)$ valid crossover points. The symbol u stands for "upper part of", while l stands for "lower part of". For example, Figure 4 (second to fifth columns) shows how $l(H,1)$, $u(H,1)$, $l(H,3)$,

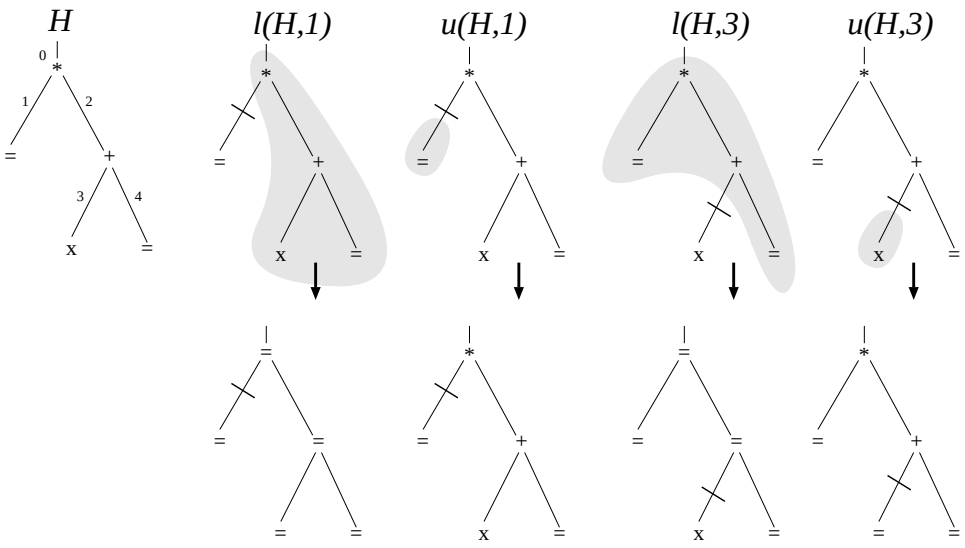

Figure 4. Example of a schema and some of its potential fixed-size-and-shape building blocks.

and $u(H,3)$ are obtained assuming that $H =$`(* = (+ x =))` and that the crossover points are numbered as in the first column.

Formally this result can be obtained by proceeding as in Section 2.5. Again we assume that while producing each individual for a new generation one first decides whether to apply selection followed by cloning (probability $1 - p_{xo}$) or selection followed by crossover (probability p_{xo}). If selection followed by cloning is applied, the new individual created samples H with a probability $p(H,t)$ (hence the first term in Equation 12). If selection followed by crossover is selected, we first choose the crossover point (to be interpreted as a link between nodes) randomly out of the $N(H)$ crossover points available. Then we select two parents and perform crossover. This will result in an individual that samples H only if the first parent has the correct lower part (with respect to the crossover point) *and* the second parent has the correct upper part. Assuming that crossover point i has been selected,[10] the parents recreate H if they belong to $l(H,i)$ and $u(H,i)$, respectively (whereby the terms in the summation in Equation 12). By combining the probabilities of all these events one obtains Equation 12.

It is important to note that, although Equation 12 seems to have exactly the same form as Stephens and Waelbroeck's result for binary GAs in Equation 4, the former is in fact a generalisation of the latter. This is because Equation 12 is applicable to any kind of tree-like structures of fixed size and shape. Only if one considers linear structures (program trees including unary functions only), the two results coincide.[11]

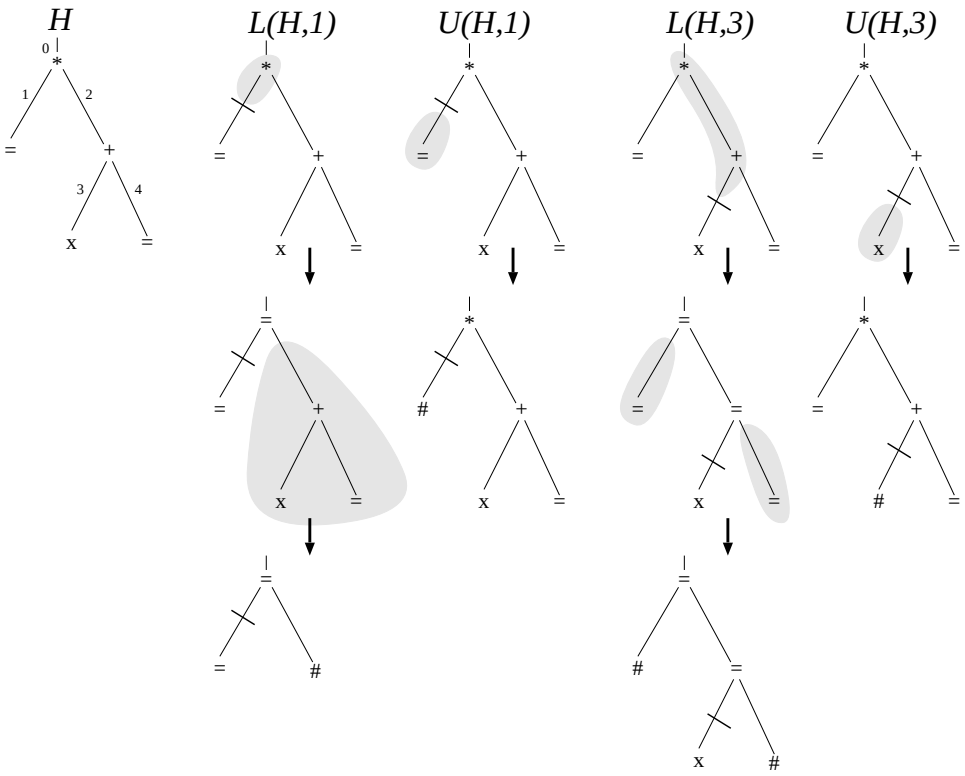

Figure 5. Example of a schema and some of its potential hyperschema building blocks.

3.2. Hyperschemata

In order to generalise the results in the previous section to the case of populations including programs of different sizes and shapes we need to introduce a new, more general definition of schema. The definition is as follows:

Definition 1 (GP hyperschema). A *GP hyperschema* is a rooted tree composed of internal nodes from the set $\mathcal{F} \cup \{=\}$ and leaves from $\mathcal{T} \cup \{=, \#\}$, where $\mathcal{F}$ and $\mathcal{T}$ are the function set and the terminal set used in a GP run. The operator `=` is a "don't care" symbol which stands for exactly one node, while the operator `#` stands for any valid subtree.[12]

An example of a hyperschema is `(* # (+ x =))`. This hyperschema 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 `+`, d) the first argument of the `+` is the variable `x`, e) the second argument of the `+` is any valid node in the terminal set.

This definition presents some of the features of the fixed-size-and-shape schema definition we used for the GP schema theory in [33, 36] and summarised in Section 2.4: hyperschemata are rooted trees, and they include "don't care" symbols which stand for one node only. However, Definition 1 also includes one of the features of the schema definitions proposed by other authors [24, 44]: hyperschemata also include "don't care" symbols which stand for entire subtrees. Indeed, the notion of hyperschema is a generalisation of both Rosca's schemata (which are hyperschemata without = symbols) and fixed-size-and-shape schemata (which are hyperschemata without # symbols). So, a hyperschema can represent a group of schemata in essentially the same way as such schemata represent groups of program trees (hence the name "hyperschema").

3.3. Microscopic Exact GP Schema Theorem

Thanks to hyperschemata, it is possible to obtain the following general result which is valid for populations of programs of any size and shape:

THEOREM 1 (MICROSCOPIC EXACT GP SCHEMA THEOREM) *The total transmission probability for a fixed-size-and-shape GP schema H under one-point crossover and no mutation is*

$$\alpha(H,t) = (1-p_{xo})p(H,t) + p_{xo}\sum_{h_1}\sum_{h_2}\frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(h_1,h_2)}\sum_{i\in C(h_1,h_2)}\delta(h_1\in L(H,i))\delta(h_2\in U(H,i)) \quad (13)$$

where: the first two summations are over all the individuals in the population; $\mathbf{NC}(h_1,h_2)$ is the number of nodes in the tree fragment representing the common region between program h_1 and program h_2; $C(h_1,h_2)$ is the set of indices of the crossover points in the common region; $\delta(x)$ is a function which returns 1 if x is true, 0 otherwise; $U(H,i)$ is the hyperschema obtained by replacing the subtree below crossover point i with a # node; $L(H,i)$ 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. If a crossover point i in the common region between two programs is outside the schema H, then $L(H,i)$ and $U(H,i)$ are empty sets.

The hyperschemata $L(H,i)$ and $U(H,i)$ are generalisations of the schemata $l(H,i)$ and $u(H,i)$ used in Equation 12 (compare Figures 4 and 5). 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.

Before we proceed with the proof of the theorem, let us try to understand with an example how $L(H,i)$ and $U(H,i)$ are built. If H =`(* = (+ x =))`, as indicated in the second column of Figure 5, then $L(H,1)$ is obtained by first replacing the root node with a = symbol and then replacing the subtree connected to the right of root node with a # symbol obtaining `(= = #)`. The schema $U(H,1)$ is instead obtained by replacing the subtree below the crossover point with a # symbol obtaining `(* # (+ x =))`, as illustrated in the third column of Figure 5. The

fourth and fifth columns of Figure 5 show how $L(H,3) =$ `(= # (= x #))` and $U(H,3) =$ `(* = (+ # =))` are obtained.

Once the concepts of $L(H,i)$ and $U(H,i)$ are available, the theorem can easily be proven by: a) considering all the possible ways in which parents can be selected for crossover and in which the crossover points can be selected in such parents, b) computing the probabilities that each of those events happens *and* the first parent has the correct lower part (w.r.t. the crossover point) to create H while the second parent has the correct upper part, and c) adding up such probabilities (whereby the three summations in Equation 13). Obviously, in order for this equation to be valid it is necessary to assume that if a crossover point i is in the common region between two programs but outside the schema H under consideration, then $L(H,i)$ and $U(H,i)$ are empty sets (i.e. they cannot be matched by any individual).

A more formal way to prove the result is the following.

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 point i. Then, let us consider the function

$$g(h_1,h_2,i,H) = \delta(h_1 \in L(H,i))\delta(h_2 \in U(H,i)).$$

Given two parent programs, h_1 and h_2, and a schema of interest H, this function returns the value 1 if crossing over h_1 and h_2 at position 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 points 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:

$$E[\gamma] = \sum_{h_1}\sum_{h_2}\sum_{i\in C(h_1,h_2)} g(h_1,h_2,i,H)p(h_1,h_2,i,t). \tag{14}$$

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

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 point 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 one-point crossover

$$p(i|h_1,h_2) = 1/\mathbf{NC}(h_1,h_2).$$

So,

$$p(h_1,h_2,i,t) = \frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(h_1,h_2)},$$

which substituted into Equation 14 leads to the term due to crossover in the r.h.s. of Equation 13. By multiplying this by p_{xo} (the probability of doing selection followed by crossover) and adding the term $(1-p_{xo})p(H,t)$, which represent the probability of doing selection followed by cloning, one obtains the r.h.s. of Equation 13. ■

Equation 13 allows one to compute the exact total transmission probability of a GP schema in terms of microscopic quantities. Therefore, this theorem populates the space of possible GP models within the ellipsoid at the back of Figure 1. By proceeding as indicated in [28] a series of new results for GP schemata can be obtained which provide important statistical information on schema behaviour (e.g. schema probability distribution, schema variance, schema signal-to-noise ratio, extinction probability, etc.). In this paper we will not study these results. Instead, in the next sections we will concentrate on understanding Equation 13 in greater depth and on transforming it into more macroscopic models.

3.4. Macroscopic Schema Theorem with Schema Creation Correction

If one restricts the first two summations in Equation 13 to include only the individuals having the same shape and size of the schema H, i.e. which belong to $G(H)$, one obtains:

$$\alpha(H,t) \geq (1-p_{xo})p(H,t)+ \\ \frac{p_{xo}}{N(H)} \sum_{i=0}^{N(H)-1} \sum_{h_1 \in G(H)} p(h_1,t)\delta(h_1 \in L(H,i)) \sum_{h_2 \in G(H)} p(h_2,t)\delta(h_2 \in U(H,i))$$

where we used the following facts: a) $\mathbf{NC}(h_1,h_2) = N(H)$ since $h_1, h_2 \in G(H)$, b) all common regions have the same shape and size as H, and so c) $C(h_1,h_2) = \{0,1,\ldots,N(H)-1\}$ (assuming that the crossover points in H are numbered 0 to $N(H)-1$). This equation can easily be transformed into the following

THEOREM 2 (GP SCHEMA THEOREM WITH SCHEMA CREATION CORRECTION) *A lower bound for the total transmission probability for a fixed-size-and-shape GP schema H under one-point crossover and no mutation is*

$$\alpha(H,t) \geq (1-p_{xo})p(H,t) + \\ \frac{p_{xo}}{N(H)} \sum_{i=0}^{N(H)-1} p(L(H,i) \cap G(H),t)p(U(H,i) \cap G(H),t), \tag{15}$$

the equality applying when all the programs in the population sample $G(H)$.

Let us compare this result to the one described in Section 2.4.

It is easy to see that by dividing the right-hand side of Equation 2 by M one obtains a lower bound for $\alpha(H,t)$. If we consider the case in which $p_m = 0$ and we assume the worst case scenario for $p_{\text{diff}}(t)$, i.e. $p_{\text{diff}}(t) = 1$, we obtain[13]

$$\alpha(H,t) \geq p(H,t)\left\{1 - p_{xo}\left[1 - p(G(H),t) + \frac{\mathcal{L}(H)}{N(H)}(p(G(H),t) - p(H,t))\right]\right\}.$$

It is easy to show that the difference between the right-hand side of this equation and the lower bound provided by Equation 15 is:

$$\Delta\alpha(H,t) = \frac{p_{xo}}{N(H)}\left(\sum_{i\in B(H)} p(L(H,i),t)p(U(H,i),t) - \mathcal{L}(H)p(H,t)^2\right), \quad (16)$$

where $B(H)$ is the set of crossover points in the tree fragment used in the definition of $\mathcal{L}(H)$. Since such a tree fragment contains exactly $\mathcal{L}(H)$ crossover points, we can rewrite this equation as

$$\Delta\alpha(H,t) = \frac{p_{xo}}{N(H)}\sum_{i\in B(H)}\Big(p(L(H,i),t)p(U(H,i),t) - p(H,t)^2\Big).$$

Since, for $i \in B(H)$, $L(H,i)$ and $U(H,i)$ are supersets of H, it follows that $p(L(H,i),t) \geq p(H,t)$ and $p(U(H,i),t) \geq p(H,t)$, whereby we see that $\Delta\alpha(H,t) \geq 0$. So, Theorem 2 provides a better estimate of the true transmission probability of a schema on the assumption that $p_{\text{diff}}(t) = 1$. This is why we called Theorem 2 the "GP Schema Theorem with Schema Creation Correction". This theorem provide schema models which share the features of Stephens and Waelbroeck's exact schema theorem but are approximate. Therefore, these models populate the space of GP models in the region above Stephens and Waelbroeck's theorem in Figure 1.

3.5. Macroscopic Exact GP Schema Theorem

In order to transform Equation 13 into an exact macroscopic description of schema propagation, let us start by numbering all the possible program shapes, i.e. all the possible fixed-size-and-shape schemata of order 0. Let us denote such schemata as G_1, G_2, $\cdots$. These schemata represent disjoint sets of programs. Their union represents the whole search space. For these reasons, we can rewrite

$$\sum_j \delta(h_1 \in G_j) = 1.$$

We append the l.h.s. of this expression and of an analogous expression for $\delta(h_2 \in G_k)$ to the terms in the triple summation in Equation 13 and reorder the terms obtaining:

$$\alpha(H,t) = \sum_{h_1}\sum_{h_2}\frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(h_1,h_2)} \sum_{i\in C(h_1,h_2)}\sum_j \delta(h_1 \in L(H,i))\delta(h_1 \in G_j)\sum_k \delta(h_2 \in U(H,i))\delta(h_2 \in G_k)$$

$$
\begin{aligned}
&= \sum_j \sum_k \sum_{h_1} \sum_{h_2} \frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(h_1,h_2)} \\
&\quad \sum_{i \in C(h_1,h_2)} \delta(h_1 \in L(H,i))\delta(h_1 \in G_j)\delta(h_2 \in U(H,i))\delta(h_2 \in G_k) \\
&= \sum_j \sum_k \sum_{h_1 \in G_j} \sum_{h_2 \in G_k} \frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(h_1,h_2)} \\
&\quad \sum_{i \in C(h_1,h_2)} \delta(h_1 \in L(H,i))\delta(h_2 \in U(H,i)) \\
&= \sum_j \sum_k \sum_{h_1 \in G_j} \sum_{h_2 \in G_k} \frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(G_j,G_k)} \\
&\quad \sum_{i \in C(G_j,G_k)} \delta(h_1 \in L(H,i))\delta(h_2 \in U(H,i)) \\
&= \sum_j \sum_k \frac{1}{\mathbf{NC}(G_j,G_k)} \\
&\quad \sum_{i \in C(G_j,G_k)} \sum_{h_1 \in G_j} p(h_1,t)\delta(h_1 \in L(H,i)) \sum_{h_2 \in G_k} p(h_2,t)\delta(h_2 \in U(H,i)).
\end{aligned}
$$

From this one obtains the following

THEOREM 3 (MACROSCOPIC EXACT GP SCHEMA THEOREM) *The total transmission probability for a fixed-size-and-shape GP schema H under one-point crossover and no mutation is*

$$
\begin{aligned}
&\alpha(H,t) = (1-p_{xo})p(H,t) + \\
&\quad p_{xo} \sum_j \sum_k \frac{1}{\mathbf{NC}(G_j,G_k)} \sum_{i \in C(G_j,G_k)} p(L(H,i) \cap G_j, t)p(U(H,i) \cap G_k, t).
\end{aligned} \tag{17}
$$

The sets $L(H,i) \cap G_j$ and $U(H,i) \cap G_k$ either are (or can be represented by) fixed-size-and-shape schemata or are the empty set $\emptyset$. So, the theorem expresses the total transmission probability of H only using the selection probabilities of a set of lower- (or same-) order schemata.

This theorem is a refinement of Equation 15 which can be obtained from Equation 17 by considering only one term in the summations in j and k (the term for which $G_j = G_k = G(H)$). The theorem is a generalisation of Equation 12, which, as noted earlier, is a generalisation of Equation 4 (which is equivalent to Stephens and Waelbroeck's schema theorem in the absence of mutation). In turn that is a refinement of Holland's work. So, in the absence of mutation, *Equation 17 generalises and refines not only earlier GP schema theorems but also old and modern GA schema theories for one-point crossover.* The relation between all these results is illustrated in Figure 6. The exact macroscopic schema theorem for GP with one-point crossover described above occupies exactly the same elongated ellipsoids

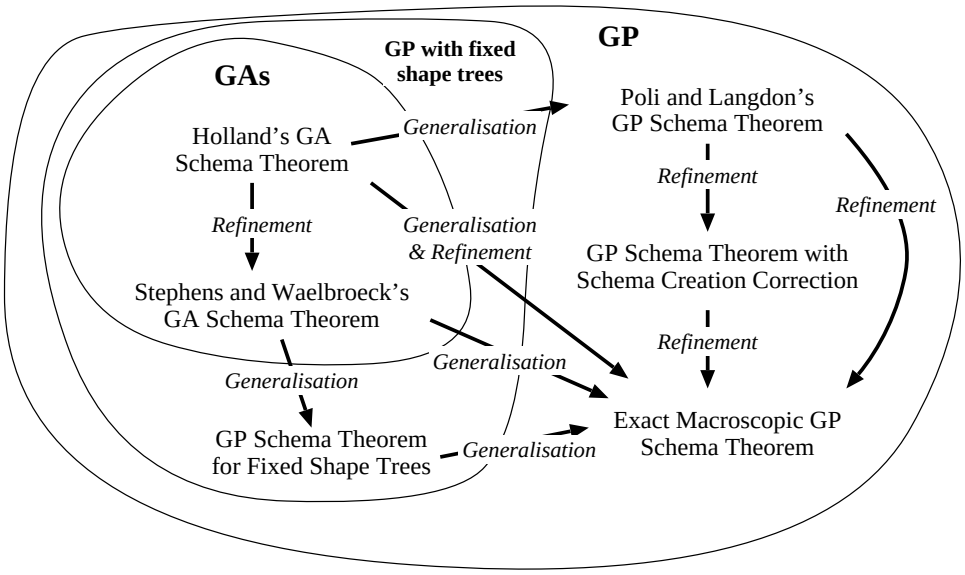

Figure 6. Relation between new, recent and old macroscopic schema theories for GAs and GP with one-point crossover.

within the triangular dashed spline in the lower part of Figure 1 as Stephens and Waelbroeck's schema theorem, but in the wider, more general space of GP models.

3.6. Effective Fitness for GP with One-point Crossover

Once the value of $\alpha(H,t)$ is available, it is easy to extend to GP with one-point crossover the notion of effective fitness provided in [54, 55]. By using the definition in Section 2.6 and the value of $\alpha(H,t)$ in Equation 17, we obtain the following

THEOREM 4 *The effective fitness of a fixed-size-and-shape GP schema H under one-point crossover and no mutation is*

$$\begin{aligned} f_{\text{eff}}(H,t) &= \frac{\alpha(H,t)}{p(H,t)} f(H,t) \\ &= f(H,t)\Big[1 - p_{xo}\Big(1 - \sum_{j,k} \sum_{i \in C(G_j,G_k)} \frac{p(L(H,i) \cap G_j, t)\, p(U(H,i) \cap G_k, t)}{\mathbf{NC}(G_j,G_k)\, p(H,t)}\Big)\Big]. \end{aligned}$$

This result gives the *true effective fitness for a GP schema* under one-point crossover: it is not an approximation or a lower bound. Thanks to this definition it is easy to see that the effective fitness of a GP schema can be bigger than

its actual fitness if its building blocks are abundant and relatively fit. This shows that crossover does not always have the destructive connotation often attributed to it in the GP literature (e.g. [24, 23]).

4. Examples

In this section we provide a few examples which show how to use the schema theory developed in this paper in practice and illustrate some of its benefits.

4.1. Linear Trees

In this example we will describe how to apply the exact schema theory in Equations 13 and 17 step by step. Since the calculations involved in applying the theory may become quite lengthy, we will consider one of the simplest non-trivial examples possible.

Let us imagine that we have a function set $\mathcal{F} = \{A_f, B_f, C_f, D_f, E_f\}$ including only unary functions, and the terminal set $\mathcal{T} = \{A_t, B_t, C_t, D_t, E_t\}$. So, for example, a program in this search space might look like $(A_f(B_f B_t))$. Since, the arity of all functions is 1, we can remove the parentheses from the expression obtaining $A_f B_f B_t$. In addition, since the only terminal in each expression is the rightmost node, we can remove the subscripts without generating any ambiguity, obtaining ABB. This can be done for every member of the search space, which can be seen as the space of variable-length strings over the alphabet $\{A, B, C, D, E\}$. So, in this example GP with one-point crossover is really a non-binary variable-length GA.

Let us now consider the schema `AB=`. We want to measure its total transmission probability under fitness proportionate selection and one-point crossover (with $p_{xo} = 1$) in two slightly different populations

Population 1	Fitness	Population 2	Fitness
AB	2	AB	2
BCD	2	BCD	2
ABC	4	ABC	4
ABCD	6	BCDE	6

where fitness has been assigned to individuals in a completely arbitrary manner. In order to do that we need first to compute the "lower part" and "upper part" building blocks of `AB=`. These are:

i	$L(\texttt{AB=}, i)$	$U(\texttt{AB=}, i)$
0	AB=	#
1	=B=	A #
2	===	AB#
3	$\emptyset$	$\emptyset$
$\vdots$	$\vdots$	$\vdots$

Let us start by calculating $\alpha(\texttt{AB=}, t)$ for Population 1, using Equation 13 (the exact microscopic schema theorem):

$$\begin{aligned}
\alpha(\texttt{AB=}, t) &= \sum_{h_1,h_2} \frac{p(h_1,t)p(h_2,t)}{\mathbf{NC}(h_1,h_2)} \sum_i \delta(h_1 \in L(\texttt{AB=}, i))\delta(h_2 \in U(\texttt{AB=}, i)) \\
&= \underbrace{\overbrace{\frac{2}{14}}^{p(h_1,t)} \times \overbrace{\frac{2}{14}}^{p(h_2,t)} \times \overbrace{\frac{1}{2}}^{1/\mathbf{NC}(h_1,h_2)} \times (0 \times 1 + 0 \times 1)}_{h_1=AB, h_2=AB, i=0,1} \qquad (18) \\
&+ \underbrace{\frac{2}{14} \times \frac{2}{14} \times \frac{1}{2} \times (0 \times 1 + 0 \times 0)}_{h_1=AB, h_2=BCD, i=0,1} + \cdots \quad \begin{matrix}(13 \text{ terms} \\ \text{are omitted})\end{matrix} \\
&+ \underbrace{\frac{6}{14} \times \frac{6}{14} \times \frac{1}{4} \times (1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1)}_{h_1=ABCD, h_2=ABCD, i=0,1,2,3} \\
&= \frac{43}{147} \approx 0.2925.
\end{aligned}$$

This equation contains a term for each pair of parents and for each crossover point. So, clearly this is a lengthy calculation, which can only produce a numerical result. It cannot really be used to understand how instances of AB= are created in different populations since it includes only microscopic quantities.

Let us now use Equation 17 to do the same calculation. First we 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, bigger shapes because the population does not contain any larger programs (i.e. $G_l = \emptyset$ for $l > 4$). Then we need to identify the schemata resulting from calculating $L(\texttt{AB=}, i) \cap G_j$ for all meaningful values of i and j:

	j			
i	1	2	3	4
0	$\emptyset$	$\emptyset$	AB=	$\emptyset$
1	$\emptyset$	$\emptyset$	=B=	$\emptyset$
2	$\emptyset$	$\emptyset$	===	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

We do the same for $U(\texttt{AB=}, i) \cap G_k$, obtaining:

	k			
i	1	2	3	4
0	=	==	===	====
1	$\emptyset$	A=	A==	A===
2	$\emptyset$	$\emptyset$	AB=	AB==
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Finally we need to evaluate the shape of the common regions to determine $\mathbf{NC}(G_j, G_k)$ and the crossover points in $C(G_j, G_k)$ for all valid values of j and k. In general common regions can naturally be represented using the program shapes G_1, G_2, etc.. For the example under consideration the shapes of common regions are:

	k			
j	1	2	3	4
1	G_1	G_1	G_1	G_1
2	G_1	G_2	G_2	G_2
3	G_1	G_2	G_3	G_3
4	G_1	G_2	G_3	G_4

By using this and the previous tables we can simplify Equation 17 removing all the null terms as follows:

$$
\begin{aligned}
\alpha(\texttt{AB=}, t) \;=\; & \sum_{j,k} \frac{1}{\mathbf{NC}(G_j, G_k)} \sum_i p(L(\texttt{AB=}, i) \cap G_j, t) p(U(\texttt{AB=}, i) \cap G_k, t) \\
= \; & \underbrace{\overbrace{\frac{1}{1}}^{1/\mathbf{NC}(G_j,G_k)} \times \overbrace{[\;\; 0 \times p(=) \;\;]}^{\sum_i p(L(\texttt{AB=},i)\cap G_j,t) p(U(\texttt{AB=},i)\cap G_k,t)}}_{j=1,k=1,i=0} \\
+ \; & \underbrace{\frac{1}{1} \times [0 \times p(==)]}_{j=1,k=2,i=0} + \cdots \quad \text{(6 more terms)} \\
+ \; & \underbrace{\frac{1}{1} \times [p(AB=) p(==)]}_{j=3,k=1,i=0} \\
+ \; & \underbrace{\frac{1}{2} \times [p(AB=) p(==) + p(=B=) p(A=)]}_{j=3,k=2,i=0,1} \\
+ \; & \cdots \quad \text{(6 more terms)},
\end{aligned}
$$

where for brevity we have used the notation $p(.)$ to represent $p(.,t)$. Simplifying yields

$$\begin{aligned}\alpha(\texttt{AB=},t) &= p(AB=)p(=) + \frac{1}{2}p(AB=)p(==) \\ &+ \frac{1}{2}p(=B=)p(A=) + \frac{1}{3}p(AB=)p(===) \\ &+ \frac{1}{3}p(=B=)p(A==) + \frac{1}{3}p(===)p(AB=) \\ &+ \frac{1}{3}p(AB=)p(====) + \frac{1}{3}p(=B=)p(A===) \\ &+ \frac{1}{3}p(===)p(AB==).\end{aligned} \tag{19}$$

The complexity of this equation can be reduced by using hyperschemata to represent groups of schemata, obtaining:

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

This is equivalent to reordering the terms by size and shape of common region and then by crossover point.

Equation 19 is quite different from the one obtained with the microscopic exact schema theorem. Thanks to the use of macroscopic quantities, it is general, i.e. independent of a particular population. Also, it clearly indicates how individuals sampling AB= can be assembled from individuals having different shapes and nodes. As discussed in Section 5.1, the schemata in this equation can be seen as the *building blocks* for AB=.

If we calculate the probabilities of selection of the schemata in the previous equation using Population 1, we obtain:

$$\begin{aligned}\alpha(\texttt{AB=},t) &= \frac{4}{14}\times 0 + \frac{1}{2}\times\frac{4}{14}\times\frac{2}{14} + \frac{1}{2}\times\frac{4}{14}\times\frac{2}{14} \\ &+ \frac{1}{3}\times\frac{4}{14}\times\frac{12}{14} + \frac{1}{3}\times\frac{4}{14}\times\frac{10}{14} + \frac{1}{3}\times\frac{6}{14}\times\frac{10}{14} \\ &= \frac{43}{147} \approx 0.2925\end{aligned}$$

The result is the same as the one in Equation 18 as expected. However, once the exact macroscopic formulation of the transmission probability of a schema is available, this is much easier to use in calculations than the corresponding microscopic description. Indeed, we can use it to calculate $\alpha(\texttt{AB=},t)$ for Population 2 with a simple pocket calculator obtaining $\alpha(\texttt{AB=},t) \approx 0.1905$.

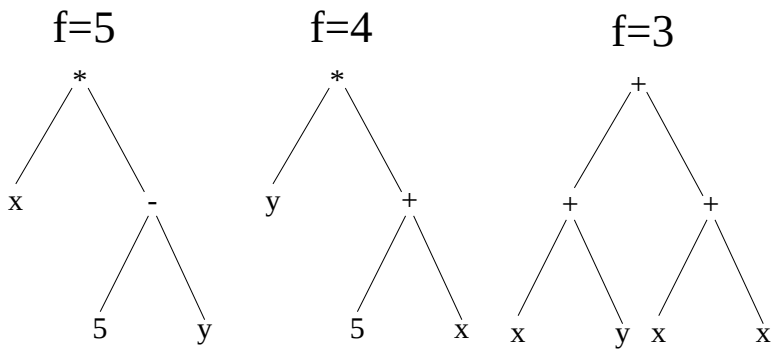

Figure 7. Example population of programs with their fitnesses.

4.2. Effective Fitness

With the values of $\alpha(\texttt{AB}=,t)$ computed in the previous section we can now compute the effective fitness for the schema `AB=` for both populations. For Population 1

$$f_{\text{eff}}(\texttt{AB}=,t) = \frac{\alpha(\texttt{AB}=,t)}{p(\texttt{AB}=,t)} f(\texttt{AB}=,t) \approx 4.1,$$

which is bigger than $f(\texttt{AB}=,t) = 4$. So, thanks to the collaboration of other schemata the schema `AB=` propagates faster with 100% crossover than with no crossover at all. On the contrary for Population 2

$$f_{\text{eff}}(\texttt{AB}=,t) = \frac{\alpha(\texttt{AB}=,t)}{p(\texttt{AB}=,t)} f(\texttt{AB}=,t) \approx 2.7.$$

So, in Population 2 the schema is effectively a below-average schema ($f_{\text{eff}}(\texttt{AB}=,t) = 2.7 < \bar{f}(t) = 3.5$) despite the fact that its fitness is above average. So, in this population, on average, the channels leading to the creation of the schema are could be said to be deceptive.

4.3. Comparison of Bounds Provided by Different Schema Theorems

Let us consider a population containing 10 copies of each of the 3 programs in Figure 7, and let us focus on the propagation of the schema $H' =$`(= = (= = =))`. Since this schema has no defining nodes, $\mathcal{L}(H') = 0$ and $B(H') = \emptyset$. Therefore, Equation 16 gives $\Delta\alpha(H',t) = 0$. So, the GP schema theorem with schema creation correction (see Section 3.4) cannot provide a better bound than the one in [33] (summarised in Section 2.4).

However, if one considers the schema $H'' =$`(* = (- 5 =))`, with simple calculations (assuming fitness proportionate selection) one obtains $\Delta\alpha(H'',t) \approx 0.07 \times p_{xo}$. This might look like a small difference, but if one multiplies this by the population

Table 1. Lower-bound and exact values for $E[m((* = (-5 =)), t+1)]$ provided by different schema theorems for the population in Figure 7.

Theorem	Value
Selection only	12.5
Exact Schema Theorem	10.5
Schema Theorem with Schema Creation Correction	9.4
Old Schema Theorem	7.3

size $M = 30$ one can see that the schema theorem with schema creation correction can provide a lower bound for $E[m(H'', t+1)]$ of up to about 2.1 higher than the old theorem (Equation 2), the maximum value being reached when $p_{xo} = 1$. This is a big difference considering that, as shown in Table 1, the correct value for $E[m(H'', t+1)]$ obtained from Equations 13 or 17 is 10.5 (for reference: the value obtained if selection only was acting is 12.5), and the lower bound provided by the old schema theorem is only 7.3. So, in this example the "schema creation correction" improves the estimate by nearly 30% providing a very tight lower bound of 9.4 for $E[m(H'', t+1)]$.

It is interesting also to compare the bounds provided by different schema theorems in the case of the linear-tree example described in Section 4.1. In that example the exact GP schema theorems gave $\alpha(\texttt{AB=}, t) \approx 0.2925$ for Population 1 and $\alpha(\texttt{AB=}, t) \approx 0.1905$ for Population 2. For comparison, for either population, the schema theorem with schema creation correction would have provided the lower bound:

$$\begin{aligned}\alpha(\texttt{AB=}, t) &\geq \frac{1}{3}p(AB =)p(===) \\ &+ \frac{1}{3}p(= B =)p(A ==) + \frac{1}{3}p(===)p(AB =) \\ &\approx 0.1088\end{aligned}$$

which is nearly one third of the correct value for Population 1 and a half of the correct value for Population 2. This is because the theorem accounts only for schema creation events in $G(\texttt{AB=}) \equiv$ `===`. Incidentally, since there are no creation events of this type in the example in Section 4.1, the old GP schema theorem (Equation 2) gives exactly the same bound.

4.4. Analysis of Schema Equations for Linear Representations

The calculations described in all the previous examples were performed by hand. As shown in Section 4.1, these kinds of calculations are quite lengthy and tedious. For more complex schemata, particularly with binary or higher-arity functions, expanding schema equations by hand is very time consuming. However, the steps involved in the calculations are simple syntactic manipulations that can all be automatised. In this and the following sections we report and analyse in detail

some schema equations for a few more realistic cases which have been obtained using a computer program implementing the expansion procedure. In all examples we will assume $p_{xo} = 1$.

Let us first consider again a linear system, but this time with individuals of up to length/depth 5. Then, the macroscopic schema-evolution equation for `ABC` is:

$$\begin{aligned}
&\alpha(\texttt{ABC}) = \\
&\quad \frac{1}{3}\,\texttt{p(ABC)}\,\texttt{p(=====)} + \frac{1}{3}\,\texttt{p(=BC)}\,\texttt{p(A====)} + \\
&\quad \frac{1}{3}\,\texttt{p(==C)}\;\texttt{p(AB===)} + \frac{1}{3}\,\texttt{p(ABC)}\,\texttt{p(====)} + \\
&\quad \frac{1}{3}\,\texttt{p(=BC)}\;\texttt{p(A===)} + \frac{1}{3}\,\texttt{p(==C)}\;\texttt{p(AB==)} + \\
&\quad \frac{1}{3}\,\texttt{p(ABC)}\;\texttt{p(===)} + \frac{1}{3}\,\texttt{p(=BC)}\,\texttt{p(A==)} + \\
&\quad \frac{1}{3}\,\texttt{p(==C)}\,\texttt{p(AB=)} + \frac{1}{2}\,\texttt{p(ABC)}\,\texttt{p(==)} + \\
&\quad \frac{1}{2}\,\texttt{p(=BC)}\,\texttt{p(A=)} + \texttt{p(ABC)}\,\texttt{p(=)},
\end{aligned}$$

where, for brevity, we omitted the generation number t. It is easy to show that the schema equation given in Section 4.1 is a special case of this equation. It is sufficient to replace `C` with `=` and assume $\texttt{p(=====)} = 0$ (whereby $\texttt{p(A====)} = \texttt{p(AB===)} = 0$).

It is interesting to specialise this equation to the shape `===` by substituting `A`, `B` and `C` with `=`, obtaining (after simplification):

$$\begin{aligned}
\alpha(\texttt{===}) &= \texttt{p(===)}\,\texttt{p(=====)} + \texttt{p(===)}\,\texttt{p(====)} + \\
&\quad \texttt{p(===)}^2 + \texttt{p(===)}\,\texttt{p(==)} + \texttt{p(===)}\,\texttt{p(=)} \\
&= \texttt{p(===)},
\end{aligned}$$

where we used the fact that the sum of the selection probability of all shapes must always be 1, i.e., in this example,

$$\texttt{p(=====)} + \texttt{p(====)} + \texttt{p(===)} + \texttt{p(==)} + \texttt{p(=)} = 1.$$

The result $\alpha(\texttt{===}) = \texttt{p(===)}$ is quite interesting: it states that the shape `===` will evolve (on average) as if selection only was acting on it. A similar result can be obtained for the other shapes in our linear GP population. So, the effective fitness of `===` is always identical to its actual fitness.

4.5. 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 expression/tree `(A B C)` (expressed in Lisp-like prefix notation) is:

$$
\begin{aligned}
&\alpha(\texttt{(A B C)}) = \\
&\quad \tfrac{1}{3}\texttt{p((= (= = =) C))}\,\texttt{p((A B (= = =)))} + \tfrac{1}{3}\texttt{p((= (= = =) C))}\,\texttt{p((A B =))} + \\
&\quad \tfrac{1}{3}\texttt{p((= B (= = =)))}\,\texttt{p((A (= = =) C))} + \tfrac{1}{3}\texttt{p((= B (= = =)))}\,\texttt{p((A = C))} + \\
&\quad \tfrac{1}{3}\texttt{p((A B C))}\,\texttt{p((= (= = =) (= = =)))} + \tfrac{1}{3}\texttt{p((A B C))}\,\texttt{p((= (= = =) =))} + \\
&\quad \tfrac{1}{3}\texttt{p((= B =))}\,\texttt{p((A (= = =) C))} + \tfrac{1}{3}\texttt{p((A B C))}\,\texttt{p((= = (= = =)))} + \\
&\quad \tfrac{1}{3}\texttt{p((= = C))}\,\texttt{p((A B (= = =)))} + \tfrac{1}{3}\texttt{p((A B C))}\,\texttt{p((= = =))} + \\
&\quad \tfrac{1}{3}\ \texttt{p((= B =))}\,\texttt{p((A = C))} + \tfrac{1}{3}\texttt{p((= = C))}\,\texttt{p((A B =))} + \texttt{p((A B C))}\,\texttt{p(=)}.
\end{aligned}
$$

It is interesting to specialise this equation for the shape `(= = =)` by substituting `A`, `B` and `C` with `=`, obtaining (after simplification):

$$
\begin{aligned}
&\alpha(\texttt{(= = =)}) \\
&= \tfrac{2}{3}\texttt{p((= (= = =) =))}\,\texttt{p((= = (= = =)))} + \texttt{p((= (= = =) =))}\,\texttt{p((= = =))} + \\
&\quad \texttt{p((= = (= = =)))}\,\texttt{p((= = =))} + \tfrac{1}{3}\ \texttt{p((= = =))}\,\texttt{p((= (= = =) (= = =)))} + \\
&\quad \texttt{p((= = =))}^2 + \texttt{p((= = =))}\,\texttt{p(=)} \\
&= \texttt{p((= = =))} + \qquad (20) \\
&\quad \tfrac{2}{3}\Big(\texttt{p((= (= = =) =))}\,\texttt{p((= = (= = =)))} - \texttt{p((= = =))}\,\texttt{p((= (= = =) (= = =)))}\Big),
\end{aligned}
$$

where, again, we used the fact that the sum of the selection probability of all shapes must always be 1. This result is quite surprising. It states that shape `(= = =)` will evolve (on average) as if selection only was acting on it, but only if the product $\texttt{p((= (= = =) =))}\,\texttt{p((= = (= = =)))}$ is exactly the same as the product $\texttt{p((= = =))}\,\texttt{p((= (= = =) (= = =)))}$. Why?

When one crosses over two parents belonging to any of the four shapes `(= = =)`, `(= (= = =) (= = =))`, `(= (= = =) =)` and `(= = (= = =))` , only offspring having one of these shapes can be produced. For example, when one crosses over an instance of the schema `(= = =)` with an instance of the schema `(= (= = =) (= = =))`, there are four possible outcomes: an instance of `(= (= = =) =)`, an instance of `(= = (= = =))` , an instance of `(= = =)` (if the crossover point is above the root node and the parent donating the root is in `(= = =)`) or an instance of `(= (= = =) (= = =))` (if the crossover point is above the root node and the parent donating the root is in `(= (= = =) (= = =))`). Likewise, if one crosses over an instance of the schema `(= (= = =) =)` with an instance of the schema `(= = (= = =))` , 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. 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 `(= = =)`, `(= (= = =) (= = =))`, `(= (= = =) =)`, and `(= = (= = =))` .

It is easy to convince ourselves that equilibrium is reached only when

$$\mathtt{p((=\ (=\ =\ =)\ =))\,p((=\ =\ (=\ =\ =)))} = \mathtt{p((=\ =\ =))\,p((=\ (=\ =\ =)\ (=\ =\ =)))},$$

at which point `(= = =)` will evolve as if selection only was acting on it. To do this, it is sufficient to compare Equation 20 with the schema evolution equations for the schemata `(= (= = =) (= = =))`, `(= (= = =) =)`, and `(= = (= = =))` :

$$\begin{aligned}
&\alpha(\mathtt{(=\ (=\ =\ =)(=\ =\ =))}) \\
&= \mathtt{p((=\ (=\ =\ =)(=\ =\ =)))} + \\
&\quad \frac{2}{3}\left(\mathtt{p((=\ (=\ =\ =)\ =))\,p((=\ =\ (=\ =\ =)))} - \mathtt{p((=\ =\ =))\,p((=\ (=\ =\ =)\ (=\ =\ =)))}\right),
\end{aligned}$$

$$\begin{aligned}
&\alpha(\mathtt{(=\ (=\ =\ =)\ =)}) \\
&= \mathtt{p((=\ (=\ =\ =)\ =))} - \\
&\quad \frac{2}{3}\left(\mathtt{p((=\ (=\ =\ =)\ =))\,p((=\ =\ (=\ =\ =)))} - \mathtt{p((=\ =\ =))\,p((=\ (=\ =\ =)\ (=\ =\ =)))}\right),
\end{aligned}$$

$$\begin{aligned}
&\alpha(\mathtt{(=\ =\ (=\ =\ =))}) \\
&= \mathtt{p((=\ =\ (=\ =\ =)))} - \\
&\quad \frac{2}{3}\left(\mathtt{p((=\ (=\ =\ =)\ =))\,p((=\ =\ (=\ =\ =)))} - \mathtt{p((=\ =\ =))\,p((=\ (=\ =\ =)\ (=\ =\ =)))}\right),
\end{aligned}$$

where, again, we used the fact that the sum of the selection probability of all shapes must always be 1. So, the schema evolution equations for the schemata `(= = =)` and `(= (= = =) (= = =))` have the form

$$\alpha(H) = p(H) + \frac{2}{3}\Delta p$$

while the schema evolution equations for the schemata `(= (= = =) =)`, and `(= = (= = =))` have the form

$$\alpha(H) = p(H) - \frac{2}{3}\Delta p$$

where

$$\Delta p = \mathtt{p((=\ (=\ =\ =)\ =))\,p((=\ =\ (=\ =\ =)))} - \mathtt{p((=\ =\ =))\,p((=\ (=\ =\ =)\ (=\ =\ =)))}.$$

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 one-point crossover.

When $\Delta p = 0$ the evolution equations of the schemata `(= (= = =) (= = =))`, `(= = =)`, `(= (= = =) =)`, and `(= = (= = =))` 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 one-point crossover becomes unbiased.

Interestingly, the components $\frac{2}{3}$`p((= (= = =) =))p((= = (= = =)))` and $\frac{2}{3}$`p((= = =))p((= (= = =) (= = =)))` in the schema equations for `(= = =)` and `(= (= = =) (= = =))` can be interpreted as *schema creation* and *schema disruption* probabilities (the role is reversed for `(= (= = =) =)` and `(= = (= = =))`). The component $p(H)$ in such equations could be interpreted as the probability of *schema survival* to crossover.

5. Discussion

Schema theories, even when exact, need to prove that they are useful. As with other theories, different people require different kinds of proofs and ask different kinds of questions:

- *Practitioners* will ask the following question: "OK, perhaps schema theories explain what happens and why in the next generation, but this does not help GP practitioners. The real question is: Which representation, operators, fitness function, search algorithm, population size, number of generations, number of runs, crossover probability, anti-bloat method, etc. should one use?"
- *Theorists* who are proponents or believers of other approaches to theory will instead ask a different question: "What does this schema theory give me that another preexisting theory does not?"

This work is very recent, so, rather obviously, it is difficult at this stage to be able to provide complete answers for all questions. However, in the following subsections we will discuss what the exact GP schema theory can provide now and what it can be expected to provide in the near future both from the theory and from the practice points of view.

5.1. Building Blocks in GAs and GP

The *building block hypothesis* [11] is typically stated informally by saying that a GA works by combining short, low-order schemata of above average fitness (building blocks) to form higher-order ones over and over again until it converges to an optimum or near-optimum solution.

The building block hypothesis and the related concept of deception have been strongly criticised in [12], in the context of binary GAs, where it was suggested

that these ideas can only be used to produce first-order approximations of what really happens in a GA. In [24] the criticisms to the building block hypothesis have been extended to the case of GP with standard crossover, arguing that the hypothesis is even less applicable to GP because of the highly destructive effects of crossover. The analysis of Equation 2 reported in [33, 36] and the experimental results in [32] suggested that the latter criticism might not apply to GP with one-point crossover. Therefore, we argued that the building block hypothesis might still be considered a reasonable first approximation of the behaviour of GP with one-point crossover.

Stephens and Waelbroeck analysed their schema theorem (equivalent to Equation 4) and stated that the presence of terms of the form $p(L(H,i),t)p(R(H,i),t)$ is a clear indication of the fact that, indeed, GAs build higher-order schemata (H) by juxtaposing lower-order ones (the $L(H,i)$'s and $R(H,i)$'s) [54, 55]. However, these building blocks are not necessarily all fitter than average, short or even low-order. The GA will use them as building material provided there are enough of them. In [56] this was demonstrated explicitly by simulations that are valid for any non-epistatic landscape.

Since the results presented in Section 3.5 are a generalisation of Stephens and Waelbroeck's schema theory, it should be expected that exactly the same can be said for GP with one-point crossover. Indeed, the presence of the terms $p(L(H,i) \cap G_j,t)p(U(H,i) \cap G_k,t)$ suggests that the schemata $L(H,i) \cap G_j$ and $U(H,i) \cap G_k$ (for all meaningful values of i, j and k) are a form of building blocks for fixed-size-and-shape schemata. So, it is arguable that also GP with one-point crossover builds higher-order schemata by juxtaposing lower-order ones. Again for this to happen the building blocks need not necessarily be all fitter than average, short or even low-order. If there are many copies of a pair of building blocks $L(H,i) \cap G_j$ and $U(H,i) \cap G_k$, GP will assemble them into the higher-order schema H even if these schemata are below-average fitness.

5.2. Introns, Bloat, Code Compression

Schemata can easily represent classes of programs with the same active code (and, therefore, the same behaviour) but different amounts of inactive code. For example, see the three schemata in Figure 8. These schemata contain programs having exactly the same fitness (assuming this is not a function of the size or shape of the trees). So, the schemata have the same fitness, too.

A few studies of bloat based on exact schema theories have recently become available for the case of linear, variable-length representations under different crossover and mutation operators [39, 19, 20]. These are very early results, but they indicate that it is possible to use schema theories and exact definitions of effective fitness, such as the one proposed in this paper, to establish whether there is an effective fitness advantage in having large amounts of inactive code, and if so why and when. So, when these studies are extended to tree like structures, for example, it might be possible to show that the schemata in Figure 8 have different effective fitnesses despite the fact that their average fitnesses are the same. In the long run, this

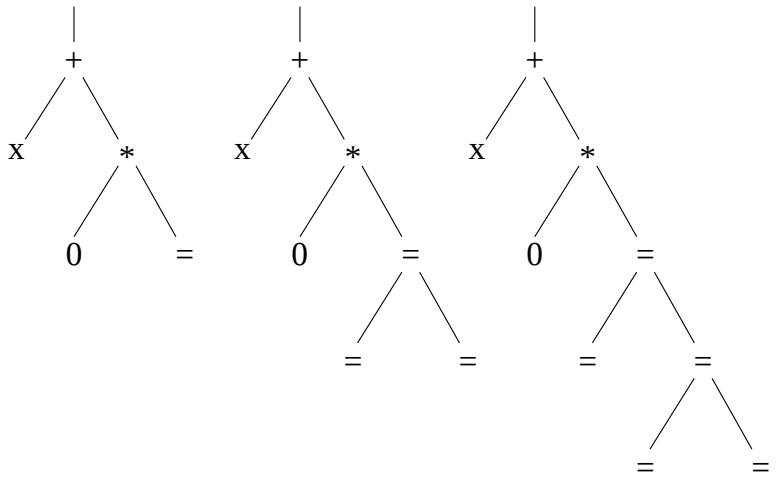

Figure 8. Schemata representing programs with the same active code (and behaviour) but with different amounts of inactive code.

kind of work will certainly lead to a better understanding of bloat and to better anti-bloat measures.

It is also possible to represent equivalent code fragments using schemata. For example, each of the three schemata in Figure 9 represents a different way of implementing the same behaviour using different amounts of active code but the same inactive code. For each program in each schema, there is a corresponding program in any another schema with exactly the same behaviour. So, the three schemata have exactly the same fitness.

Although, no exact study on this subject is available yet, it seems possible to use the concept of effective fitness to establish whether there is an advantage in having more compact active code, and if so why and when. For instance, it might be possible to show that the effective fitness of the schema on the left in Figure 9 is higher than that of the other two schemata. This kind of study may lead to a better understanding of code compression and to better generalisation-promotion measures.

5.3. Operators, Biases, Parameters, Initialisation

Most people see GA/GP theory as a way of modelling existing algorithms, representations and operators. However, theory can also indicate new directions. For example, the introduction of one-point crossover for GP in [33] was not an act of creative innovation. One-point crossover was suggested as a natural operator to act on fixed-size-and-shape schemata by the theory itself: in developing the the-

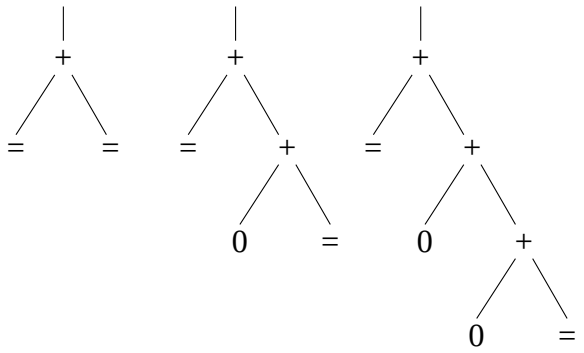

Figure 9. Schemata representing different ways of implementing the same behaviour using different amounts of active code but exactly the same inactive code.

ory it became clear that without an operator that swapped subtrees in exactly the same position in the two parents, the mathematics involved in obtaining a schema theorem was much too complex. In turn, this operator suggested other "homologous" operators based on the concept of a common region, such as GP uniform crossover [34], which later led to the notion of smooth operators [25, 41, 40].

Not many people in the GP community presently use one-point crossover in practical applications. Most people still use standard crossover. So, as often happens with GA/GP theory, the theory presented in this paper might seem not to have a big impact for practitioners. However, there is really considerable scope for extending the work presented in this paper to other operators and representations. For example, an exact schema theorem for standard crossover is ready and will be presented in the near future [31]. Also, by extending to GP the ideas introduced, for binary GAs, in [2][Section 3 and Appendix], it is possible to derive a macroscopic schema theorem valid for *any* type of homologous GP crossover operator, including uniform crossover. The exact schema theorem presented in [54, 55] point mutation was present. So, it seems possible to extend the results presented in this paper to the case of point mutation (either alone or with some form of crossover). A similar extension for subtree mutation has recently been been obtained [38].

As shown, for example, by a study based on a version of the schema theorem for standard crossover applicable to linear structures [39], the availability of exact models for different operators allows a formal study of the biases of those operators and a comparison of their performance.

As mentioned earlier, the quantity $\delta(h_1 \in L(H,i))\delta(h_2 \in U(H,i))$, which is used to obtain the exact schema theorems in this paper, can be considered as a measurement function (see [2]) for the distribution of parents and crossover points. By using different measurement functions one can obtain schema-theorem-like results that clarify the behaviour (and highlight other biases) of the operators as well as the influence of their parameters on the search. For example, in [34] we studied the

amount of genetic material exchanged by different crossovers in GP. This revealed that, for trees, standard crossover is generally a local search operator. Therefore, standard GP searches the space of programs like a set of hill-climbers. Also, we showed that one-point crossover is slightly less local, while uniform crossover is a global search operator.

Knowing the biases introduced by the operators is very important, since it allows an informed choice of operators and parameter settings for particular problems. In the future exact mathematical formulations of these biases, such as the ones derivable from schema theories, might also allow the definition of optimum operators and parameters for different problems or even classes of problems. Optimality of operators and parameters would obviously depend not only on the particular problem(s) to be solved, but on the initial population. So, one would probably have to define optimum operators and parameters as those which maximise the average performance of a GA/GP system w.r.t. the probability distribution of the populations produced by the initialisation algorithm.

Many initialisation strategies have been proposed in the GP literature [15, 4], some of which have a theoretical motivation [16, 17]. Other theoretically sound initialisations strategies could be derived thanks to the knowledge of the biases of the operators obtained from exact schema theories. In general purpose evolutionary algorithms, a good strategy might be to initialise the population so as to minimise the biases of the operators in the first generation — a particularly important stage in a run. This could be achieved, for example, by minimising the linkage disequilibrium for shapes for the particular operators chosen (for one-point crossover acting on binary trees this corresponds to minimising quantities like the Δp introduced in Section 4.5). If instead one is interested in optimising/specialising an evolutionary algorithm for a particular class of applications, a good initialisation strategy might be to maximally bias the search operators towards areas of the search space with the highest density of solutions. Steps forward in this direction have recently been made in [39, 19, 20].

5.4. Convergence, Population Sizing, GP Hardness, Deception

In recent work [26] schema theorems were presented in which the expectation operator is absent. These are based on the idea of applying Chebyshev's inequality [51] to the stochastic variable $m(H, t+1)$. This allows one to transform an exact statement on the expected value and the variance of $m(H, t+1)$, into a probabilistic statement on the interval with which $m(H, t+1)$ will vary. By using this in conjunction with Stephens and Waelbroeck's schema theory, in [29] a recursive conditional schema theorem for GAs was derived which allows one to predict the probability of obtaining a solution for a problem in a fixed number of generations assuming that the fitness of the building blocks and of the population are known. This in turn led to the formulation of conditional population sizing equations.

Although these results are very conservative, nothing prevents the extension of the conditional-convergence theory for GAs reported in [29] to GP by using the macroscopic exact GP schema theorem presented in this paper. In the future, this

might allow to identify rigorous strategies to size the population and, therefore, to calculate the computational effort required to solve a given problem using GP.

The availability of rigorous computational effort equations would open the way to a precise definition of "GP-friendly" ("GP-easy") fitness functions. Such functions would simply be those for which the number of fitness evaluations necessary to find a solution with say 99% probability in multiple runs is smaller (much smaller) than 99% of the effort required by exhaustive search or random search without resampling. An alternative approach to define GP-hardness would be to use the exact definition of effective fitness to obtain a rigorous definition of *deception* for GP (a topic that has been firstly studied in [18]).

6. Conclusions

Schema theories are models of GAs and GP the level of coarse graining of which can be varied with some freedom. Unlike fully microscopic models, which made a huge step forward thanks to the work of Michael Vose and others in the early nineties, only approximate or worst-case-scenario schema theories have been available until very recently. This was particularly true for GP, whose theory historically has always been well behind due to the fact that it is a younger field and to the complexities of building theories for variable size structures.

In the last ten years or so some researchers have overinterpreted these approximate schema models, leading to the formulation of hypotheses on how genetic algorithms work, firstly highly celebrated and later disproved or shown to be only first order approximations. Other researchers have just incorrectly interpreted such models, while still others have understood and correctly criticised them for their limitations. All this has led to the perception that schema theories are intrinsically inferior to other models and that they are basically only a thing of the past to be either criticised to death or just swept under the carpet.

We believe that this is simply wrong: schema theories are neither inferior to microscopic models nor things of the past. It is important to understand that whatever their coarse graining, exact models must be equivalent. They only differ on the basis of the level of abstraction of the language (i.e. the quantities) used to describe what happens inside a GA. In any modelling effort, the choice of the quantities on which to focus the analysis determines crucially which phenomena are put in the foreground and which in the background. So, any model makes the calculations involved in explaining a particular behaviour of a system very easy and, unavoidably, others very difficult. This applies to exact models of GAs: in no way can a microscopic model be superior to a more macroscopic model from all points of view, and *vice versa*. Had the evolutionary computation community understood this, the early momentum in the development of schema theories would have continued and would have led to the production and acceptance of exact macroscopic schema theories much, much earlier.

Fortunately, the situation is now changing. After a period of skepticism at last the importance of Stephens and Waelbroeck's 1997 work is now recognised and schema theories are making a comeback. In this paper we have provided an exact

macroscopic schema theory which extends Stephens and Waelbroeck's GA work to genetic programming. As shown in Figure 6, this theory is very general: when mutation is absent, it extends and/or refines not only Stephens and Waelbroeck's theory, but also Holland's schema theorem and our earlier schema theory for GP. This shows that, although doing theory for tree-like structures of variable size and shape can be quite complex, the rewards are bigger since the resulting theory is more general and includes GAs as a special case. This is not to say that the theory for GAs is obsolete: the theory of genetic algorithms can give very valuable suggestions as to how to produce good theory for genetic programming. So, GP and GA theories are not only converging more and more and but they also mutually support the development of each other.

At present no Vose-like exact microscopic model is available for structures of variable size and shape and for infinite search spaces. So, it is arguable that macroscopic models are presently well ahead of microscopic models. Obviously, we expect that a microscopic theory equivalent to our macroscopic results will sooner or later be proposed. We would really welcome such a theory, because, although in theory equivalent to our results, it would certainly also complement them in practice.

To summarise, we think the main contributions of this paper are the following:

- We have clarified how different types of exact and approximate GA and GP models are related.
- We have developed an exact schema theory that makes the effects and the mechanisms of schema creation and schema survival in GP clear at last.
- We have shown how this theory is equally applicable to GP as well as to fixed-length and variable length GAs.
- We have proven that, in the absence of mutation, the theory is a generalisation and/or refinement of Holland's schema theorem, of Stephens and Waelbroeck's exact schema theory for GAs, and of our previous GP schema theorem.
- We have clarified that building blocks in GP and variable-size GAs with one-point crossover exist, but that they are not necessarily all short, low-order or highly fit.
- We have been able to express exactly the notion of effective fitness for GP. Since this was previously used to explain the reasons for bloat and active-code compression we have established a formal link between two important areas of theoretical research in GP: the study of bloat and the theory of schemata.
- We have indicated how the exact definition of effective fitness might lead to a formal definition of deception in GP.
- We have shown how schema theories can be used to study the bias of the operators and we have highlighted the existence of disequilibria for shapes similar to the linkage disequilibrium for genes.

- We have indicated how it will be possible to use schema theories to obtain mathematical recipes to initialise the population and choose the operators and their parameters.

- We have delineated ways in which to prove GP convergence, which may lead to an exact formulation of GP hardness and to population sizing equations.

We hope this work will convince other researchers that schema theories are (and will be) very useful in analysing and designing GAs and that the scepticism with which they are dismissed in the evolutionary computation community is becoming less and less justifiable.

Acknowledgements The author would like to thank the members of the EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) group at Birmingham, Nic McPhee, Chris Stephens, Jonathan Rowe and W. B. Langdon for useful discussions and comments. Wolfgang Banzhaf and the reviewers are also warmly thanked for their help in improving this paper.

Notes

1. The paper is an integration and extension of two previous papers [28, 27] which appeared in conference proceedings.
2. This is a slightly different version of Holland's original theorem. This applies when crossover is performed taking both parents from the mating pool [11, 60] and is applicable to any selection-with-replacement scheme. It is also assumed that the crossover point before the first bit in a bit string is allowed. This makes it easier to compare Equation 1 with the corresponding results for GP.
3. The model proposed in [1] is a microscopic model of the propagation of program components (subtrees) in GP with standard crossover, rather than a theorem about schemata as sets. This model allows one to compute the exact proportion of subtrees of a given type in an infinite population in the next generation given information about the programs in the current generation.
4. For symmetry, in this paper we use the convention that the root node of a program or a schema also has an output link. This leads to minor changes in the results obtained in our earlier work. Whenever these are reported here we also modify them to take this extra link into account.
5. The definition is lower-level than those proposed by others, as a smaller number of trees are represented by schemata with the same number of "don't care" symbols and it is possible to represent other types of schemata by using a collection of these. This makes it possible to export some results of the fixed-size-and-shape schema theory to other kinds of schemata.
6. For example the schema `(+ (- = =) x)` has order 3 and defining length 2. See [33, 36] for more details and examples.
7. The symbol L stands for "left part of", while R stands for "right part of".
8. In [54, 55] the summation in Equation 4 is performed only over the crossover points between the extreme defining bits in a schema.
9. Clearly the quality of the approximation depends also on the degree of linkage disequilibrium in the population. Here, by this we mean the quantity $p(H,t) - p(L(H,i),t)p(R(H,i),t)$ rather than the quantity $\frac{m(H,t)}{M} - \frac{m(L(H,i),t)}{M} \times \frac{m(R(H,i),t)}{M}$ normally used in genetics.
10. Any numbering scheme for the crossover points produces the same result.

11. This is because GP one-point crossover and GA one-point crossover behave in exactly the same way under these conditions. Since Equation 12 is applicable only to structures of a fixed size (and shape), one could think of the action of GP one-point crossover on non-linear trees as the action of a non-standard GA crossover acting on the flattened representation of such trees. For example, GP one-point crossover operating on structures matching the schema $(= \cdots =)$ of size N (a tree including only a root node whose arguments are $N-1$ terminals) would behave like a GA crossover operator acting on strings of length N which, with equal probability, does one of the following actions: swaps all the bits in one parent with all the bits in the other, swaps bit 2 only, swaps bit 3 only, ... , swaps bit N only.
12. The operators `=` and `#` should be interpreted as polymorphic functions with as many arities as the number of different arities of the elements of $\mathcal{F} \cup \mathcal{T}$, the terminals in $\mathcal{T}$ being 0-arity functions.
13. We assume $p_{\text{diff}}(t) = 1$ to maintain the generality of our results.

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. World Scientific Publishing, 1994.
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, Estes Park, Colorado, USA, 31 July–2 Aug, 1994. Morgan Kaufmann: San Francisco, 1995.
3. T. Bäck and D. B. Fogel. Glossary. In T. Bäck, D. B. Fogel, and T. Michalewicz, editors, *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing: Bristol and Philadelphia, 2000.
4. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann: San Francisco, 1998.
5. S. W. Chung and R. A. Perez. The schema theorem considered insufficient. In *Proceedings of the Sixth IEEE International Conference on Tools with Artificial Intelligence*, pages 748–751, New Orleans, IEEE Press: Piscataway, NJ, Nov 6–9 1994.
6. T. E. Davis and J. C. Principe. A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, 1(3):269–288, 1993.
7. K. A. De Jong, W. M. Spears, and D. F. Gordon. Using Markov chains to analyze GAFOs. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 115–137. Morgan Kaufmann: San Francisco, CA, 1995.
8. D. B. Fogel and A. Ghozeil. Schema processing under proportional selection in the presence of random effects. *IEEE Transactions on Evolutionary Computation*, 1(4):290–293, 1997.
9. D. B. Fogel and A. Ghozeil. The schema theorem and the misallocation of trials in the presence of stochastic effects. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII: Proc. of the 7th Ann. Conf. on Evolutionary Programming*, pages 313–321, Berlin, Springer, 1998.
10. D. E. Goldberg. Genetic algorithms and Walsh fuctions: II. Deception and its analysis. *Complex Systems*, 3(2):153–171, 1989.
11. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley: Reading, Massachusetts, 1989.
12. J. J. Grefenstette. Deception considered harmful. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA, Morgan Kaufman: San Francisco, 1993.
13. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.
14. J. H. Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):373–391, 2000.
15. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press: Cambridge, MA, USA, 1992.

16. W. B. Langdon. Size fair and homologous tree genetic programming crossovers. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1092–1097, Orlando, Florida, USA, Morgan Kaufmann: San Francisco, 13-17 July 1999.
17. W. B. Langdon. Size fair and homologous tree genetic programming crossovers. *Genetic Programming And Evolvable Machines*, 1(1/2):95–119, 2000.
18. W. B. Langdon and R. Poli. Why ants are hard. 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 193–201, University of Wisconsin, Madison, Wisconsin, USA, Morgan Kaufmann: San Francisco, 22-25 July 1998.
19. N. F. McPhee and R. Poli. A schema theory analysis of the evolution of size in genetic programming with linear representations. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, Milan, Springer-Verlag: Berlin, 18-20 Apr. 2001. Forthcoming.
20. N. F. McPhee, R. Poli, and J. E. Rowe. A schema theory analysis of mutation size biases in genetic programming with linear representations. Technical Report CSRP-00-24, University of Birmingham, School of Computer Science, December 2000.
21. A. E. Nix and M. D. Vose. Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.
22. P. Nordin and W. Banzhaf. Complexity compression and evolution. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA, Morgan Kaufmann: San Francisco, 15-19 July 1995.
23. P. Nordin, F. Francone, and W. Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In J. P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 6–22, Tahoe City, California, USA, 9 July 1995.
24. 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, Estes Park, Colorado, USA, Morgan Kaufmann: San Francisco, 31 July–2 Aug. 1994, 1995.
25. J. Page, R. Poli, and W. B. Langdon. Smooth uniform crossover with smooth point mutation in genetic programming: A preliminary study. In R. Poli, P. Nordin, W. B. Langdon, and T. C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 39–49, Goteborg, Sweden, Springer-Verlag: Berlin, 26-27 May 1999.
26. R. Poli. Schema theorems without expectations. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, page 806, Orlando, Florida, USA, Morgan Kaufmann: San Francisco, 13-17 July 1999.
27. R. Poli. Exact schema theorem and effective fitness for GP with one-point crossover. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 469–476, Las Vegas, Morgan Kaufmann: San Francisco, July 2000.
28. R. Poli. Hyperschema theory for GP with one-point crossover, building blocks, and some new results in GA theory. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 163–180, Edinburgh, Springer-Verlag: Berlin, 15-16 Apr. 2000.
29. R. Poli. Recursive conditional schema theorem, convergence and population sizing in genetic algorithms. In W. M. Spears and W. Martin, editors, *Proceedings of the Foundations of Genetic Algorithms Workshop (FOGA 6)*, Charlottesville, VA, USA, July 2000. In press.
30. R. Poli. Why the schema theorem is correct also in the presence of stochastic effects. In *Proceedings of the Congress on Evolutionary Computation (CEC 2000)*, pages 487–492, San Diego, USA, IEEE: Piscataway, NJ, July 2000.
31. R. Poli. General schema theory for genetic programming with subtree-swapping crossover. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, Milan, Springer-Verlag: Berlin, 18-20 Apr. 2001. Forthcoming.

32. R. Poli and W. B. Langdon. An experimental analysis of schema creation, propagation and disruption in genetic programming. In T. Back, editor, *Genetic Algorithms: Proceedings of the Seventh International Conference*, pages 18–25, Michigan State University, East Lansing, MI, USA, Morgan Kaufmann: San Francisco, 19-23 July 1997.
33. R. Poli and W. B. Langdon. A new schema theory for genetic programming with one-point crossover and point mutation. 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 278–285, Stanford University, CA, USA, Morgan Kaufmann: San Francisco, 13-16 July 1997.
34. 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, Morgan Kaufmann: San Francisco, 22-25 July 1998.
35. R. Poli and W. B. Langdon. A review of theoretical and experimental results on schemata in genetic programming. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 1–15, Paris, Springer-Verlag: Berlin, 14-15 Apr. 1998.
36. 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.
37. R. Poli, W. B. Langdon, and U.-M. O'Reilly. Analysis of schema variance and short term extinction likelihoods. 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 284–292, University of Wisconsin, Madison, Wisconsin, USA, Morgan Kaufmann: San Francisco, 22-25 July 1998.
38. R. Poli and N. F. McPhee. Exact GP schema theory for headless chicken crossover and subtree mutation. Technical Report CSRP-00-23, University of Birmingham, School of Computer Science, December 2000.
39. R. Poli and N. F. McPhee. Exact schema theorems for GP with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, Milan, Springer-Verlag: Berlin, 18-20 Apr. 2001. Forthcoming.
40. R. Poli and J. Page. Solving high-order boolean parity problems with smooth uniform crossover, sub-machine code GP and demes. *Genetic Programming And Evolvable Machines*, 1(1/2):37–56, Apr. 2000.
41. R. Poli, J. Page, and W. B. Langdon. Smooth uniform crossover, sub-machine code GP and demes: A recipe for solving high-order boolean parity problems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1162–1169, Orlando, Florida, USA, Morgan Kaufmann: San Francisco, 13-17 July 1999.
42. A. Prügel-Bennett and J. L. Shapiro. An analysis of genetic algorithms using statistical mechanics. *Physical Review Letters*, 72:1305–1309, 1994.
43. N. J. Radcliffe. Schema processing. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.5–1–10. IOP Press: NY, 1997.
44. 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, Morgan Kaufmann: San Francisco, 13-16 July 1997.
45. 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. Morgan Kaufmann: San Francisco, 1999.
46. G. Rudolph. Convergence analysis of canonical genetic algorithm. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.
47. G. Rudolph. Genetic algorithms. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.4–20–27. IOP Press: NY, 1997.
48. 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. IOP Press: NY, 1997.

49. G. Rudolph. Stochastic processes. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.2–1–8. IOP Press: NY, 1997.
50. W. M. Spears. Aggregating models of evolutionary algorithms. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 631–638, Mayflower Hotel, Washington D.C., USA, IEEE Press: Piscataway, NJ, 6–9 July 1999.
51. M. R. Spiegel. *Probability and Statistics.* McGraw-Hill: New York, 1975.
52. C. R. Stephens and J. M. Vargas. Effective fitness as an alternative paradigm for evolutionary computation I: general formalism. *Genetic Programming and Evolvable Machines*, 1(4):363–378, 2000.
53. C. R. Stephens and J. M. Vargas. Effective fitness as an alternative paradigm for evolutionary computation II: examples and applications. *Genetic Programming and Evolvable Machines*, 2001. Forthcoming.
54. 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, Morgan Kaufmann: San Francisco, 1997.
55. C. R. Stephens and H. Waelbroeck. Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124, 1999.
56. C. R. Stephens, H. Waelbroeck, and R. Aguirre. Schemata as building blocks: Does size matter? In W. Banzhaf and C. Reeves, editors, *Foundations of Genetic Algorithms 5*, pages 117–133. Morgan Kaufmann: San Francisco, CA, 1999.
57. M. D. Vose. *The simple genetic algorithm: Foundations and theory.* MIT Press: Cambridge, MA, 1999.
58. P. A. Whigham. A schema theorem for context-free grammars. In *1995 IEEE Conference on Evolutionary Computation*, volume 1, pages 178–181, Perth, Australia, IEEE Press: Piscataway, NJ, 29 Nov. - 1 Dec. 1995.
59. 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, 14 Oct. 1996.
60. D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Department of Computer Science, Colorado State University, Aug. 1993.