On the Advantages of Variable Length GRNs for the Evolution of Multicellular Developmental Systems

Martin A. Trefzer, Tüze Kuyucu, Julian F. Miller, Andy M. Tyrrell

***Abstract*—Biological genomes have evolved over a period of millions of years and comprise thousands of genes, even for the simplest organisms. However, in nature, only 1–2% of the genes play an active role in creating and maintaining the organism, while the majority are evolutionary fossils. This raises the question whether a considerably larger number of (partly redundant) genes are required in order to effectively build a functional developmental system, of which, in the final system only a fraction is required for the latter to function. This paper investigates different approaches to creating artificial developmental systems (ADSs) based on variable length gene regulatory networks (GRNs). The GRNs are optimised using an evolutionary algorithm (EA). A comparison is made between the different variable length representations and fixed length representations. It is shown that variable length GRNs can achieve both reducing computational effort during optimisation and increasing speed and compactness of the resulting ADS, despite the higher complexity of the encoding required. The results may also improve the understanding of how to effectively model GRN based developmental systems. Taking results of all experiments into account makes it possible to create an overall ranking of the different patterns used as a testbench in terms of their complexity. This ranking may aid to compare related work against. In addition this allows a detailed assessment of the ADS used and enables the identification of missing mechanisms.**

***Index Terms*—Artificial Development, Gene Regulatory Network, Computational Evolution, Evolvability**

I. Introduction

BIOLOGICAL GENOMES have evolved over a period of millions of years and comprise thousands of genes, even in the case of the simplest organisms. However, in nature, only a very small fraction of the genes encode protein sequences and more than 98 % are non-coding DNA. While most protein functions are known, the role of the non-coding genes in creating and maintaining an organism is not yet fully understood [1], [2]. When using biological development as an inspiration for systems in engineering and computer science with the aim to harness key features of nature, such as adaptive growth, scalability, robustness and self-repair, one realises quickly that mechanisms that make things work in biology cannot be easily realised in the same fashion in computational models and systems. Furthermore, the sequence and parallelism in which genes are expressed, the spatial distribution and the stochastic nature of information (chemicals) processing within cells and organisms, and the large amount of redundancy and non-coding genes in the genome influence how mechanisms of development work in biology. The entire range of approaches can be taken between modelling development as a set of simple (conditional, reproductive) rules to accurately modelling cell chemistry/topology.

Although the process of development in biology is clearly defined—the ability of biological organisms to grow from a single cell into a multicellular organism via gene expression and gene regulation using the same genotype for all cells, which can then specialise to form different parts of an organism—its definition in evolutionary computation (EC) can be different depending on the author [3]–[11]. While some algorithms try to closely model biological development, others are simply inspired by a few specific mechanism of biological development, which results in different understandings of development. For example, Roggen's diffusion based developmental model [12], the self modifying cartesian genetic programming by Harding et al. [7], and Stanley's pattern producing networks [9] are examples of systems that use simple inspirations from a few biological developmental mechanisms. Although most commonly referred to as artificial development, its name can also take many other forms; computational embryology, artificial embryology, artificial embryogeny, artificial ontogeny, organic computing, computational development [3]. In this paper, the term development refers to the formation and maintenance of a multicellular organism, according to Wolpert's definition of biological development [13].

However, while the field of computational biology and modelling biology continuously progresses and provides useful and established models for theoretical biology, applications in real-world systems like electronic systems, embedded systems, developmental software and robotics still remain at very basic proof-of-concept level. There are several reasons for this: first, when the goal is to create accurate models for biology in order to make predictions for biological systems, the trade-off between computational resource demands and complexity/accuracy can usually be made in favour of the latter. Second, when researching nature the questions asked arise solely from the subject itself, which does not make modelling any easier but it removes the requirement to implement and interface such systems in non-natural embodiments of which it is not clear how to interface them and whether they are designed in a way that compliments the principles of biological development. Third, while it seems to be obvious which properties of biological organisms are desired in designed systems, it is still poorly understood what the crucial mechanisms are to achieve these behaviours and how to model those mechanisms effectively and in ways that are feasible

Martin A. Trefzer, Julian F. Miller and Andy M. Tyrrell are with the Department of Electronics, University of York, UK. e-mail: {mt540, jfm7, amt}@ohm.york.ac.uk

Tüze Kuyucu is with the Department of Information Systems Design, Doshisha University, Japan. e-mail: tkuyucu@mail.doshisha.ac.jp

for artificial systems. Fourth, time and space affect nature and engineering in different ways, thus, they are required to be represented in different ways. For example, a biological cell can indeed afford to keep a fossil record of genes—and it is likely that it even requires it to function—, because, in terms of actual, physical size, the genome is small and, in terms of energy required, it consumes only few resources when passive. This is not generally the case in engineered systems, where both computational power and memory come at a high cost.

Furthermore, when designing artificial developmental systems (ADSs), possibly most important questions are:

- Which entity of the designed system should be represented by one cell?
- What should be the complexity of an artificial cell?
- Is it necessary or advantageous at all to create a complex multi-cellular system when, for instance, a control task can be accomplished using a single cell?
- In nature, cells that are faulty or no longer required simply die and new ones are grown to replace them; in what ways is this ability limited in artificial systems?

According to Wolpert in his book *Principles of Development* [13], biology answers those questions quite clearly: "The development of multicellular organisms from a single cell–the fertilised egg–is a brilliant triumph of evolution". This short phrase expresses one of the fundamentals of natural development, namely the fact that a single cell may arguably be the most complex part of any organism, but a single (or few) cell organism is vastly limited in the tasks it can achieve. Thus, it is more vulnerable to environmental threats, since it cannot afford the loss of a number of cells to the same extent as a multi-cellular organism would, without being harmed beyond recovery. In addition, a multicellular organism offers the capability of multi-tasking via division of labour amongst the cells. Although multicellularity could be merely the result of cell division failure or chance mutation in the evolutionary history of organisms, it has become a key approach harnessed by biology to create complex and intelligent organisms capable of executing sophisticated behaviours and surviving harsh and changing environmental conditions [14].

Multicellular organisms are a product of an either natural or artificial process called development that builds these organisms from a single cell. Wolpert defines development as "The process of gene activity that directs a sequence of cellular events in an organism which brings about the profound changes that occur to the organism" [13]. In other words, development is a sequence of steps where all the genetic information gathered over the history of an organism is put to use to create an adult organism from a single cell. However, development is also a mechanism that maintains the stability and functionality of an organism throughout its lifetime, and not merely the genotype-phenotype encoding mechanism of biology for creating complex multicellular organisms. Thanks to multicellular development an organism is capable of surviving damage and even loss of its physical parts, which otherwise would be lethal to the organism.

Interpreting the biological inspiration in a straightforward manner in order to apply it to systems design would demand building a very large number of highly complex entities (cells), all running the same sophisticated program. In addition, these entities should be expendable, small and able to physically reproduce. The developed organisms can be large, reproduce and die after a time. Time scales of evolution are in the order of millions of years whereas those of development, i.e. the life-spans of organisms, are in the order of decades. From an engineering point of view it seems to be desirable to incorporate a large number of cells into a system in order to achieve scalability and fault tolerance. However, due to resource constraints being entirely different to biological systems, a number of trade-offs need to be made when creating an ADS and many biological mechanisms and entities need to be implemented in different ways. For example, the size of one cell can greatly vary depending on the target architecture. In case of a software simulation, a cell might require memory and processor time to run its program, in the case of an electronic system a cell might be represented by a circuit and in the case of robotics, a cell might represent a single sensor or an entire robot.

A. Investigating Variable Length GRNs: Questions That Arise

Due to the vast and complex nature of ADSs, the problems that arise when designing them can only be solved by the research community as a whole. However, multicellularity is one of the identified key features required to achieve scalability and fault tolerance in systems, and at the same time sufficient complexity of cells needs to be ensured in order to achieve adaptivity, specialisation and the ability to perform all functions required to develop and maintain an organism. Therefore, compact cell programs are desired that achieve a high degree of functionality while providing a small resource footprint. This paper aims to achieve this by introducing a number of genetic representations to achieve variable length GRNs. These GRNs feature small resource footprints and can be processed faster due to their reduced size, while providing efficient encodings for evolutionary optimisation that exhibit a high degree of evolvability. The hypotheses of this work are:

- Variable length GRN representations have no drawbacks in terms of complexity, evolvability and success rate when compared to fixed length ones tested on the task presented.
- Variable length GRN representations feature increased compactness, require less computational effort during optimisation and running development, hence, provide shorter time-to-solution.
- The reduced size makes them more viable for resource critical applications like robotics and embedded systems. It also allows for applications where the target is to encode information in a compact fashion.
- The overall performance and success rates of experiments undertaken on the different test patterns enable assessment of the current implementation of the ADS and identification of potentially missing/ineffective mechanisms.

B. Organisation of the Paper

In Section II, a general overview of currently achieved developmental models in the field of EC is given. The implementation of the ADS used in this work is detailed in

Section III, followed by a description of the different genetic encodings used to achieve variable length GRNs in Section IV.

The experimental setup and the set of test-patterns is described in Section V. The three subsequent sections contain results: in Section VI, the parameters of a fixed length representation (GRN) are optimised, in order to establish a competitive system for the variable length approaches to compare with. Section VII, presents results obtained with the different approaches including a comparison. Section VIII, provides an investigation into the usage and functions of the genes in the GRN for the different approaches.

Following the results, the ADS presented is analysed based on the overall results obtained in Section IX, before concluding the work in Section X.

There is supplementary material to this paper available online on IEEE Xplore[1], which provides tables and graphs showing the entire data gathered from all results, including an analysis of statistical significance, computational effort and success rates of the independent evolutionary runs. Summary tables are provided in the results Sections VI and VII for a better overview and to facilitate accessibility of the results.

II. Models of Artificial Development

The main goal when designing ADSs is to create a system based on principles of biological development that would then automatically inherit desired properties of its biological inspiration, such as scalability, adaptivity, robustness and fault-tolerance. This work is an example where a GRN based model of an ADS is used in a multicellular environment, hence, the focus is set on such systems here. The ability of multicellular developmental systems to be robust, adaptive and scalable makes them interesting to computational intelligence as these properties are difficult to design with traditional approaches to computation or engineering. Biological organisms can grow from a single cell into a multicellular organism using the same genotype for all cells. This offers advantages for implementation, as it is generally easier to design (and scale) tissues of identical structures. In order to carry out specific tasks, cells can then specialise to form different parts of an organism. Although the process of development in biology is clearly defined, its definition in EC varies between different authors [3]–[11]. Whatever the definition, multicellular design approaches in EC benefiting from the decentralised organisational mechanism achieved in biological organisms should bring about scalability, fault tolerance and adaptivity to systems.

As mentioned before, when it comes to implementing an ADS the entire range of approaches can be taken from creating models that comprise a set of basic rules to elaborate mathematical models that capture mechanisms of gene expression, cell functionality and topology or the behaviour of an organism as a whole. There are a number of different approaches that are published in the literature, which cover the whole range.

Some systems model development on the cellular level including cell chemistry, interactions and signalling between cells. Such systems are aimed at investigating both single cells and whole organisms, focussing on mechanisms of growth, cell signalling, cell specialisation, emerging form and behaviour of organisms. All cells in such a system are usually designed to be identical, i.e. they contain the same genome. In the case of this work the genome is represented by a GRN and the time required to process this network once is defined as one *iteration of development* or one *developmental step*. Examples for models of ADSs based on cell chemistry can be found in [3], [6], [15], [16]. Although developmental models that implement some sort of cell chemistry can be regarded as being closer to the biological example, or at least more biologically inspired, they do not necessarily model biological development perfectly. In fact there is much work on this type of artificial development using diverse design constraints; those that closely model biology [5], [8], [17]–[19], those that aim to model biological development in a more simplistic fashion [12], [20], [21], and models that aim to be as close to biology as possible without striving to accurately model it [4], [22]–[25]. Mimicking biology closely should provide a developmental system with high evolvability [26], [27], whereas a simplistic model would reduce the number of complicated processes that exist in biological development, reducing simulation times drastically. Cellular automata (CAs) model biological systems with a grid of cells that determine their states using the local information from their neighbours and simple update rules; in this respect cellular automata (CA) [21] can be regarded as the first and one of the simplest examples of a developmental systems where the cell chemistry is reduced to a few simple rules.

In contrast to that, some developmental systems aim to model biological development at a higher level of abstraction where the focus is set on the overall behaviour of developmental mechanisms, systems and organisms. Unlike in cases where a chemistry or update rules are modelled on the cellular level, the function and behaviour of smaller entities of the system (usually cells in ADSs) is not modelled but rather the response and behaviour of the system as a whole. Consequently, such systems are significantly different from the biological example, but can provide insight into natural phenomena without the computational overhead of modelling complex cell chemistry of a large number of components. The developmental aspect is achieved by including time and iterative behaviour in these models. A famous example are Lindenmayer Systems (L-Systems) [28], which model growth processes of plant development without using any kind of cell chemistry or cell interaction; a parallel grammar rewriting system is used instead, based on a set of rules, thus aiming to imitate biological development of plants using recursive functions. L-systems have also been applied to circuit design problems [29], [30], neural networks [31], and the design of 3D structures [32], [33]. Another example of a grammatical developmental system is cellular encoding (CE) [34]. CE was designed to be used in the design of neural networks. Using CE, a neural net would learn recurrence and solve arbitrarily large parity problems such as a 51-bit parity problem [34]. An example of a non-grammatical developmental system is the self-modifying cartesian genetic programming (CGP) [7]. In this case a CGP program is evolved, which is able to

[1] ieeexplore.ieee.org

modify its own structure over time at runtime, thus creating a developmental system.

The process of designing an effective and efficient artificial developmental model reduces to selecting, simplifying and implementing the right biological mechanisms. This aims to achieve an evolvable developmental system while maintaining a system that is not constrained or overwhelmed by undesirable biological processes. However, it is not obvious which mechanisms and processes are useful and which are not. Implementations of artificial development in evolutionary computation (EC) have been proposed since the early 90's by Fleischer and Dellaert [17], [35]. However, until today, in most examples of developmental models proposed, design decisions are based on educated guesses and various assumptions on the suitability of the biological developmental processes for EC applications, rather than being derived from thorough statistical investigation and understanding of the models achieved so far. The need to investigate the behaviour as well as efficient and effective ways of implementing artificial development is already acknowledged by various researchers [4], [16], [23], [24], [36]–[39]. Probably the most important difference between artificial development in EC and biology is the entirely different embodiment: the physics of the digital medium are entirely different from the organic building blocks of biology. In addition, a biological cell has a number of advantages and disadvantages when compared with a digital cell. For example, an artificial cell is able to copy the genotype perfectly, without any degradation or alteration, from one cell to another whereas biological cells are prone to small errors, particularly with increasing biological age. On the other hand, cell growth in biology naturally occurs by the physics and chemistry of organisms whereas this has to be engineered—usually at a high cost of resources—in designed systems. In summary, the usefulness of a certain biological process or mechanism is not always directly correlated to the embodiment of the artificial system. In the worst case they might even be inversely correlated.

III. Artificial Developmental System

The artificial developmental system presented in this paper is based on a gene regulatory network. Although in the case of this work the ADS is carried out in software on a PC, it is designed with consideration of being suitable to run in embedded and robotic systems. As a consequence, this imposes simplifications on the choice of data types (boolean, integers) and arithmetic operations used (addition, subtraction, modulus). Within the latter boundaries, the selected mechanisms are kept as close as possible to their biological counterparts. Thus, there is no loss of generality caused in terms of operation of the GRN and developmental mechanisms included. The variables of the GRN are represented through chemicals, which are responsible for gene regulation, cell signalling and also for performing functions of the cells and organism.

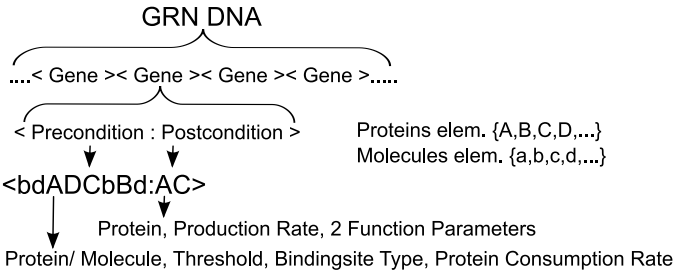

Fig. 2. An example DNA is shown. The GRN comprises a number of genes that are separated by brackets. A gene consists of pre- and postcondition, which are separated by a colon. Each letter refers to one type of chemical and is linked to three integer parameters. Upper-case letters represent proteins, which can be directly produced by the GRN. Lower-case letters represent molecules, which can only be produced in a secondary process, i.e. a gene function. Both proteins and molecules are present in the gene's precondition and contribute to gene regulation whereas only proteins can be directly produced in the postcondition. The way the string of protein types and parameters is interpreted depends on whether the site is located in the pre-conditional part or the post-conditional part of the gene. In the case of precondition, the letter defines which type of chemical can bind to this site. Associated parameters are interpreted as the protein threshold that is necessary to activate the binding site, the type of binding site (inhibitory or excitatory), and the rate at which the binding protein is consumed while the site is active. In the case of postcondition, the parameters are interpreted as the production rate of the respective protein and as parameters for the respective protein function. An overview of the protein functions is shown in table I.

A. Representation and Gene Regulation

In the case of developmental systems and particularly in this case where different genetic representations and encodings are involved, it is necessary to distinguish between the genome that is evolved and the decoded genome, i.e. the GRN, which represents the developmental program. Therefore, the genome and its genes, which are evolved using an evolutionary algorithm (EA), are denoted genome_{EA} and gene(s)_{EA} respectively. In contrast, the genome representing the GRN used in development is denoted GRN and gene(s)_{GRN}. Note that the term *encoding* may be used to refer to genome_{EA}. The process of generating the GRN from genome_{EA} is referred to as *decoding* or *(variable length) mapping*. In turn both genome_{EA} and GRN are referred to as *fixed length representation*, in case the resulting GRN has a fixed length, or as *variable length representation*, in case the resulting GRN has a variable length. Consequently, the steps performed for an experiment are: first, to evolve genome_{EA}. Second, decode genome_{EA} into GRN. Third, run development using the GRN. Finally, evaluate and assign fitness to genome_{EA}.

The core of the proposed developmental model is represented by a GRN, which is executed as shown in Figure 1. GRN is implemented as a string of characters that represent binding sites and gene actions as well as separators between genes and pre-/postconditions, as shown in Figure 2. genes_{GRN} consist of pre- and postcondition, which both comprise a number of binding sites. Each binding site has three integer parameters attached, which represent different things in different contexts: in the pre-conditional case they represent the activation threshold, the binding site type (excitatory, inhibitory) and the protein consumption rate. Note that proteins are only consumed when a binding site is activated, i.e. the

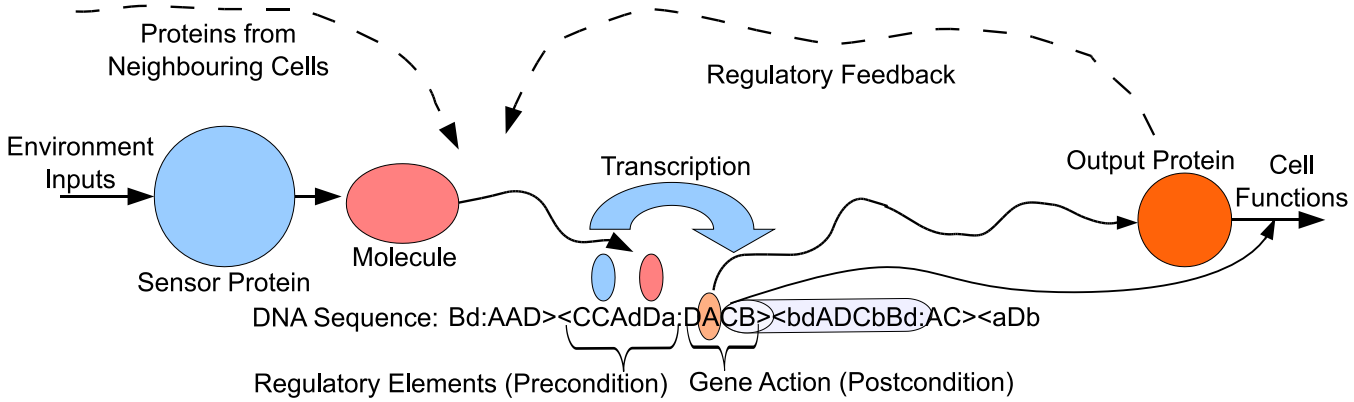

Fig. 1. The genes in an artificial DNA sequence are activated when a sufficient amount of protein is produced by the GRN, which then causes transcription of the respective gene. In the example shown in the figure, the activity of the GRN results in the correct transcription factors (the promoter proteins) that bind to the sites of a gene sequence. This initiates the production of another protein, which can then affect cell functionality using further information that is encoded in the gene. The protein produced also provides feedback to the GRN, which regulates the transcription of other genes, thus achieving dynamic gene regulation. A gene is expressed or inhibited according to the result of the evaluation of the precondition.

TABLE I

THE CURRENT GRN WORKS WITH FOUR TYPES OF PROTEINS AND FOUR TYPES OF MOLECULES. PROTEINS ARE DIRECTLY PRODUCED BY THE GRN AND PERFORM GENE ACTIONS, WHEREAS MOLECULES ARE ONLY PRODUCED AS A RESULT OF A *measurement* THAT IS PERFORMED BY THE *sensor protein*. PROTEINS AND MOLECULES TOGETHER ARE DENOTED AS CHEMICALS. THE ROLES AND FUNCTIONS OF CHEMICALS IN THE ADS ARE DESCRIBED IN THE TABLE. NOTE THAT ALL CHEMICALS TAKE PART IN GENE REGULATION, IN ADDITION TO THEIR SPECIAL PURPOSE, AND AS SUCH ARE CONSUMED WHEN THEY BIND.

Protein/ Molecule	Role (apart from gene regulation)
Structuring (Protein A)	Structuring proteins are used to alter the cell state, represented by an integer value. In this paper, the cell state represents a colour value in the range of $0 \ldots 3$ (white, red, blue, yellow) in order to represent a pattern with a 2D organism. The modulus of the integer representing the cell state by 4 is calculated in order to identify the colour of a cell. Structural changes of cells are made as follows: 1) Get parameters 1 and 2 (p_1,p_2) from genome (see Figure 2). 2) First 5 bits of p_1 determine the number n of bits to read from p_2. 3) The remaining bits of p_1 determine the position where the n bits are inserted into the cell structure.
Sensory/ Interacting (Protein B)	Sensory proteins provide a means to react to changes in the cell state or the environment. Molecules are produced, based on the current cell function, cell location and possible rules that can be encoded in the genes: 1) Get parameters 1 and 2 (p_1,p_2) from genome (see Figure 2). 2) p_1 modulus the number of molecule types used determines which one is going to be produced. 3) p_2 modulus the maximum chemical production rate determines the amount of molecules produced.
Diffusion Layer (Protein C)	Diffusion layer proteins release chemicals to the diffusion layer and absorb them from the diffusion layer. Only chemicals which are released to the diffusion layer are subject to the diffusion process. Diffusion takes place by distributing half of the amount of each chemical on the diffusion layer in equal shares to its 4 von Neumann neighbours (i.e. $\frac{1}{8}$). Chemicals on the diffusion layer need to be absorbed by the cell first, before they affect gene regulation. This mechanism effectively models a simple cell membrane and works as follows: 1) Get parameters 1 and 2 (p_1,p_2) from genome (see Figure 2). 2) p_1 modulus the number of chemical types used determines which one will be released to the diffusion layer. 3) p_2 modulus the current amount of chemical present determines the amount of chemical released.
Plasmodesmata (Protein D)	Plasmodesma proteins provide a mechanism to form Plasmodesmata [40] in order to share their proteins with neighbour cells, and they initiate growth. In case there is a neighbouring cell present it is necessary that both cells offer chemicals to each other in order to form a Plasmodesmata. If a Plasmodesmata is established the cells equally share the sum of both amounts offered. For growth to occur, the cell space towards which chemicals are offered needs to be empty. 1) Get parameters 1 and 2 (p_1,p_2) from genome (see Figure 2). 2) The first 8 bits of p_1 modulus 4 determine which one of the neighbour cells is offered chemical (N, S, E, W). 3) Bits $8 \ldots 15$ of p_1 modulus 4 determine which type of chemical is offered to one of the neighbour cells. 4) p_2 modulus the current amount of chemical present determines the amount of chemical offered.
Molecules (a,b,c,d)	Molecules are produced as a result of the *sensor protein* being expressed and reacting according to the current cell state or environment input. Thus, they only have a regulatory function in the GRN.

concentration of the matching chemical is above the threshold. In the post-conditional case, the parameters represent the production rate of the respective protein and provide two integer parameters p_1 and p_2 as inputs to the associated protein function. Functions of the different proteins are described in Table I. Although the system can principally work with an arbitrary number of chemicals, eight chemicals are currently used: four proteins (A,B,C,D) and four molecules (a,b,c,d). gene_{GRN} regulation occurs as described in Figure 2.

The genome_{EA} is implemented as a list of lists of integers, which represent the genes_{EA} that encode the genes_{GRN}. A gene_{EA} refers to one integer value of genome_{EA} and a minimum of 4 genes_{EA} are required to encode one binding site or gene action of a gene_{GRN} (see Figure 2). Therefore, in order to encode, for instance, a GRN featuring two genes_{GRN} comprising 4 binding sites in the precondition and two gene actions in the postcondition, two lists of 24 integers are required in the genome_{EA}. Furthermore, if the numbers of binding sites and gene actions are not previously known, additional genes_{EA} (integer values) need to be added to those list in order to parameterise the genes_{GRN}. Such encodings enable variable length GRNs.

Binding sites of natural genes, which are defined by upstream and downstream gene sequences (binding sites) that accept proteins to bind and transcribe their genetic code, allow for probabilistic binding as one result of variable length. The probability that certain proteins (transcription factors) bind to the DNA is given by how well the signature of a protein matches the one of the DNA binding site. There are examples of artificial GRNs where probabilistic (smooth) binding is included [6], [41]. However, in this case this feature is not included for simplicity, although it continues to be an open question whether soft binding would bring important benefits to computing and engineering.

B. Cell Signalling and Growth

Wolpert [13] describes three different types of cell signalling (cell communication): protein diffusion, direct contact of complementary proteins on the cell's surface, and gap junctions (equivalent of Plasmodesmata in plants). Two different types of cell signalling are realised in the proposed model. Protein diffusion has been implemented in a similar way as it takes place in physical systems, e.g. a drop of ink dissolving in water. The second cell signalling mechanism is an implementation of Plasmodesmata in plants (protein tunnels) [40], which effectively are tunnels between cells that allow chemical sharing. Cell growth is also based on the Plasmodesmata mechanism; if the neighbour cell, which is targeted by the tunnel, is an empty cell space, a new cell will grow into that empty cell space and become alive (i.e. will be processed by the ADS) in the next developmental step (see Figure 3). Cell death is not implemented at the current stage.

Protein levels are credited and debited after the GRN has processed the next developmental step for all cells. Thus, the GRN always works with the original protein levels and the order of cell update should therefore not bias the behaviour of development or certain regions within the organism. Diffusion is a long range signalling mechanism that should help to create

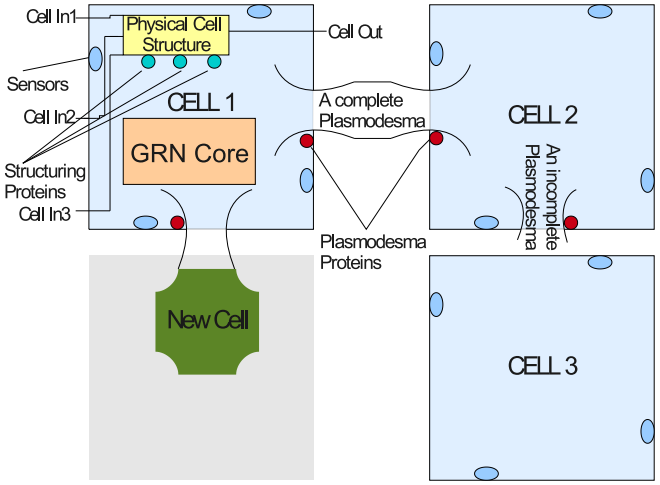

Fig. 3. In a multi-cellular environment using the four basic protein types (ABCD), which are described in Table I, a cell is able to: interact with its environment, replicate (grow), structure itself, and form a complex multicellular organism. The basic functions of some proteins are demonstrated in this figure. Only cell 1 is drawn completely, certain components are omitted in other cells for clarity. In the example, cells 1 and 2 both have active Plasmodesmata proteins, which cause the formation of a channel on both cells towards the other, creating a Plasmodesmata to allow free movement of proteins from one cell to other. Cells 1 and 2 both also have active Plasmodesmata proteins on their southern sides. Cell 1's southern neighbour is a dead cell, so the active Plasmodesmata protein initiates a growth process in that direction. However, cell 2's southern neighbour is an alive cell with no Plasmodesmata protein, thus cell 2 forms an unconnected channel on its southern wall. The direction in which the Plasmodesmata proteins are active, is encoded in the postcondition of the gene, hence, there is one species of Plasmodesmata protein with four different behaviours. The four sensors drawn monitor the outside activity on four sides of each cell and produce different messenger molecules with changing environment. The structuring proteins are produced by the GRN to change the physical structure of the cell, which is connected to the physical inputs and outputs of the cell.

and maintain symmetries within the system.

1) Chemical Diffusion: The ADS is set up to work on an array of 6×6 cells in a von Neumann neighbourhood, i.e. direct cell communication takes place with the 4 orthogonal neighbours. The 6×6 cell tissue which is used to develop patterns is surrounded by inactive cell spaces. Whilst the organism cannot grow into those inactive cell spaces, chemicals can diffuse—but not tunnel—into those places. These inactive border cell spaces are configured to consume all chemicals, hence, chemicals that diffuse outwards are lost. This mechanism makes the environment appear infinite to the organism, although the actual size of the organism is obviously limited. The rule for diffusion is that the 4 neighbours receive $\frac{1}{8}$ of the total amount of each chemical taking part in diffusion, which results in half of the total amount being evenly distributed amongst the four neighbours.

Inspired by natural cells, the model of the cells includes a simple cell membrane. Thus, chemicals first have to penetrate the cell membrane in order to be diffused to neighbouring cell spaces. Chemicals which are outside of cells—in the intra cell space—need to penetrate the cell membrane again in order to take part in gene regulation, i.e. only chemicals inside cells affect the GRN. The types and amounts of chemicals that penetrate the cell membrane in either way, inwards or outwards, are determined by the function of the *diffusion*

protein. This mechanism enables the GRN to control the types and amounts of chemicals that are taking part in diffusion, hence, removes the need to make the arbitrary decision of whether to include diffusion in the system per se or not. It might also help to make a clearer distinction between chemical diffusion and chemical tunnelling, the first one being a longer distance physical effect, the second one being a means of direct cell-to-cell signalling.

2) Chemical Tunnelling: Chemical tunnelling is a contact based cell signalling mechanism, i.e. it takes place only between adjacent cells. Tunnel formation between cells is controlled via the *Plasmodesmata protein*. Only if both cells try to establish a tunnel towards each other, a complete Plasmodesmata (tunnel from cell to cell) is formed. Once a Plasmodesmata is established, the cells can share certain types and amounts of chemicals that are, again, determined via the gene function of the *Plasmodesmata protein*. If the concentration of *Plasmodesmata protein* is too low in either one of the two connected cells, the Plasmodesmata will collapse.

Unlike chemical diffusion, chemical tunnelling is a direct cell-to-cell signalling mechanism controlled by the GRN, which enables the formation of specific signalling pathways through the organism. Hence, although a tunnel between two cells enables short-range signalling, a number of tunnels between different cells can form large, complex chemical signalling networks within the organism. An example for a Plasmodesmata is shown in Figure 3.

3) Organism Growth: The ability of the artificial developmental system to grow is based on the chemical tunnelling mechanism (Plasmodesmata), which is described in Section III-B2. If an existing cell tries to establish a tunnel with an empty cell space, i.e. there is no cell occupying the neighbouring cell space, growth is initiated. A newly grown cell starts its lifetime with all chemical levels set to zero, hence, there are genes required that are by default always expressed in order to supply the cell initially with protein. Although this is an arbitrary choice—it would be possible to start off with a predefined amount of chemicals—it is usually a good idea in evolved systems not to provide evolution with predefined values for parameters that are likely to change in different experiments.

C. Cell Specialisation and Structuring

As the GRN is intended to eventually form a pattern of $2 \ldots 4$ colours, it has to be able to change the specialisation of a cell in such a way that it can represent different colours. This is achieved by the *structuring proteins*. When expressed, these proteins use the encoded action and parameters of the gene (see figure 2) to alter the cell's structure, which is represented by an integer value. This integer value is then used to calculate a colour value for the cell by taking the modulus of the number of desired cell states. The function of the *structuring protein* is described in detail in Table I.

In the current implementation the ADS cells are totipotent, i.e. although the cells 'specialise' in representing a certain value during development, this is never an irreversible state. However, cells which are at all times able to change their specialisation might have an adverse effect on stability of the organism. At the current stage it is still subject to research whether (and how) cell specialisation should be implemented in a reversible or irreversible fashion.

D. Environment Coupling

Key features that are expected from development are self-repair and adaptivity. Due to this, it is crucial that gene regulation can be affected by structural change or damage to enable the GRN to react. In the proposed system this is achieved by introducing sensor proteins that read the cell state and produce second messenger molecules according to rules, which can either be canonical—i.e. four cell states, four molecules—or encoded in the gene action. Experiments presented are carried out using four molecules, which are produced according to a predefined protein function B and information encoded in the GRN. Refer to Table I for a description. The production of the four molecules (abcd) is therefore not dependent on the environment in this case, since inherent pattern formation is studied rather than changes induced by the environment. As in biological cells, the molecules cannot be directly produced by the GRN but play a significant role in gene regulation [13].

IV. Mappings to Achieve Variable Length GRN

Development in biology describes the process where a mature organism is grown from a single cell, the zygote. In nature, every organism features its specific DNA configuration which has evolved over a very long time scale (hundreds of millions of years). The DNA represents the genetic program that is carried out during development. It is interesting to note that evolution of the DNA takes magnitudes longer than the development and lifetime of organisms, which is in the order of decades or even less. Therefore, artificial developmental systems too comprise two stages, namely the evolution of the DNA and the developmental process that creates the final organism or system by processing the regulatory network formed by the genes (and chemicals) encoded in the DNA.

This section introduces different mechanisms of the genome_{EA}-to-GRN mapping for the evolution of ADSs. There are essentially two classes of mapping mechanisms, those which produce fixed length GRNs and those which produce variable length GRNs. Variable length is defined separately for the length of the GRN, i.e. the number of genes_{GRN}, and for the length of the genes, i.e. number of binding sites and gene actions of a single gene. Hence, the following Sections IV-A and IV-B discuss two different kinds of encodings: firstly, to achieve variable and fixed length *GRNs* and secondly, to achieve variable length genes_{GRN}. Both kinds of mapping mechanisms can be combined in different ways in order to achieve variable length on all levels, e.g. variable number of genes_{GRN} in the GRN/ fixed length genes_{GRN}, fixed number of genes_{GRN} in the GRN/ variable length genes_{GRN}. Combinations used in this paper are listed at the beginning of Section VII. The more complex mappings provide properties that may aid evolutionary search, such as increasing amounts of parts of the genome that are neutral towards consequential mutation due

to larger numbers of (redundant) genes [42]. However, this might also be a drawback, since particularly in the case of development where computational complexity is already high, such mappings impose further increase in computational effort. Related work that includes variable length GRN based ADSs can be found in [43], [44], although the variable length aspect is not in the centre of research focus in these papers.

A. Mappings to Achieve Variable Number of Genes$_{GRN}$ in the GRN

This section describes different representations of the genome$_{EA}$ and genome$_{EA}$-to-GRN mapping mechanisms that allow for a variable number of genes$_{GRN}$ in the resulting GRN. In this section, a gene$_{GRN}$ is represented by a list of genes$_{EA}$, which are represented by integer values. Therefore, the genome$_{EA}$ is effectively a 2-D genome: dimension one defines the maximum number of genes$_{GRN}$ and dimension two defines the size of the genes$_{GRN}$, as indicated in Figure 4. The representation and mapping of the genes$_{EA}$ to genes$_{GRN}$ are introduced and discussed in Section IV-B, as those are effectively separate, additional mechanisms. In other words the translation of a genome$_{EA}$ to a GRN comprises two stages: first, it is determined which subset of genes$_{EA}$ will be selected to create the GRN and in which order they will occur. Second, only those genes$_{EA}$ are then translated into genes$_{GRN}$, resulting in the final GRN.

1) Direct Translation (Fixed Length GRN): In the case of direct translation all genes$_{EA}$ are selected to create the GRN in the same order as they occur in the GRN. Hence, a list of genes$_{EA}$ represents one gene$_{GRN}$ in the resulting GRN. Each list of genes$_{EA}$ contains (at least) as many integers as necessary to encode a single gene$_{GRN}$. An example is shown in Figure 4, *Direct translation*. Note that this mapping results in a fixed length GRN, which contains all genes possible. However, when this method is combined with a variable length genes$_{EA}$ to gene$_{GRN}$ mapping, the resulting GRN as a whole will still vary in length although the number of genes$_{GRN}$ is fixed.

2) Active/Inactive Flag (Variable Length GRN): A basic mechanism to achieve variable length genomes$_{ADS}$ is to introduce an additional gene$_{EA}$ into each list of genes$_{EA}$, which serves as a flag to determine whether a particular list is active or not: in the first case, the remaining genes$_{EA}$ in the list are decoded into a gene$_{GRN}$. In the second case, the remaining genes$_{EA}$ in the list are skipped. As a consequence, the EA can control which genes$_{EA}$ are represented in the GRN via this additional gene$_{EA}$, thereby varying the number of genes$_{GRN}$ that form the GRN. Hence, this method allows the EA to produce GRNs of variable length, although when combined with a fixed length genes$_{EA}$-to-gene$_{GRN}$ mapping the number of genes$_{GRN}$ varies but they will be constant in length. Note that this method requires one more integer per encoded gene, resulting in a larger genome$_{EA}$. An example is shown in Figure 4, *Encoding no of sites*.

3) CGP Mapping (Variable Length GRN): A more advanced way of encoding the GRN is to use an indirect encoding as it is used, for instance, in cartesian genetic programming (CGP) [45]. In this case the genome$_{EA}$ consists of a number of nodes (lists), which contain two genes$_{EA}$ (integers) that encode the inputs to the node and a list of genes$_{EA}$ that encode the gene$_{GRN}$. The decoding process of the genome$_{EA}$ takes place in two stages: in the first stage, the genome$_{EA}$ is parsed starting with the last node (output) and then recursively determining the set of active nodes by following the node numbers of the encoded inputs. This process will yield a directed graph, which consists of the active (visited) nodes of genome$_{EA}$. These active nodes are then ordered according to the prefix notation of the directed graph obtained, resulting in a list of nodes. In the second stage, the CGP header, i.e. the two genes$_{EA}$ that encode the inputs are stripped from the nodes, leaving the lists of genes$_{EA}$ that encode genes$_{GRN}$. Subsequently, the resulting genome$_{EA}$ is decoded in the same straight forward fashion as described previously in Section IV-A1. An example is given in Figure 4, *CGP mapping*.

Using this kind of encoding should provide all the advantages of CGP representations, such as increased evolvability, neutrality [45], [46] and therefore the ability to sample the search space more effectively, when evolving the genome$_{EA}$. Additional benefits of this—and also the *Active/Inactive Flag* encoding—might be the fact that certain properties of nature are captured better. For example the fact that the rate of consequential mutations is generally low due to neutrality in the genome, which results in increased degeneracy of the genome$_{EA}$. This might enable mechanisms like gene duplication, as it occurs in nature.

TABLE II
THE GRAMMAR THAT IS USED FOR MAPPING genome$_{EA}$ TO GRN. THE SYMBOL ∅ STANDS FOR 'IGNORE SYMBOL AND MOVE TO NEXT ONE'.

Grammar	$G = (V, \omega, R)$	
Alphabet	$V = [,], G, <, :, >, \_, X, P, M, a \ldots h, A \ldots H$	
	$[,]$	delimiter for entire GRN
	G	symbol for a gene
	$<, :, >$	delimiter for genes (start, pre-/postcondition, end)
	$\_$	symbol for precondition binding sites
	X	symbol for postcondition sites
	P, M	symbols for proteins and molecules
	$a \ldots h$	one of eight specific molecules
	$A \ldots H$	one of eight specific proteins
Axiom	$\omega = [G]$	
Production Rules	R	
	$[,]$	$\varnothing$
	G	G, $< \_ : X >$, $G < \_ : X >$, $< \_ : X > G$
	$<, :, >$	$\varnothing$
	$\_$	$\_$, P, $P\_$, $\_P$, M, $M\_$, $\_M$
	X	X, P, PX, XP
	M	$a \ldots h$
	P	$A \ldots H$

B. Variable Length Genes$_{GRN}$ / Binding Sites

The mapping algorithms that are introduced in this section represent different approaches to decoding single genes$_{GRN}$, rather than the entire GRN. In combination with the mapping algorithms presented in the previous Section IV-A, these mechanisms enable the evolution of variable length GRNs, where

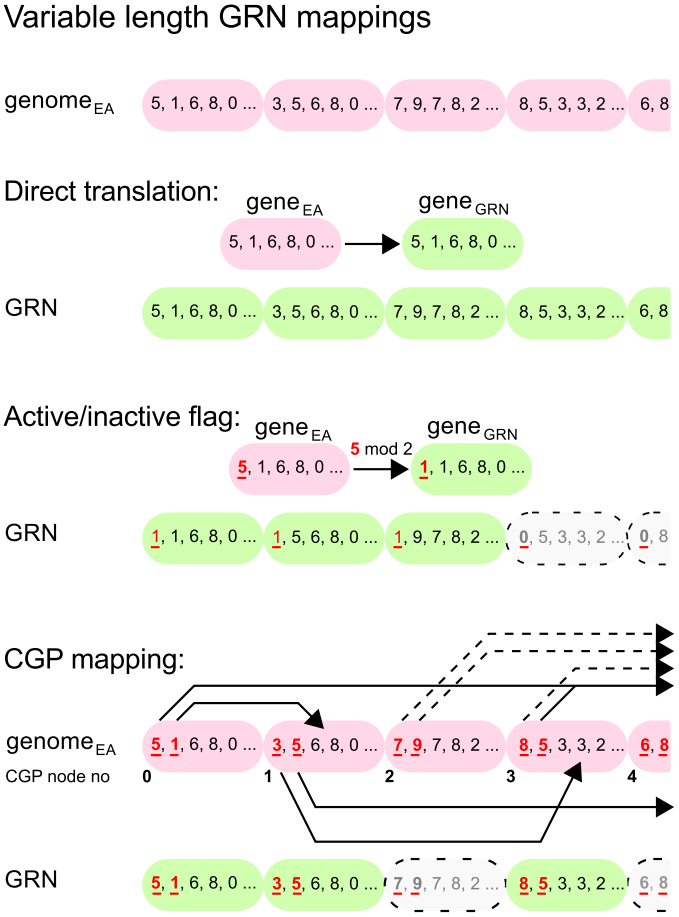

Fig. 4. Mechanisms to achieve variable length GRNs are shown. In the case of *direct translation*, the genome$_{EA}$ is copied to GRN without changes. In the case of *active/inactive flag*, the first gene$_{EA}$ in each list of genes$_{EA}$ determines whether the other ones in the list are decoded into a gene$_{GRN}$ or whether they are skipped. In the *CGP mapping* the first two genes$_{EA}$ of each list represent CGP inputs. Only lists of genes$_{EA}$ that are connected in the CGP tree are decoded into genes$_{GRN}$. Thus, the latter two mechanisms enable evolution to control the length and content of GRN on the number of genes$_{GRN}$ level.

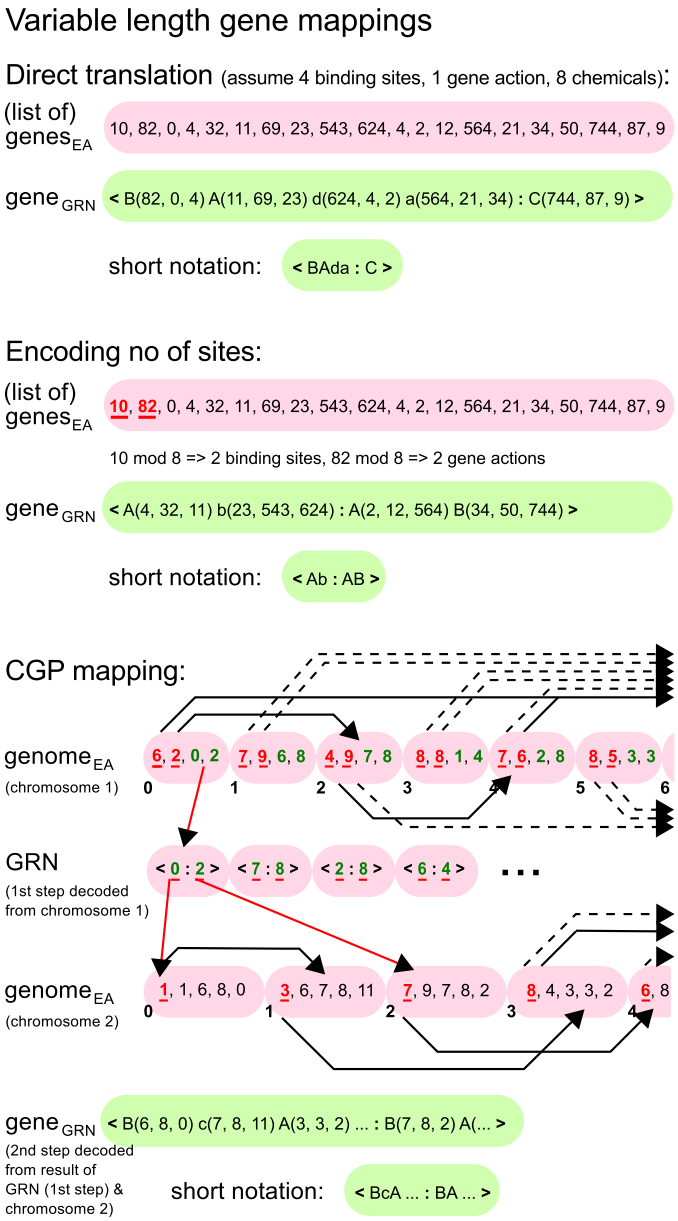

Fig. 5. Mechanisms to achieve variable length genes$_{GRN}$ (variable number of binding sites and gene actions) are shown. In the case of *direct translation*, the number of binding sites and gene actions is fixed and groups of 4 genes$_{EA}$ are directly translated into binding sites or gene actions. The meanings of the various types of genes$_{EA}$ are detailed in Figure 2. In the case of *encoding no of sites* there are two extra genes$_{EA}$, which determine the number of binding sites and gene actions. In the case of the *CGP mapping* the genome$_{EA}$ consists of two chromosomes: the first one determines which genes$_{GRN}$ will be decoded in exactly the same fashion as described in the *CGP mapping* in Figure 4. The second chromosome encodes pre-conditions and post-conditions of the genes$_{GRN}$, again, via a CGP representation. The numbers in the resulting genome from the 1st decoding step genes$_{GRN}$ point to different nodes in the second chromosome. In terms of standard CGP those would be the output numbers. The CGP nodes in the second chromosome represent single binding sites and gene actions, similar to a "sea of chemicals". If the number of nodes is large (in the order of 2000-4000) a large number of different CGP trees can be obtained when starting from different nodes. Writing the different CGP trees obtained down in pre-fix notation provides strands of binding sites and gene actions, which can then be assembled to form the genes$_{GRN}$. Thus, the genes of the first chromosome are interpreted as different "entry points" into the second chromosome, in order to extract different pre-conditions and post-conditions. Note that the 1st decoding step works in a similar way than the *CGP mapping* in Figure 4. It has been included here again for clarity.

both the number of genes$_{GRN}$ and the length of the genes$_{GRN}$ is variable. In the case of the precondition of a genes$_{GRN}$, variable length means a varying number of binding sites. For example, this enables evolution to create genes$_{GRN}$ that are always expressed simply via reducing the number of binding sites to zero. On the other hand genes$_{GRN}$ can be formed which are more likely to be inhibited by inserting a large number of binding sites. In the case of the postcondition, variable length refers to the number of types of proteins produced, this also defines the number of gene$_{GRN}$ actions carried out, which are associated with the different types of proteins.

1) Direct Translation (Fixed Length Genes$_{GRN}$): In the case of direct translation the list of genes$_{EA}$ (integers) is decoded into a gene$_{GRN}$ in a straight forward fashion. The number of genes$_{EA}$ in each list depends on the desired number of binding sites in the precondition and the desired number of gene$_{GRN}$ actions in the postcondition. In both cases 4 integer values are required for encoding a binding site or gene$_{GRN}$ action. In the case of decoding a binding site, integers are decoded as follows: the first integer is interpreted as the type of the site,

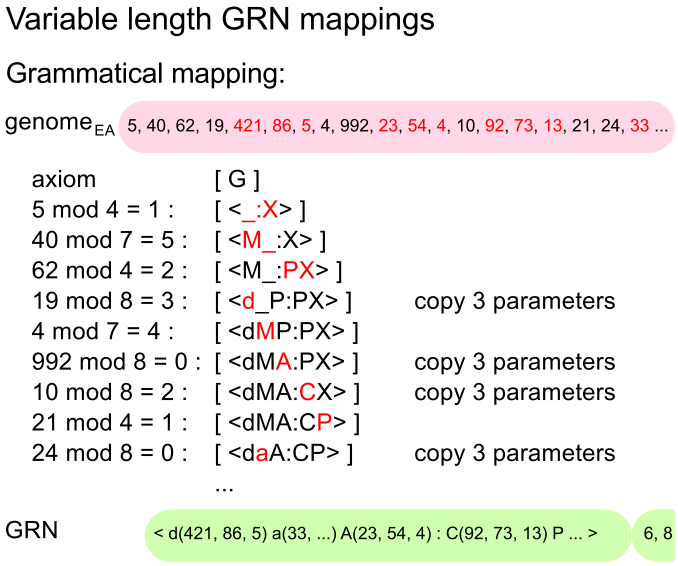

Fig. 6. A *grammatical mapping* is used to decode the genome$_{EA}$. This is a special case, since variable length GRN and genes$_{GRN}$ are achieved at the same time. The genes$_{EA}$ are used to select which rule of the grammar from Figure II to apply.

i.e. it determines the chemical that binds to it. The second integer determines the concentration threshold that must be exceeded before binding can take place. The third integer determines the function of the binding site, i.e. inhibitory, excitatory or neutral. Whenever a chemical binds, its concentration decreases. Hence, the fourth integer determines the chemical consumption rate in the latter case. In the case of determining a gene$_{GRN}$ action of the postcondition, the genes$_{EA}$ are decoded as follows: the first integer determines the type of protein that is produced and the second one encodes the production rate. The third and fourth integers represent input parameters to the function that implements the gene$_{GRN}$ action, which is associated with each type of protein produced. The functions of the different proteins are described in Section III-A and Table I.

In the case of fixed length genes, the number of binding sites and gene actions is predefined and fixed. Thus, the number of genes$_{EA}$ in a list of genes that encode a gene$_{GRN}$ equals $4 \times (\#\text{bindingsites} + \#\text{geneactions})$. The structure of the GRN is shown in Figure 2, and a decoding example is given in Figure 5, *Direct translation*.

2) Encoding No of Sites (Variable Length Genes$_{GRN}$): A simple way of achieving variable length genes is to add dedicated genes to genome$_{EA}$ (similar to the encoding from Section IV-B1), which are decoded into numbers that define the size of precondition and postcondition of each decoded gene$_{GRN}$. Hence, the length of each list of genes in genome$_{EA}$ increases by 2 and genome$_{EA}$ increases by $2 \times \#\text{genes}_{GRN}$. As a consequence, the length of the decoded genes$_{GRN}$ can vary between 0 and the maximum number of binding sites and gene actions encoded in one list of genes of genome$_{EA}$. In case the length is decoded as a small number (for example between 0 and 4), the number of unused genes per list increases. This provides an additional amount of neutrality in genome$_{EA}$, due to the increasing number of unused genes$_{EA}$ at the ends of the aforementioned integer lists. An example is given in Figure 5, *Encoding no of sites*.

3) CGP Mapping (Variable Length Genes$_{GRN}$): In Section IV-A3, CGP representation is used for genome$_{EA}$ in order to achieve a variable length GRN, where each CGP node contains the genetic code (a list of genes$_{EA}$) required to decode one gene$_{GRN}$. In this case, the genome$_{EA}$ consists of two chromosomes: the first one determines which genes$_{GRN}$ will be decoded. This works in the same way as described in the *CGP mapping* in Figure 4. The second chromosome encodes pre-conditions and post-conditions of the genes$_{GRN}$ as described in Figure 5. The numbers in the 1st step of decoding genes$_{GRN}$ point to different nodes (entry points) in the second chromosome. The CGP nodes in the second chromosome represent single binding sites and gene actions similar to a "sea of chemicals". If the number of nodes is large (in the order of 2000-4000) a large number of different CGP trees can be obtained when starting from different nodes. Writing the different CGP trees obtained down in pre-fix notation provides strands of binding sites and gene actions, which can then be assembled to form the different pre-conditions and post-conditions of the genes$_{GRN}$. Since each of the extracted CGP trees from the second chromosome will have different lengths, variable length genes$_{GRN}$ are achieved.

Each node in the second "sea of chemicals" chromosome can be re-used in a number of genes$_{GRN}$. Each chemical (CGP node) in this pool contains four integer values, one that encodes the next CGP input and four to encode a binding site or gene$_{GRN}$ action. In order to yield smaller CGP graphs, there is only one input per CGP node in this case. The information that needs to be encoded in the genes$_{EA}$ is now reduced to two pointers (integers) to nodes within the second chromosome, one for the pre-condition and one for the post-condition. Note that both binding sites and gene actions are extracted from the second chromosome. In the first case the nodes are decoded in pre-condition mode, i.e. they are decoded into proteins and molecules. In the latter case the nodes are decoded in post-condition mode, i.e. they are decoded into proteins only in accordance with the GRN model described in Section III.

Note that in both cases *CGP Mapping (Variable Length GRN)* and *CGP Mapping (Variable Length Genes)* the common term *CGP Mapping* is used. This is done because both mechanisms are essentially based on the same CGP decoding mechanisms, but are distinguished by their context: the mechanism from Figure 4 works on the GRN level whereas the one from Figure 5 works on the gene level. In Section VII it is described which mechanisms exactly are used in the experiments.

C. Grammatical Mapping (Variable Length GRN & Genes$_{GRN}$)

The grammatical genome$_{EA}$ to GRN mapping is an exception to the previously described mappings, because it encompasses both variable number of genes$_{GRN}$ and variable length genes$_{GRN}$ at the same time. The decoding is realised via a grammatical rule re-writing system, pioneered in [47]

and similar to a Lindenmeyer System (L-System) [28], which produces genomes$_{ADS}$ that comply with the format used here (see Figure 2). The grammar employed is shown in Table II. In this case genome$_{EA}$ is not divided into lists or nodes, the grammatical mapping algorithm simply interprets it as a single unformatted stream of integers which are decoded into appropriate rules. The grammar is described in Table II and an example decoding is shown in Figure 6, *Grammatical mapping*.

The grammatical approach is intended to provide a true alternative to the other ones, as it is the only mapping that operates on an unstructured genome$_{EA}$. However, in this case there is no guaranteed neutral space for evolution in the genome$_{EA}$, which might render evolutionary optimisation more difficult.

V. Development of Patterns via Variable Length GRNs

This section describes the experimental setup used, including EA parameters, configuration of the ADS and the set of test patterns. In addition the statistical methods that are used to evaluate the results presented are provided.

A. Set of Test Patterns

The set of test patterns for this work has been chosen according to examples that are widely used in the related literature [4], [6], [12], [12], [23], [25], [28], [29], [48]–[52] and which provide a variety of characteristics and symmetries in order to test different aspects and capabilities that are expected from developmental systems. For example self-organisation, specialisation and modularisation. An overview of the 11 test patterns is shown in Figure 7. The patterns used can be classified as *mosaic patterns*, *border(ed) patterns* and *patch patterns*. As can be seen from Figure 7, there is a further class of *symmetric flag patterns*. This kind of classification follows the tradition in this area of research, however, it would be more beneficial to classify those patterns according to their symmetries as indicated in the results Section IX, in Table IX. All patterns used are of the size of 6×6 pixels, where each pixel is represented by a cell.

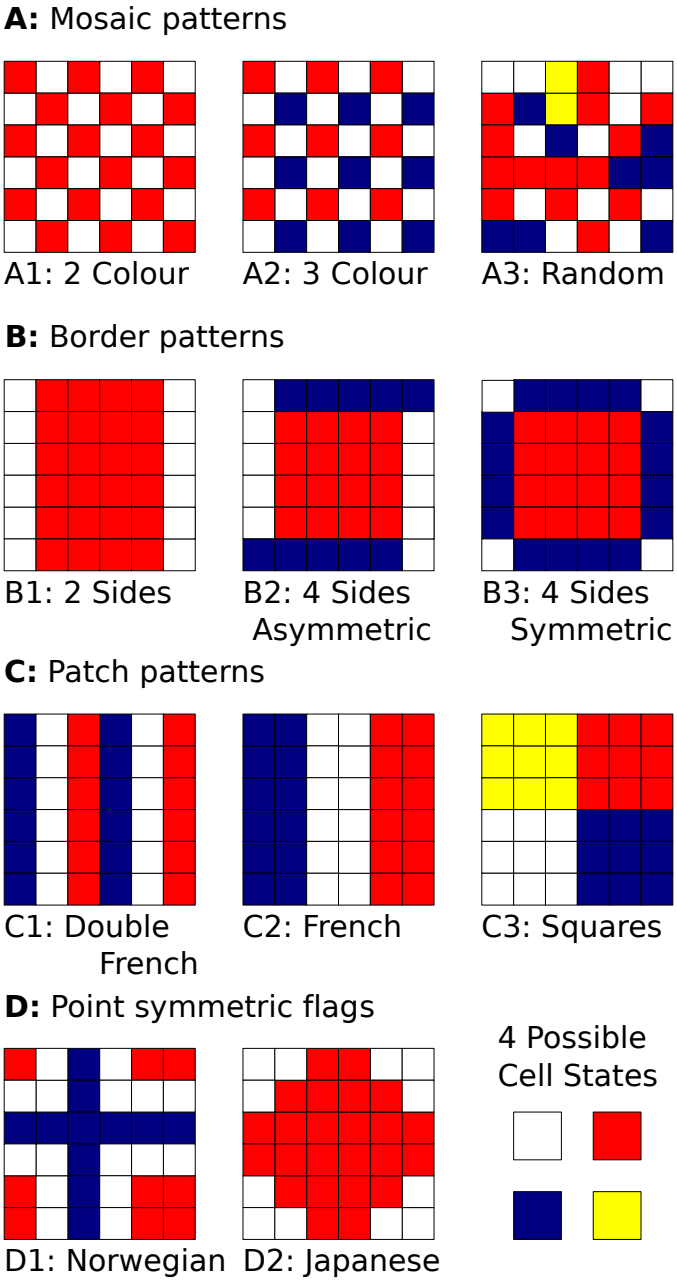

Fig. 7. The set of test patterns used for development is shown in the figure. The naming of the different patterns follows the tradition in the research field, however, the important distinctive features of the patterns show are their different symmetries.

B. Evolutionary Algorithm and Parameters

The genome$_{EA}$, which encodes the genes$_{GRN}$, is optimised using a $(1+4)$ evolutionary strategy (ES). Hence, the population size is 5, one elite individual from which 4 offspring individuals are created in each generation. If one (or more) of the offspring individuals achieves equal or better fitness than the current elite, that one (or one of those) is promoted as the new parent (elite). An ES has been chosen to be consistent with previous work where its design was optimised for operation on a hardware platform [24], [49]. An overview of the parameters, genotype-phenotype mappings and genome$_{EA}$ resources used is given in Table III.

C. Configuration of the ADS & Fitness Evaluation

In order to develop different patterns, the ADS is set up to work on an array of 6×6 cells. Each cell state represents one pixel of each pattern. Even though the GRN can consist of a variable number of variable length genes, one iteration of development (or one developmental step) is defined by sequentially processing all genes of the entire genes$_{GRN}$ once.

As described in Section III-C, the GRN develops a structural part of the cell, which is represented by an integer value. The cell state is determined by calculating the modulus of this integer value by the number of possible cell types. In the case of patterns, the number of cell types corresponds to the number of colours used. Since there are a maximum of 4 colours used for the experiments in this paper, the ADS is configured to produce 4 cell types (excluding the empty cell space, which

TABLE III
OVERVIEW OF THE GENOTYPE SIZES OF THE DIFFERENT EXPERIMENTS AND THE SETUP OF THE EVOLUTIONARY STRATEGY; VAR = VARIABLE LENGTH, FIX = FIXED LENGTH, GRAM = GRAMMATICAL.

GRN $genes_{GRN}$	fix fix	fix var actions	fix var genes	var fix	var var genes	var var actions	var gram	gram CGP	CGP CGP
no $genes_{GRN}$	60	60	15,30,60,120	0..60	0..60	0..60	0..1000	0..180	0..180
no proteins	4,6,8	4	4	4	4	4	4	4	4
no molecules	0,2,4	4	4	4	4	4	4	4	4
no binding sites	4,6,8	8	8	8	0..8	8	0..1000	0..8	0..1800
no actions	1	1,2,4,8	1	4	0..4	0..4	0..1000	0..4	0..1800
$genome_{EA}$ size	240,330, 420	420,480, 600,840	420,840, 1680,3360	1740	1860	1800	5000	5940	11,520
mutation rate	1% of genome size								
population size	1+4 evolutionary strategy								
max. generations	200,000								

is not actually a cell state). Also for patterns that consist only of 2 colours, the number of possible cell types is still kept at 4 in order to keep the complexity of the organism the same.

In all experiments a single cell in the middle of the organism is initialised with all chemical levels set to zero at the beginning of development. The GRN is then carried out for 30 developmental steps with the aim to find the closest match to the target pattern in any of those steps. The maximum number of 30 developmental steps is arbitrarily chosen, however, results from previous work [24], [49] suggest that this should be enough for the ADS to find solutions at least for the simpler patterns. Furthermore, previous work shows that development can stabilise in the case of the ADS presented; but this is not within the focus of this paper.

The fitness is calculated by counting the number of cell states that do not match the target pattern, hence, the best fitness is 0, which means the pattern is perfectly achieved, and worst fitness 36, which corresponds to the case where the states of none of the cells match. Target cell states are enumerated as shown in Figure 7. An evolutionary run is considered to be successful, if it achieves a fitness value of 0 in at least one iteration of development. Thus, it is not predefined in which particular iteration the solution should be found.

D. Statistical Analysis of Results

In order to analyse and assess the performance of the experiments presented, a variety of statistical methods are employed: first, histograms that depict the success rate of each set of independent evolutionary runs are provided for each experiment. Second, box-and-whisker plots are shown for the respective subsets of successful runs. Since the resulting distributions are highly skewed, non-parametric measures are used, namely median, and upper/lower quartile ranges into which 50% of the samples fall. The dashed lines indicate a region which equals to $1.5 \times$ interquartilerange. Samples outside these whiskers are considered as outliers. The notches around the median show the confidence interval into which 5% of the samples fall. These plots provide information on the distribution of the number of generations required to find a solution thus, providing more detailed information on the performance of an approach tested: the lower the median number of generations and the smaller the interquartile ranges and the confidence intervals, the better the performance. Note that the box-and-whisker plots are less meaningful in cases when the success rate is low (< 20), i.e. there are not enough samples in the distribution. However, for completeness all data is included. Third, a rank-sum test is performed in order to obtain a p-value which evaluates the statistical difference of two experiments/approaches. A 5% significance level is used for all the rank-sum tests presented, hence a p-value of $< 5\%$ indicates a significant difference in two results being compared. Fourth, in order to further analyse the scientific significance, Vargha-Delaney statistics is performed on the results, which delivers an A statistic that represents a measure of effect size compared to stochastic noise [53]. This test is independent of the sample size and has an output value range of 0-1. A value of 0.5 indicates identical performance between the two samples, values smaller or larger than 0.5 suggest increasingly large effect sizes. For example, a value larger than 0.5 indicates a better performance for the first of the two samples, an A statistic greater than 0.64, or less than 0.36, indicates a "medium" or "large" effect size [53], which is considered to be scientifically significant. Fifth, in order to obtain information of how long the optimisation process would take in practice, the computational effort (CE), which has been introduced by Koza [54] is also calculated. In addition, the upper and lower bounds of the CE are included in order to indicate the significance of the CE. Those bounds are calculated according to the extension to the CE introduced in [55].

VI. OPTIMISING NO GENES, CHEMICALS AND ACTIONS FOR FIXED LENGTH

As a first step three series of experiments are carried out using fixed length representations where the number and size of $genes_{GRN}$ are fixed. The absolute number of genes, chemicals and gene actions are varied across different experiments. The experiments are aimed at finding optimal values for the latter parameters in the case of fixed length GRNs, which are within ranges similar to those used in the variable length representation experiments. This allows for a more fair comparison between variable and fixed length representations. Acronyms used in the graphs consist of letters: P = proteins, M = molecules and A = gene actions/proteins produced. Each

letter is followed by a number determining the number of types of proteins and molecules, as well as the number of gene actions/types of proteins produced. In cases where the number of $genes_{GRN}$ is variable a number of *nodes* is specified, which determines the (maximum) number of $genes_{GRN}$.

Note that in the following sections tables that summarise the results are provided for a better overview. Tables and graphs showing detailed results and statistical data can be found in supplementary material provided online on IEEE Explore (ieeexplore.ieee.org). In order to present the results in a condensed and concise manner, ∘ and • symbols are used to indicate whether an approach performs better, or significantly better than another one. The more ∘ and • symbols a certain approach obtains, the better is its overall performance compared with the other ones. The • symbol indicates significantly better performance than one of the other approaches, i.e. an A value greater than 0.64 accompanied by a p-value of less or equal to 5%. The ∘ symbol indicates better performance than one of the other approaches, i.e. an A value greater than 0.6 accompanied by a p-value of less or equal to 5%. In a few cases a p-value of less than 10% is accepted, based on a high success rate of a certain approach in finding solutions.

A. Varying Number of Genes

In the first series of experiments, the number of $genes_{GRN}$ in the GRN is set to $15, 30, 60$ and 120 nodes. Four types of proteins (A,B,C,D) and four types of molecules (a,b,c,d) are used. Accounting for the number of chemicals, the number of binding sites is set to 8 and the number of gene actions is set to 1. The results are summarised in Table IV.

In this experiment, the *Border 2 Sides*, *Mosaic 2 Colours*, *Mosaic 3 colours*, *Patch French* and *Patch Japanese* patterns achieved high success rates and feature significant values of the A statistic. As can be seen from Table IV, setting the number of $genes_{GRN}$ to 30 or 60 yields the best results with 60 being the optimal choice in most cases. The results suggest that either 15 $genes_{GRN}$ might not be enough to solve the problem, or the number of 15 $genes_{GRN}$ is not enough for evolution to optimise the GRN. Note that there are two mechanisms that can turn $genes_{GRN}$ on or off: first, evolution can simply turn on or off $genes_{GRN}$ permanently by creating according pre-conditions (binding site configurations). Second, $genes_{GRN}$ can be expressed (turned on) or inhibited (turned off) during development both permanently, in case the system settles into a stable state (attractor), or temporary, in case the system is in transition, due to changing chemical levels. Hence, if there are not enough neutral or redundant $genes_{EA}$ for evolution to perform non-destructive mutation, effective search will be impeded. On the other hand, as the performance decrease in the case of 120 nodes indicates, the exponential increase in search space size will become too costly, even if there is the possibility of further increasing the success rate when running evolutionary search for longer. Even though the computational effort (CE) will not significantly increase when using 120 genes, the CPU time still increases in practice due to the increased model size.

B. Varying Number of Chemicals

In the second series of experiments, the number of protein and molecule types available to the system are varied. 50 independent runs are carried out with the following configurations: 4 proteins and 0 molecules (4 chemicals in total), 6 proteins and 0 molecules (6 chemicals in total), 8 proteins and 0 molecules (8 chemicals in total), 4 proteins and 2 molecules (6 chemicals in total), 4 proteins and 4 molecules (8 chemicals in total). The number of $genes_{GRN}$ actions is set to 1. Based on the results from Section VI-A, the number of genes is set to 60 in these experiments.

In this case, the *Border 2 Sides*, *Mosaic 2 Colours*, *Mosaic 3 Colours*, *Mosaic 3 Stripes*, *Patch French* and *Patch Japanese* patterns achieved success rates high enough to deliver meaningful A values in the statistical analysis. As can be seen from the results shown in Table V, it is difficult to draw a general conclusion in this case. For the *Border 2 Sides*, *Mosaic 2 Colours* and *Mosaic 3 Colours* having only 4 proteins is beneficial, whereas the number of molecules does not seem to play a significant role. In the case of *Mosaic Stripes* and *Patch French* there is no significant difference in performance.

Reasons for this may be that either 4 chemicals are already sufficient to solve the pattern problems, or significantly more than 8 chemicals would be required in the system to gain measurable performance. Going below 4 proteins cannot be tested with the current implementation of the ADS, since proteins ABCD are carrying out system relevant functions. Thus, a minimum of 4 proteins is required for the ADS presented to work. Experiments with a much larger number of chemicals are omitted at this point as there is no indication for significant benefit, such as being able to better solve one of the more complex patterns or reduce computational effort; in fact, CE is entirely uncorrelated with the success rates and numbers of chemicals for all patterns. Only in the cases of *Border 2 Sides* and *Mosaic 2 Colours*, there is a clear trend that CE and runtime increases when using more types of chemicals. The fact that the experiments with these two patterns achieve almost 100% success rate hints that this trend might be valid in general, although the results do not suffice to draw a definite conclusion. Due to this, the number of protein and chemical types is set to 4 respectively, resulting in a total of 8 chemicals for all other experiments in this paper.

C. Varying Number of Gene Actions

In the third series of experiments, the number of $gene_{GRN}$ actions is varied. Four series of 50 independent runs are carried out for each pattern, where the number of $gene_{GRN}$ actions is set to 1, 2, 4 or 8. According to the findings from Sections VI-A and VI-B, the number of protein types is set to 4, the number of molecule types is set to 4 and the number of genes is set to 60.

As can be seen from the summarised results in Table VI, like in the previous experiments, the ADS achieves good performance for *Border 2 Sides*, *Mosaic 2 Colours*, *Mosaic 3 Colours*, *Mosaic Stripes*, *Patch French* and *Patch Japanese*. The results indicate that the optimal setting for the number

TABLE IV

THE TABLE PROVIDES A SUMMARY OF THE RESULTS FOR VARYING THE NUMBER OF GENES ACROSS A SERIES OF EXPERIMENTS USING FIXED LENGTH REPRESENTATION. IN ORDER TO PRESENT THE RESULTS IN A CONDENSED AND CONCISE MANNER, ○ AND ● SYMBOLS ARE USED TO INDICATE WHETHER AN APPROACH PERFORMS BETTER, OR SIGNIFICANTLY BETTER THAN ANOTHER ONE. THE MORE ○ AND ● SYMBOLS, THE BETTER IS THE OVERALL PERFORMANCE.

Pattern	Number of Genes				Successful Runs			
	15	30	60	120	15	30	60	120
Border 2 Sides		●	●●○	●	50	49	49	46
Border 4 Sides Asymmetric		○	○○	●○	20	20	38	29
Border 4 Sides Symmetric		○	○	○	15	26	26	18
Mosaic 2 Colour		●	●●	●●○	50	50	50	50
Mosaic 3 Colour		●	●	○	48	49	48	47
Mosaic Stripes		●	●	○	35	42	37	31
Patch 4 Colour			●		12	15	17	7
Patch French		●	●●	●	34	39	43	36
Patch Japanese		●	●●	●	38	48	48	48
Patch Norwegian	—	—	—	—	4	3	2	4
Patch Random	—	—	—	—	9	12	13	10

TABLE V

THE TABLE PROVIDES A SUMMARY OF THE RESULTS FOR VARYING THE NUMBER OF CHEMICALS ACCROSS A SERIES OF EXPERIMENTS USING FIXED LENGTH REPRESENTATION. IN ORDER TO PRESENT THE RESULTS IN A CONDENSED AND CONCISE MANNER, ○ AND ● SYMBOLS ARE USED TO INDICATE WHETHER AN APPROACH PERFORMS BETTER, OR SIGNIFICANTLY BETTER THAN ANOTHER ONE. THE MORE ○ AND ● SYMBOLS, THE BETTER IS THE OVERALL PERFORMANCE.

Pattern	Number of proteins and molecules					Successful Runs				
	4P0M	6P0M	8P0M	4P2M	4P4M	4P0M	6P0M	8P0M	4P2M	4P4M
Border 2 Sides	●●●●	●		●○		50	50	48	50	50
Border 4 Sides Asymmetric						25	13	5	16	20
Border 4 Sides Symmetric		○○				13	7	6	8	16
Mosaic 2 Colour	●●	●		●●	●●	50	48	48	50	50
Mosaic 3 Colour	●●			●●	●●	50	29	24	47	48
Mosaic Stripes			●○			40	43	34	40	35
Patch 4 Colour	—	—	—	—	—	10	2	2	10	12
Patch French						36	45	38	35	34
Patch Japanese	●●●●			○○		34	43	36	40	38
Patch Norwegian	—	—	—	—	—	3	3	3	3	4
Patch Random	—	—	—	—	—	4	4	2	6	9

TABLE VI

THE TABLE PROVIDES A SUMMARY OF THE RESULTS FOR VARYING THE NUMBER OF GENE ACTIONS ACCROSS A SERIES OF EXPERIMENTS USING FIXED LENGTH REPRESENTATION. IN ORDER TO PRESENT THE RESULTS IN A CONDENSED AND CONCISE MANNER, ○ AND ● SYMBOLS ARE USED TO INDICATE WHETHER AN APPROACH PERFORMS BETTER, OR SIGNIFICANTLY BETTER THAN ANOTHER ONE. THE MORE ○ AND ● SYMBOLS, THE BETTER IS THE OVERALL PERFORMANCE.

Pattern	Number of gene actions				Successful Runs			
	4P4M1A	4P4M2A	4P4M4A	4P4M8A	4P4M1A	4P4M2A	4P4M4A	4P4M8A
Border 2 Sides		●	●●	●	50	50	50	49
Border 4 Sides Asymmetric			○		20	28	32	27
Border 4 Sides Symmetric		○	●	●	15	18	21	22
Mosaic 2 Colour		●	●●	●●	50	50	50	50
Mosaic 3 Colour		●	●	●	49	48	49	44
Mosaic Stripes			●○	○	35	40	38	25
Patch 4 Colour			●●	○	11	13	13	10
Patch French		●	●	●	35	39	42	35
Patch Japanese		●	●●	●●	39	48	49	45
Patch Norwegian	—	—	—	—	4	1	4	3
Patch Random			○		9	13	13	9

of gene actions is 4 in this case. Due to this, the number of gene actions is set to range from 0 to 4 for the variable length experiments in Section VII.

Surprisingly, the CE decreases in all cases when increasing the number of gene_{GRN} actions. This suggests that it is more beneficial to have genes_{GRN} in the system that are able to carry out a number of tasks. It would be interesting to see whether: first, the reason for this is the fact that in the case of 4 proteins—remember that only proteins occur in the post-condition and perform gene_{GRN} actions—, where each protein controls a viable function of the ADS; and second, whether performance would decrease when using a much larger number of protein types. However, if single genes_{GRN} are able to perform a large number of tasks, there is a danger that evolution will solve the problem by evolving only a few genes that are always active resulting effectively in a one-to-one mapping, rather than producing a dynamic system with the desired behaviour. One can argue that in case evolution can easily solve a given problem with a static mapping, this would be a an advantage and should not be deliberately excluded. To a certain extent this is a valid argument, particularly in the case of developing static patterns. However, in this case the aim is to answer the question whether variable length representations of GRNs are superior to fixed length ones. Therefore, mechanisms that potentially bias the ADS too much towards static solutions are being avoided. On the other hand, these experiments are aimed to find (at least locally) optimal parameters for the fixed length case and to determine reasonable bounds for the variable length experiments.

VII. Comparison of Different Methods to Achieve Variable Length GRN

In this section a range of investigations are undertaken using different setups to create variable length GRNs, based on the methods described in Section IV. Methods for variable length encoding of genes_{GRN} are combined with methods for encoding variable length GRNs in different ways, in order to create the following 9 experimental setups:

1) fixed length GRN (see Section IV-A1) + fixed length genes_{GRN} (see Section IV-B1).
2) fixed length GRN (see Section IV-A1) + variable number of gene_{GRN} actions via encoding no of sites (IV-B2).
3) fixed length GRN (see Section IV-A1) + variable number of both binding sites and gene_{GRN} actions via encoding no of sites (see Section IV-B2).
4) variable length GRN via active/inactive flags (see Section IV-A2) + fixed length genes_{GRN} (see Section IV-B1).
5) variable length GRN via active/inactive flags (see Section IV-A2) + variable number of gene_{GRN} actions via encoding no of sites (see Section IV-B2).
6) variable length GRN via active/inactive flags (see Section IV-A2) + variable number of both binding sites and gene_{GRN} actions via encoding no of sites (see Section IV-B2).
7) variable length GRN and variable length genes_{GRN}, both binding sites and gene_{GRN} actions, via grammatical encoding (see Section IV-C).
8) variable length GRN via CGP encoding (see Section IV-A3) + variable number of both binding sites and gene_{GRN} actions via encoding no of sites (see Section IV-B2).
9) variable length GRN via CGP encoding (see Section IV-A3) + variable number of both binding sites and gene_{GRN} actions via CGP encoding (see Section IV-B3).

TABLE VII
Explanation of acronyms used for the different setups shown.

Acronym	Description
fix GRN	The GRN length (number of genes_{GRN}) is fixed using *direct translation* as shown in Figure 4.
var GRN	The GRN length (number of genes_{GRN}) is variable using *active/inactive flag* as shown in Figure 4.
fix genes	Numbers of binding sites and gene actions are fixed using *direct translation* as shown in Figure 5.
var actions	Number of binding sites is fixed and number of gene actions is variable using *encoding no of sites* as shown in Figure 5.
var genes	Numbers of binding sites and gene actions are variable using *encoding no of sites* as shown in Figure 5.
GRM-GRM	Both number and length of genes_{GRN} are determined by the grammatical mapping described in Figure 6 and Table II.
CGP-CGP	Both number and length of genes_{GRN} are determined by the *CGP mappings* described in Figures 4 and 5.
CGP-varG	The number of genes_{GRN} is determined by the *CGP mapping* described in Figure 4 and the numbers of binding sites and gene actions are determined using *encoding no of sites* as shown in Figure 5.

The meanings of the labels of the different experiments are explained in Table VII

For setups $1-6$ the maximum number of genes_{GRN} is set to 60, the maximum number of binding sites (pre-condition) is set to 8, and the maximum number of gene actions is set to 4. The actual values are evolved and range from 0 to their respective upper limits. In the case of the CGP based encodings used in setups 8 and 9, the maximum number of genes_{EA} (CGP nodes) is set to 180 in order to account for the fact that these representations inherently require a larger number of nodes to work effectively. These choices are made based on publications on CGP [45] as well as observations that are discussed in Section VIII, in order to balance the average number of genes_{GRN} that are used to form the GRN in the context of the different variable length GRN encodings. As can be seen from Figure 8, this has successfully been achieved as the medians of the numbers of genes_{GRN} are similar for all variable length GRN setups. In the case of the grammatical encoding in setup 7, there is no straight forward method to constrain the number of the resulting genes_{GRN} to a certain range. Hence, the size of the genome_{EA} is based on an estimate in this case. The results shown in Figure 8 indicate that the choice of genome_{EA} might have been slightly too conservative (too few genes), since the median of the number of resulting genes_{GRN} is slightly lower in the case of the grammatical encoding.

Furthermore the list of experiments is not exhaustive, since there would be further possible combinations when using all variable length GRN methods and all variable length

gene methods that are introduced in Section IV. However, in order to keep the paper focussed on its two main aims, namely to assess what advantages and disadvantages variable length GRNs bring and to suggest a portfolio of potentially useful approaches for encoding such GRNs, the selected range of experiments is biased towards the more straight forward mappings. Moreover, it can be argued whether it makes sense to compare a method that simply switches $genes_{GRN}$ on/off against more complex encodings like CGP or grammatical encodings; it is most certainly problem dependent as well as target architecture dependent which encoding would be the most suitable. For example, in the case of using a PC in an entirely result-driven fashion, i.e. CE would not be major concern, a CGP encoding would probably be the method of choice due to its proven capabilities of neutral search and convergence. In case the result is required within a tight time limit, or even an embedded system is targeted, straight forward implementations would offer a better trade-off.

As can be seen from the success rates shown in Table VIII and as observed in previous experiments in Section VI, the best performance is achieved in the cases of *Border 2 Sides*, *Mosaic 2 Colours*, *Mosaic 3 Colours*, *Mosaic Stripes*, *Patch French* and *Patch Japanese*. However, in terms of success rate, no approach can be identified that features a significantly better performance in all test cases. There is also no obvious correlation between CE and complexity of the encoding used, except for experiments 7 and 8 where the CE is significantly increased, particularly in the case of *Border 2 Sides*, *Mosaic 3 Colours*, *Mosaic Stripes*, *Patch French* and *Patch Japanese*. Moreover, the success rate is considerably lower in those cases when compared with experiments $1-6$ and 9. The reason for this is most likely the increased CE that is required because of the more complex mappings used. This does not necessarily mean that the mappings used in experiments 7 and 8 are generally worse than the other ones, but it does indicate that the increased complexity is a drawback in the case of the problem class tackled, namely the development of static patterns of different symmetries.

It might be concluded that variable length representations—and the increased complexity of the encoding that comes with them—are not necessary in this case. On the other hand, it has to be kept in mind that the parameters for the fixed length representation are the optimised ones from Section VI. In addition, the results show that the variable length encodings ($4-6$ and 9) perform equally well as those using fixed length encodings, despite the increased complexity and the computational overhead. This makes them competitive, since their major advantage of producing more compact GRNs can be fully exploited. An analysis of the immediate consequences of the latter property is detailed in Section VIII and broader conclusions are made in Section X.

VIII. Statistics on the Lengths of Genes and Protein Usage of the Different Approaches

An important feature of the variable length encodings explored in this paper is that the same functionality and performance can be represented with a smaller GRN model

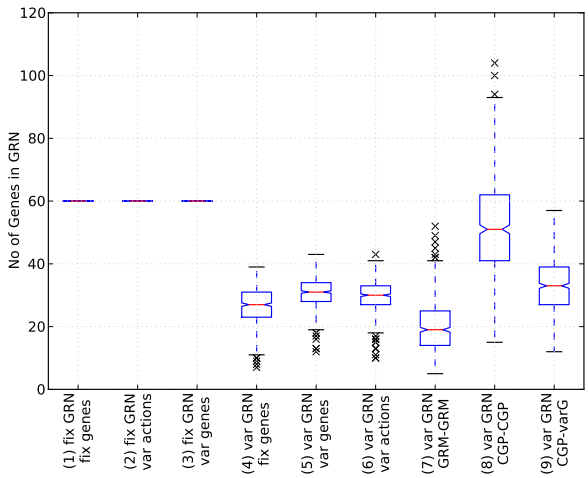

Fig. 8. Number of genes in GRN.

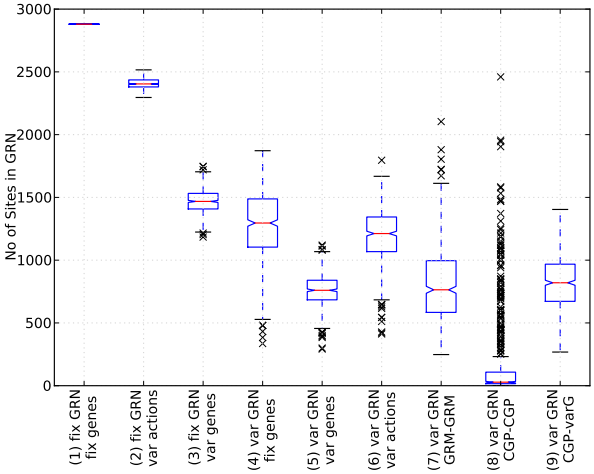

Fig. 9. Number of sites in pre-/postconditions of genes.

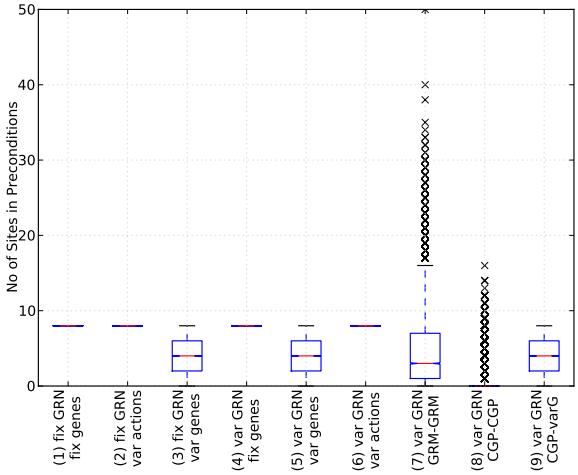

Fig. 10. Number of sites in preconditions of genes.

TABLE VIII

THE TABLE PROVIDES A SUMMARY OF THE RESULTS OBTAINED FOR THE DIFFERENT REPRESENTATIONS USED FOR THE EVOLUTION OF THE GRN. IN ORDER TO PRESENT THE RESULTS IN A CONDENSED AND CONCISE MANNER, ○ AND ● SYMBOLS ARE USED TO INDICATE WHETHER AN APPROACH PERFORMS BETTER, OR SIGNIFICANTLY BETTER THAN ANOTHER ONE. THE MORE ○ AND ● SYMBOLS, THE BETTER IS THE OVERALL PERFORMANCE. DIFFERENT MECHANISMS TO CREATE GRNS

Pattern	Experiment Setups								
	1 fix GRN fix Gen.	2 fixGRN var Act.	3 fixGRN var Gen.	4 var GRN fix Gen.	5 var GRN var Act.	6 var GRN var Gen.	7 var GRN Grammar	8 var GRN CGP-CGP	9 var GRN CGP-v.Gen.
Border 2 Sides	●○			●●○	○○	○		●●	●●●●●
Border 4 S Asym.				●	○	●			
Border 4 S Sym.					○				
Mosaic 2 Colour	●○○	●●●○○	●●●	●○	●			●●●●●●○	●●
Mosaic 3 Colour	●	●	●	●○	●●○	●○		●	●○
Mosaic Stripes	●●●○○○	●	○	●●○	●●○	●		●	●○
Patch 4 Colour	●○○	●●	○						
Patch French	●	●○○	●	●●●●●●	●	●		○○○ ●○	
Patch Japanese	●●○○○	●	●○	●○○	●●○○	●○			●
Patch Norge	—	—	—	—	—	—	—	—	—
Patch Random	●	●	○	●		○			●
	Successful Runs								
Border 2 Sides	50	50	48	50	49	50	50	45	50
Border 4 S Asym.	32	21	25	24	28	37	3	0	28
Border 4 S Sym.	21	14	15	16	15	23	2	0	18
Mosaic 2 Colour	50	50	50	50	50	50	50	18	50
Mosaic 3 Colour	48	46	45	47	50	24	24	29	49
Mosaic Stripes	39	35	26	31	36	5	5	2	40
Patch 4 Colour	12	10	8	9	11	1	1	0	14
Patch French	42	35	32	35	40	10	10	2	43
Patch Japanese	49	44	38	48	49	7	7	1	49
Patch Norge	4	1	5	1	2	3	0	0	1
Patch Random	13	12	14	11	9	1	1	1	10

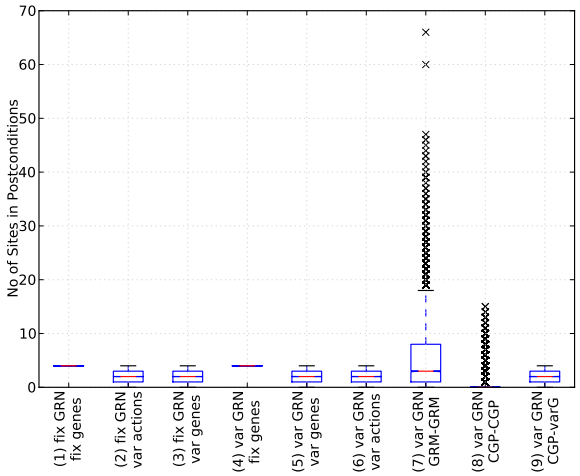

Fig. 11. Number of sites in pre-/postconditions of genes.

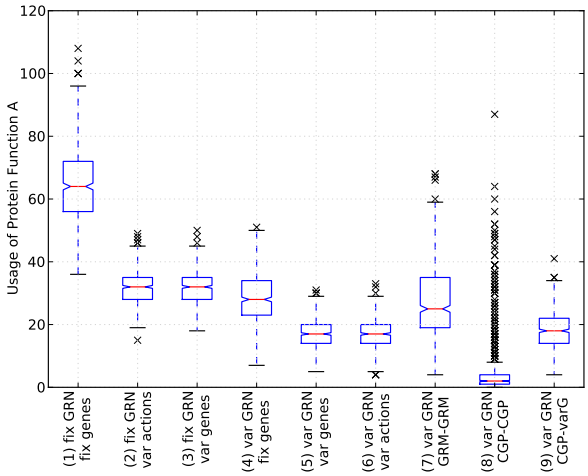

Fig. 12. Usage of protein A and its function.

and thus, using less resources. Whilst this argument is more obviously valid in the case of embedded developmental systems, more compact models also make a significant difference when running a multicellular ADS on a PC. For example, let us assume that the evaluation of a GRN with 60 genes$_{GRN}$ requires $5\mu s$ on a Core2 running at $2.6GHz$. For simplicity, it is assumed here that genes$_{GRN}$ are fixed length (8 binding sites, 1 gene action). The numbers used here are based on runtimes observed from the *Border 2 Sides* experiments from SectionVI-A, which represent one of the easier and faster to develop cases. The evaluation of a population of 5 individuals, each developed for 30 steps in a 36 cells environment results in processing the GRN $5 \times 30 \times 36 = 5400$ times. For the *Border 2 Sides* pattern the median of number of generations required to find a solution is about 1000. Hence, the time required to find a solution would be about 27 seconds. If the

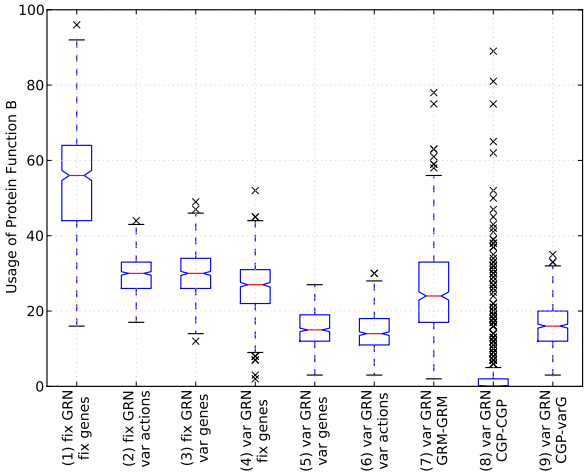

Fig. 13. Usage of protein B and its function.

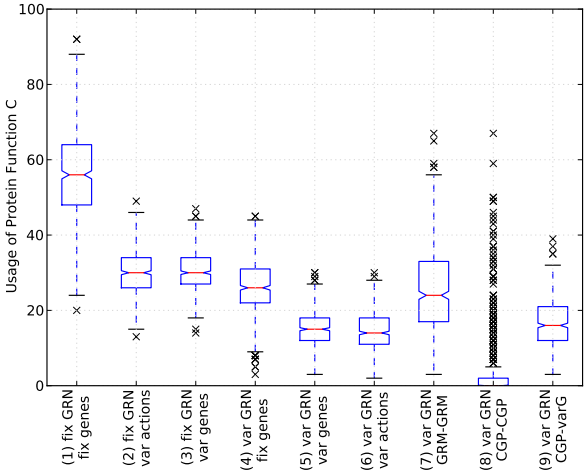

Fig. 14. Usage of protein C and its function.

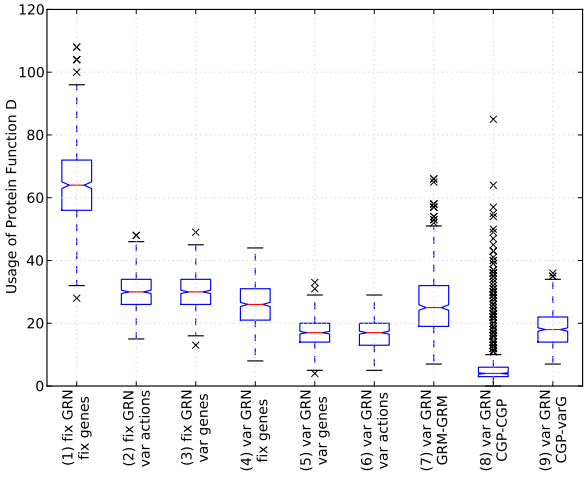

Fig. 15. Usage of protein D and its function.

same functionality could be encoded in a smaller GRN of, for instance, 15 genes without decreasing the evolvability of the ADS, the time required would drop to about 6.75 seconds. In the case of an experiment comprising 50 evolution runs, the time required would be just above 22 minutes in the first case, and only about 6 minutes in the second case. For harder problems, the numbers of generations required can easily be 50-100 times higher. In this paper the worst case would be the 200,000 generations cut-off, which is 200 times higher (resulting in an EA runtime of about 73 hours, as opposed to about 18 hours). There is a small time overhead when decoding complex representations, however, the overall time consumption is dominated by the ADS, rather than the EA.

In addition to the runtime, more compact models offer a smaller memory footprint. This argument, of course, usually only applies to embedded systems. Nevertheless, the memory requirements can be drastically reduced. For example, a fixed length GRN with 60 fixed length genes$_{GRN}$ (8 binding sites, 1 gene action) would require at least $60 \times (8+1) \times 4 = 2160$ integer numbers to store the model. In the case of a 32-bit processor, this would equal to about 8 kByte. Note that in the current implementation of the ADS, 4 integers are required to encode one binding site/gene action. If the number of genes was reduced to 15, without any drawbacks, only a fourth of the space would be required. The reduced memory footprint only applies to the GRN model and not to the encoding used by the EA during optimisation.

In order to analyse the resource consumption of the different representations introduced in Section IV, the number of genes$_{GRN}$ used, the genes$_{GRN}$ lengths and the frequency of usage of certain protein functions are gathered from the experiments and presented in Figures 8-15. The data is separated according to the different representations used and the resulting GRNs for all patterns are incorporated respectively.

As can be seen from Figures 8 and 9, both number and size of the genes$_{GRN}$ is significantly reduced in the case of all variable length representations. Thus, the aim of achieving more compact representation is reached. It is interesting to see that in the case of experiment 8, where the multi-chromosome CGP representation is used, the GRNs tend to comprise a larger number of genes$_{GRN}$ which are, however, much smaller than in the other cases. On the one hand this shows that this representation favours larger numbers of, on average, smaller genes$_{GRN}$. On the other hand, this indicates the requirement to optimise the parameters of the encoding in this case and explains why the performance of this approach was not optimal. It is likely that this encoding is sensitive to the number of provided CGP nodes, particularly since there are two nested CGP representations that are strongly dependent on each other.

As can be seen from Figures 10 and 11, the CGP and grammatical encodings feature a wider range of sizes of pre- and post-condition of the genes$_{GRN}$. This is simply due to the way in which the different experiments are set up; in experiments $1-6$ and 9, maximum sizes for pre- and post-condition are specified. However, it is advantageous that this is not one of the parameters in the system which is greedily exploited by evolution by always promoting maximum length, which is expected in the case of a dynamic developmental system,

where there are multiple mechanisms for gene expression.

It is interesting to see that the frequency of usage of the four different proteins (ABCD) and their associated functions are uniform within each experiment. In between experiments the absolute numbers do vary, but this is simply scaling due to the differences in length of the GRNs produced. From this it can be concluded that at the stage of evolution, there is no bias towards certain functions of the ADS. This suggests that the ADS's mechanisms of gene inhibition/expression determine the behaviour of the system, rather than merely an evolved one-to-one mapping of cell states achieved by extensive use of structuring protein A.

IX. Assessment of ADS and Patterns

When looking at the results of all experiments presented, the patterns used for development can be ranked according to the overall success rate achieved by the ADS. The result of this ranking is listed in Table IX. The overall ranking is consistent with observations made in previous sections where it is found that the patterns which can be most successfully developed are *Border 2 Sides*, *Mosaic 2 Colours*, *Mosaic 3 Colours*, *Mosaic 3 Stripes*, *Patch French* and *Patch Japanese*. Note that it is not considered how closely the patterns were matched in non-successful runs, i.e. there is a possibility that the ADS achieved a smaller number of perfect matches (successes) for a particular pattern, but still was able to develop a closer match than it was achieved in one of the higher ranked ones.

The question arises whether these patterns have any common features or symmetries, which work in favour of the ADS used. Furthermore, there might also be common properties in the ones that are less successfully developed. As can be seen from Table IX, it is not only the number of possible cell states that makes a particular pattern harder or easier to develop, but also its symmetries. It appears that the ADS is most successful in producing linear periodic patterns, such as *Border 2 Sides*, *Mosaic 3 Stripes* and *Patch French*, as well as superpositions of two orthogonal, linear periodic modulations, such as *Mosaic 2 Colours* and *Mosaic 3 Colours*. In addition, the fact that *Patch Japanese* features a high success rate confirms the expectation that circular patterns should be achievable relatively easily due to the diffusion mechanism (the initial cell is (3,3), (1,1) being the top left corner). However, it is then surprising to see that the performance decrease is quite significant in the cases of *Border 4 Sides Asymmetric* and *Border 4 Sides Symmetric*. This suggests that a circular periodic modulation is more difficult to achieve by the ADS, which is also supported by the fact that the success rate is low in the case of the *Patch Norwegian*. Furthermore, the low success rates in the case of *Patch 4 Colour* hints that the ADS lacks ability to establish specialised, spatially separated regions. In biological terms this could be regarded as the ability to self-organise and potentially enable formation of organs in more complex organisms. The *Patch Random* was originally intended as a reference measurement, since it is expected to be a hard problem for an ADS to match a particular random pattern without any regularities. It is now surprising to see that *Patch Random* only seems to be the second last ranked and *Patch Norwegian* appears to be the hardest problem. However, this may simply be due to the particular random pattern chosen.

There are two issues with the current implementation of the ADS that may thwart the ADS in forming circular modulations and spatially separated, specialised regions: first, the fact that the cell mechanisms operate in a von Neumann neighbourhood, i.e. two developmental steps are required for information to reach all surrounding cells. This is a potential drawback when the task is to establish a certain pattern in a confined area like, for instance, the blue cross in the *Patch Norwegian*. Second, the cells are totipotent. This means that every single cell in the organism has the potential to grow the entire organism at any time, when put into a suitable environment. In other words there are no irreversible steps occurring during development as, for example, irreversible cell speciation, although this mechanism might benefit the ADS when the task is to develop stable organs. Irreversible cell speciation is not included in the current system since, from an engineering point of view, it can be argued that this would mean allocating resources which then could not be freed and re-used. One way to overcome the latter drawback is to implement apoptosis, (programmed) cell death. Including the latter mechanisms would, however, considerably increase the complexity of the ADS.

In conclusion, the results show that the ranking of the patterns corresponds to the results found by others [4], [6], [12]. Suggested questions for future work would be to investigate whether the two mechanisms mentioned above, namely using a Moore neighbourhood and including more complex cell speciation/apoptosis, would improve the performance of the ADS on the patterns that are identified to be the more challenging ones.

X. Conclusion

An investigation into different encodings for variable length GRN based ADSs is presented in this paper. The mechanisms of the ADS used are introduced and its design considerations are motivated and discussed with regard to biology and evolutionary computation. The class of problems tackled is the development of patterns with different organisational properties on a 6×6 cell array. The set of patterns comprises 11 different patterns including those which are widely used in the field of artificial development. The focus of the work was set on the introduction of a variety of different approaches to achieve variable length GRNs and compare them with a tuned fixed length approach in terms of ability to perfectly match a pattern, the time required to find a solution and resource consumption. Resource consumption can be further divided into model size achieved (GRN length) and computational effort required to run the model and find a solution. Furthermore, the frequency with which proteins and their associated functions were represented in the resulting GRNs has been analysed. Finally, the different patterns have been classified and ranked according to their overall success rate achieved when developed with the ADS used. Based on this data, the ADS was assessed and potentially missing mechanisms and features have been identified.

TABLE IX
OVERVIEW OF THE OVERALL SUCCESS RATES FOR THE DIFFERENT PATTERNS. MAXIMUM SCORE (= TOTAL NUMBER OF EXPERIMENTS) 1100.

Rank	Pattern	Symmetry	Overall Solutions Found
1	Border 2 Sides	y axis, centre point	1085
2	Mosaic 2 Colour	x,y periodic 1st order	1064
3	Mosaic 3 Colour	x,y periodic 2nd order	959
4	Patch Japanese	x,y axis, centre point	889
5	Patch French	y periodic	776
6	Mosaic 3 Stripes	y periodic	729
7	Border 4 Sides Asymmetric	centre point	489
8	Border 4 Sides Symmetric	x,y axis, centre point	338
9	Patch 4 Colour	centre point	222
10	Patch Random	none	203
11	Patch Norwegian	x,y axis, off centre point	60

A. Optimisation of the fixed length representation.

In order to provide both an optimised fixed length approach for comparison with the variable length ones and to identify reasonable value range boundaries for number of genes$_{GRN}$, genes$_{GRN}$ length, and number of chemical types a series of experiments have been carried out where those parameters are varied. It has been found that a larger number of chemicals tends to increase the number of generations required to find a solution, hence, increased the complexity of the ADS. However, it was interesting to see that using a larger number of molecule types helped to improve the success rate, while in the case of more protein types the success rate drops. This is interesting to observe, since introducing more molecule types provides the ADS with a means to form more complex preconditions, whereas in the case of introducing more protein types the increased complexity does not benefit the success rate. Remember that unlike the proteins, which are directly produced by the genes$_{GRN}$, the molecules only occur in the preconditions of the genes$_{GRN}$ and can only be produced as a secondary product of the GRN via protein function B. It is further observed that increasing the number of genes$_{GRN}$ in the fixed length representation generally tends to improve the performance of the ADS. In practice, of course, the feasible improvement is bounded by increased processing time requirements of a larger number of genes$_{GRN}$. The same applies to increasing the number of actions (proteins produced) per genes$_{GRN}$.

B. Comparison of the different mappings, and statistics on the resource consumption.

As the experiments undertaken show, varying the number of genes$_{GRN}$, the number of chemical types and the number of gene$_{GRN}$ actions—hence, effectively changing the length of GRN and genes$_{GRN}$—, has a major impact on the performance of the ADS. A series of experiments using a variety of encodings of the GRN were carried out in order to answer the question whether it would be beneficial to pass control over the shape and length of the GRN to the EA. The results obtained suggest that, apart from minor variations due to varying computational overhead of the different encodings, the ADS exhibits similar performance for all encodings used. Although the variable length encodings do not feature better performance when merely looking at the success rates, they do provide important advantages; the runtime of the EA is significantly reduced, since almost 100% of optimisation time is consumed by running the ADS multiple times. Processing a smaller number of genes$_{GRN}$ in each run of the ADS, while still retaining a sufficiently evolvable (large) genome$_{EA}$ directly reduces processing time, hence, enables tackling larger problems in the same amount of time. In addition, smaller resulting GRNs provide the same functionality in a more compact form, which will be desirable when moving towards larger, more complex problems. A smaller GRN also means reduced resource consumption, particularly in the case of embedded systems and robotic systems which fall into the scope of targeted applications for artificial development. Note that the reasons for the reduced success rate in the case of grammatical encoding and nested CGP encoding are most likely that those significantly more complex mappings would require more thorough optimisation of their parameters and extended runtime of the EA.

C. Classification and ranking of the patterns. Assessment of the ADS.

Using a set of static patterns featuring different properties has been adopted as a test problem for ADSs by many researchers in the field [4], [6], [12], [15], [16], [22]–[25]. In this paper a set of patterns has been chosen that includes all prominent patterns in the literature and represents an example where a large set of different patterns is used. Important is that each pattern used requires a different, unique cellular organisation, providing different test cases for the ADS. In most cases proposed ADSs are only tested on a subset of 4-5 of the set of patterns used here, and in some cases the ADSs are configured in such a way that it would obviously work in favour of the selected subset. Thus, as a contribution to the area of research into artificial development, which aims to create novel ADSs and improve existing ones, the results in this work are used to create a ranking of the patterns used according to their complexity. This ranking, including the overall success rates achieved in 1100 independent runs for each pattern, might be suitable for others in the field to compare their results against. Patterns that are particularly hard to develop have been identified and, due to their symmetries, revealed two potential shortcomings of the current implementation of the ADS, namely the absence of (irreversible) cell speciation and

cell death to compensate for this, as well as working in a von Neumann neighbourhood of cells which imposes potentially unfavourable delays when communicating with the diagonal nearest neighbours.

D. Conceptual conclusions and benefits of the ADS.

The aim of this paper has been to investigate the potential benefits and the effects of variable length GRNs on the evolutionary performance of an existing developmental system. We have observed a number of advantages when undertaking the experiments in this paper; using variable length representations reduces the number of arbitrary, a priori design choices that have to be made when creating the computational model of development. In this case, for instance, using variable length genes and GRN removes the requirement of pre-defining the number of genes, the size and shape of binding sites and—in the case of the CGP encodings—the order of the genes.

Furthermore, this paper focusses upon achieving variable length on the evolutionary level, rather than including mechanisms that insert and remove genes during the developmental process. Genetic representations are used for evolution, which are then decoded into variable length GRNs that are carried out during development. The advantage of taking the latter approach is that unused genes on the evolutionary level can be regarded as truly neutral to mutation, whereas it cannot be guaranteed that a gene is never going to be expressed during development under any circumstances when it is part of the GRN. In addition, it may make the system more evolvable when optimising a lower dimensional genome before moving into the higher-dimensional gene space of ADSs, which is achieved with the indirect mappings.

In addition, variable length representations also offer the benefit of more compact solutions, which provide a computational speed-up when running the ADS. This is helpful when performing a large number of evolutionary runs that are necessary to obtain sufficient statistical data. More compact solutions also provide a higher degree of compression of the information and regularities of the encoded organisms, which may enable scalability in the system as it becomes accessible for techniques where evolution starts with small genomes and then moves towards larger ones. Examples for this are scalability via complexification or building up higher-level modules.

Although the problem domain focusses on pattern formation, the work undertaken in this paper is an important step where the effectiveness, behaviour and the performance of a variable length ADS is tested on one of the widely used benchmarks in the field. Due to the fact that there are additional conceptual benefits of variable length representations and compact GRNs, finding variable length encodings perform equally well as the fixed length ones should have broader consequences on the design of GRN based ADSs. For example, in the case of the variable length representations the entire optimisation process that was done for the fixed length ones in Section VI would be unnecessary.

E. Contributions.

- A portfolio of encodings that are suitable to create variable length GRNs, which feature the same performance as simple encodings but deliver smaller GRNs and reduce optimisation processor time are presented.
- ADSs that have the ability to yield compact solutions, which may aid with scalability of the system when tackling more complex problems.
- Better understanding of how GRN based ADSs work and which mechanisms are important to include is illustrated.
- An ADS that is capable of producing compact GRNs that will be suitable for robotics/embedded systems is proposed.
- A classification of widely used patterns in the field which is based on a variety of different encodings and a total of 1100 evolution runs is suggested.

F. Future Work and Improvements of the ADS.

The results look promising since compact representations of GRNs that feature good performance can be achieved, which enabled the image compression application introduced in [56] and an adaptive robot controller presented in [57]. Compact representations will be more than advantageous when taking the ADS introduced into hardware, as outlined in [24], [49]. Future work will therefore follow three threads: first, further investigating and improving the ADS presented with the focus set on a dynamic pattern task (suggested by the success in the image compression application), which might prove to be a more meaningful task for dynamic systems and potentially establish a new benchmark beyond static patterns and stable patterns alone. Second, observations made with the current system concerning fault tolerance suggest that in order to make the transition to fault tolerant real-world applications, either single cell approaches or significantly higher resolutions of the cell grid would be beneficial. However, the need for multicellularity is often hard to justify as there are examples where an ADS using a single cell proves to be sufficient to carry out complex control tasks. On the other hand, for distributed systems a multicellular system would lend itself easier to implement and also appears to be a more "natural" structure in the case of multiple physical entities, such as arrays of processors, chips or groups of robots. Third, in order to fully exploit the benefits of compact GRNs, the ADS presented will be used to create developmental models for a distributed hardware system. This will allow to investigate whether the representations introduced prove useful and are applicable to real-world applications.

ACKNOWLEDGMENT

This work is part of a project that is funded by EPSRC - EP/E028381/1.

REFERENCES

[1] W. W. Gibbs, "The unseen genome: gems among the junk." *Scientific American*, vol. 289, no. 5, pp. 26–33, 2003.

[2] I. B. Dawid, "The Regulatory Genome, by Eric H. Davidson (2006), Academic Press," *The FASEB Journal*, vol. 20, no. 13, pp. 2190–2191, Nov. 2006.

[3] A. Chavoya, *Foundations of Computational, Intelligence Volume 1*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 201/2009, pp. 185–215.

[4] A. Devert, N. Bredeche, and M. Schoenauer, *Robust multi-cellular developmental design*. New York, New York, USA: ACM Press, 2007.

[5] P. Eggenberger, "Evolving morphologies of simulated 3d organisms based on differential gene expression," in *Fourth European Conference on Artificial Life*, vol. pages. The MIT Press, 1997, pp. 205–213.

[6] N. Flann, J. Hu, M. Bansal, V. Patel, and G. Podgorski, "Biological development of cell patterns: Characterizing the space of cell chemistry genetic regulatory networks," *Advances in Artificial Life*, pp. 57–66, 2005.

[7] S. L. Harding, J. F. Miller, and W. Banzhaf, "Self-modifying cartesian genetic programming," in *Proc. of the 9th annual conference on Genetic and evolutionary computation (GECCO)*. New York, NY, USA: ACM, 2007, pp. 1021–1028.

[8] S. Kumar and P. J. Bentley, "Biologically Plausible Evolutionary Development," in *In Proc. of the 5th International Conference on Evolvable Systems: From Biology to Hardware (ICES)*. Springer, 2003, pp. 57–68.

[9] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, Jun. 2007.

[10] G. Tufte, "The discrete dynamics of developmental systems," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, May 2009, pp. 2209–2216.

[11] J. F. Miller, "Evolving a self-repairing, self-regulating, french flag organism," in *Genetic and Evolutionary Computation (GECCO)*, 2004, pp. 129–139.

[12] D. Roggen, "Multi-cellular reconfigurable circuits: Evolution morphogenesis and learning," *Ph.D. dissertation, EPFL*, pp. 91–94, 2005.

[13] L. Wolpert, R. Beddington, T. Jessell, P. Lawrence, E. Meyerowitz, and J. Smith, *Principles of development*. Oxford University Press Oxford, 2002.

[14] J. T. Bonner, "The origins of multicellularity," *Integrative Biology: Issues, News, and Reviews*, vol. 1, no. 1, pp. 27–36, 1998.

[15] D. Federici, "Using Embryonic Stages to increase the evolvability of development," in *In Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*. Springer Verlag, 2004.

[16] K. O. Stanley and R. Miikkulainen, "A Taxonomy for artificial embryogeny," *Artif. Life*, vol. 9, no. 2, pp. 93–130, 2003.

[17] K. Fleischer and A. H. Barr, "A Simulation Testbed for the Study of Multicellular Development: The Multiple Mechanisms of Morphogenesis," in *Third Artificial Life Workshop*, Santa Fe, New Mexico, USA, Jun. 1993, pp. 389–416.

[18] N. Jakobi, "Harnessing Morphogenesis," in *International Conference on Information Processing in Cells and Tissues*, 1995, pp. 29–41.

[19] H. Kitano, "A simple model of neurogenesis and cell differentiation based on evolutionary large-scale chaos," *Artif. Life*, vol. 2, no. 1, pp. 79–99, 1995.

[20] G. Tempesti and P.-A. Mudry, "A Move Processor for Bio-Inspired Systems," in *Proc. of the NASA/DoD Conference on Evolvable Hardware*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 262–271.

[21] S. Wolfram, *A New Kind of Science*. Champaign, IL, USA: Wolfram Media, 2002.

[22] T. G. W. Gordon, "Exploiting Development to Enhance the Scalability of Hardware Evolution," Ph.D. dissertation, University College London, Jul. 2005.

[23] P. C. Haddow and J. Hoye, "Achieving a Simple Development Model for 3D Shapes: Are Chemicals Necessary?" in *Proc. of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. New York, NY, USA: ACM, 2007, pp. 1013–1020.

[24] T. Kuyucu, M. A. Trefzer, J. F. Miller, and A. M. Tyrrell, "On the Properties of Artificial Development and Its Use in Evolvable Hardware," in *Proceedings of the IEEE Symposium Series on Computational Intelligence (ISSCI)*, Nashville, USA, Mar. 2009.

[25] J. Miller, "Evolving developmental programs for adaptation, morphogenesis, and self-repair," *Advances in Artificial Life*, pp. 256–265, 2003.

[26] P. Bentley and S. Kumar, "Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem," in *Proc. of the Genetic and Evolutionary Computation Conf.* Orlando, Florida, USA: Morgan Kaufmann, 1999, pp. 35–43.

[27] R. Dawkins, *On Growth, Form and Computers*. Elsevier Academic Press, 2003, pp. 239–255.

[28] A. Lindenmayer, "Mathematical models for cellular interactions in development. I. Filaments with one-sided inputs." *Journal of Theoretical Biology*, pp. 280–299, 1968.

[29] P. C. Haddow, G. Tufte, and P. van Remortel, "Shrinking the Genotype: L-systems for EHW?" in *ICES*, 2001, pp. 128–139.

[30] H. Kitano, "Building Complex Systems Using Developmental Process: An Engineering Approach," in *ICES '98: Proceedings of the Second International Conference on Evolvable Systems*. London, UK: Springer-Verlag, 1998, pp. 218–229.

[31] E. J. W. Boers and H. Kuiper, "Biological Metaphors and the Design of Modular Artificial Neural Networks," Master's thesis, Leiden University, 1992.

[32] G. Hornby and J. Pollack, "The advantages of generative grammatical encodings for physical design," in *Congress on Evolutionary Computation (CEC)*. IEEE, 2001, pp. 600–607.

[33] K. Sims, "Evolving 3D morphology and behavior by competition," *Artif. Life*, vol. 1, no. 4, pp. 353–372, 1994.

[34] F. Gruau, "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm." Ph.D. dissertation, Ecole Normale Supirieure de Lyon, France, 1994.

[35] F. Dellaert and R. D. Beer, *Toward an Evolvable Model of Development for Autonomous Agent Synthesis*. MIT Press Cambridge, 1994.

[36] P. C. Haddow and J. Hoye, "Investigating the effect of regulatory decisions in a development model," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, May 2009, pp. 293–300.

[37] T. Steiner, Y. Jin, and B. Sendhoff, "A cellular model for the evolutionary development of lightweight material with an inner structure," in *Proc. of the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2008, pp. 851–858.

[38] C. Y. Lee and E. K. Antonsson, "Variable Length Genomes for Evolutionary Algorithms." In Proceedings of the Genetic and Evolutionary Computation Conference, 806. Las Vegas, 2000.

[39] T. Reil, "Dynamics of Gene Expression in an Artificial Genome - Implications for Biological and Artificial Ontogeny," pp. 457–466, 1999.

[40] O. Leyser and S. Day, *Mechanisms in Plant Development*. Blackwell, 2003.

[41] J. Knabe, C. Nehaniv, and M. Schilstra, "Regulation of gene regulation-smooth binding with dynamic affinity affects evolvability," in *IEEE Congress on Evolutionary Computation (CEC)*, 2008, pp. 890–896.

[42] Y. Jin, R. Gruna, I. Paenke, and B. Sendhoff, *Evolutionary multi-objective optimization of robustness and innovation in redundant genetic representations*. IEEE, Mar. 2009.

[43] Y. Jin and J. Trommler, "A Fitness-Independent Evolvability Measure for Evolutionary Developmental Systems," in *Symposium on Computational Intelligence in Bio-informatics and Computational Biology*. Montreal: IEEE, 2010, pp. 69–76.

[44] J. F. Knabe, C. L. Nehaniv, and M. J. Schilstra, "Genetic regulatory network models of biological clocks: evolutionary history matters." *Artificial life*, vol. 14, no. 1, pp. 135–148, Jan. 2008.

[45] J. F. Miller and P. Thomson, "Cartesian Genetic Programming," in *Genetic Programming, EuroGP*. Springer, 2000, pp. 121–132.

[46] ——, "Aspects of digital evolution: Evolvability and architecture," *in Proc. Parallel Probl. Solving Nature V, A. E. Eiben, T. Bäck, M. Schoenauer, and H. P. Schwefel, Eds., Lecture Notes in Computer Science, vol. 1498, Berlin, Germany, Springer-Verlag*, pp. 927–936, 1998.

[47] C. Ryan, J. J. Collins, and M. O'Neill, "Grammatical Evolution: Evolving Programs for an Arbitrary Language," *Lecture Notes In Computer Science*, vol. 1391, pp. 83–96, 1998.

[48] G. Hornby, "Generative representations for evolving families of designs," in *Genetic and Evolutionary Computation (GECCO)*, vol. 12. Springer, 2003, pp. 209–217.

[49] M. A. Trefzer, T. Kuyucu, J. F. Miller, and A. M. Tyrrell, "A Model for Intrinsic Artificial Development Featuring Structural Feedback and Emergent Growth," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, Norway, 2009, pp. 301–308.

[50] R. A. Dhanasekaran, G. J. Podgorski, and N. S. Flann, "Co-option and Irreducibility in Regulatory Networks for Cellular Pattern Development," in *2007 IEEE Symposium on Artificial Life*. IEEE, Apr. 2007, pp. 179–186. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4218884

[51] A. Chavoya and Y. Duthen, "Evolving an Artificial Regulatory Network for 2D Cell Patterning," in *2007 IEEE Symposium on Artificial Life*. IEEE, Apr. 2007, pp. 47–53. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4218867&tag=1

[52] A. Devert, N. Bredeche, and M. Schoenauer, "Robustness and the Halting Problem for Multi-Cellular Artificial Ontogeny," *IEEE Transactions on Evolutionary Computation*, 2011. [Online]. Available: http://hal.inria.fr/inria-00566879_v1/

[53] A. Vargha and H. D. Delaney, "A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, Jan. 2000.

[54] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT, 1992.

[55] M. Walker, H. Edwards, and C. Messom, "Confidence intervals for computational effort comparisons," *Lecture Notes In Computer Science*, pp. 23–32, 2007.

[56] M. A. Trefzer, T. Kuyucu, J. F. Miller, and A. M. Tyrrell, "Image Compression of Natural Images Using Artificial Gene Regulatory Networks," in *Genetic and Evolutionary Computation Conference (GECCO)*. Portland: ACM, 2010.

[57] ——, "Evolution and Analysis of a Robot Controller Based on a Gene Regulatory Network," in *International Conference on Evolvable Systems (ICES)*. York: Springer, 2010.