Lateralized Learning to Solve Complex Boolean Problems

Abubakar Siddique, *Member, IEEE*, Will N. Browne, *Member, IEEE*, and Gina M. Grimshaw

***Abstract*—Modern classifier systems can effectively classify targets that consist of simple patterns. However, they can fail to detect hierarchical patterns of features that exist in many real-world problems, such as understanding speech or recognizing object ontologies. Biological nervous systems have the ability to abstract knowledge from simple and small-scale problems in order to then apply it to resolve more complex problems in similar and related domains. It is thought that lateral asymmetry of biological brains allows modular learning to occur at different levels of abstraction, which can then be transferred between tasks. This work develops a novel evolutionary machine-learning (EML) system that incorporates lateralization and modular learning at different levels of abstraction. The results of analyzable Boolean tasks show that the lateralized system has the ability to encapsulate underlying knowledge patterns in the form of building blocks of knowledge (BBK). Lateralized abstraction transforms complex problems into simple ones by reusing general patterns (e.g., any parity problem becomes a sequence of the 2-bit parity problem). By enabling abstraction in evolutionary computation, the lateralized system is able to identify complex patterns (e.g., in hierarchical multiplexer (HMux) problems) better than existing systems.**

***Index Terms*—Building blocks, cognitive neuroscience, lateralization, learning classifier systems (LCSs), modular learning.**

I. Introduction

CURRENT learning systems exhibit world-leading behavior in playing complicated strategic games, but they often struggle on other apparently simple tasks, especially when an abstract concept needs to be discovered and reapplied in a new setting [1]. They struggle to identify patterns within patterns, and so they cannot reuse knowledge efficiently [2]. Instead, repeated patterns within a layer of a neural network must be learned separately. Although new innovations, such as attention mechanisms [3]–[5] and capsules [6], [7] address some of these problems, most conventional systems still depend on a homogeneous approach, in which all features within a layer are treated in the same manner.

Manuscript received March 3, 2021; revised September 22, 2021 and January 21, 2022; accepted March 25, 2022. This work was supported by the Science for Technological Innovation National Science Challenge, New Zealand. This article was recommended by Associate Editor K.-C. Tan. *(Corresponding author: Abubakar Siddique.)*

Abubakar Siddique is with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: absiddique@ecs.vuw.ac.nz).

Will N. Browne is with the Faculty of Engineering, School of Electrical Engineering and Robotics, Queensland University of Technology, Brisbane, QLD 4000, Australia.

Gina M. Grimshaw is with the Cognitive and Affective Neuroscience Lab, School of Psychology, Victoria University of Wellington, Wellington 6140, New Zealand.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2022.3166119.

Digital Object Identifier 10.1109/TCYB.2022.3166119

Evolutionary machine-learning (EML) alternatives show promise in classifying targets that consist of simple patterns of data, but existing EML-based systems still struggle with many real-world problems that entail hierarchical patterns. An effective alternative might be found in learning classifier systems (LCSs), an EML approach that naturally splits a solution into niches, allowing it to handle hierarchical patterns in a heterogeneous manner.

These heterogeneous features could represent knowledge at different levels of abstraction in compact building blocks of knowledge (BBK) that are relevant and sufficient to solve a specific problem [8]. A BBK is a unit of knowledge utilized by the artificial agent to represent either part of a problem or the complete problem. Levels of abstractions are perspectives that can be used to address a problem. For example, in a visual task, a local viewpoint (eyes, nose, and mouth of a cat), and a global viewpoint (whole cat) are two different levels of abstraction [9], [10].

The use of heterogeneous knowledge representation can be seen in biological intelligence(s), which can learn new knowledge from simple and small-scale problems and then apply it to resolve more complex problems in similar and related domains [11], [12]. Two organizing principles of vertebrate brains—1) lateralization and 2) modularity of function—support this reuse of learned knowledge. We propose that the incorporation of these principles could overcome the limitations inherent in homogeneous systems.

Lateral asymmetry in vertebrate (and many invertebrate) brains allows for modular learning at multiple levels of abstraction [13], [14]. For example, in many domains, the left hemisphere processes elementary (constituent) information, while the right hemisphere simultaneously processes the same information at a higher (holistic) level of abstraction. This organization has been shown to produce advantages in neural efficiency and cognitive performance [15]–[17].

It is hypothesized that a lateralized EML system can efficiently learn complex problems by splitting knowledge into constituent and holistic components. One half of the system (which we call the *left* half) needs to represent the most basic elements of knowledge; that is, individual features and simple

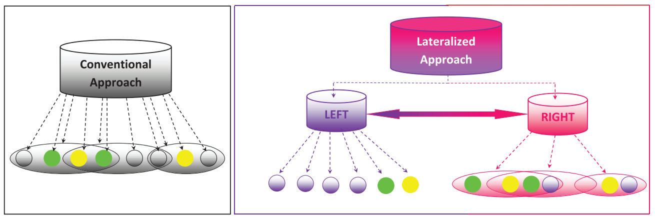

Fig. 1. Conventional EML approach (left) considers individual features and niches in a homogeneous manner. A lateralized approach (right) splits a complex problem into constituent and holistic knowledge.

niches.[1] At the same time, the other (*right*) half needs to create a more abstract knowledge representation by reapplying basic knowledge to represent higher order features across niches. Finally, the left half and the right half need to cooperate to efficiently resolve the problem. Many real-world problems are constructed from subproblems. Hence, the system must be able to consider the problem at two different levels of abstraction (constituent level and holistic level) simultaneously. A schematic explanation that contrasts the conventional homogeneous and lateralized approaches is shown in Fig. 1.

Apart from LCSs, other component-based EML techniques exist but they provide only a partial solution to this problem. For example, a layered learning GP (LLGP) technique has been used to cope with the hierarchical distribution of knowledge. This technique requires a human-in-the-loop hierarchical decomposition of the task and the selection of suitable algorithms for the learning of subtasks. Moreover, LLGP requires a strict sequence of learning to resolve large-scale problems [18], [19]. Similarly, a transfer learning-based GP technique can easily solve simple classification tasks but struggles to completely learn complex (e.g., 7-bit parity) problems [20]. Cartesian GP (CGP) is another GP-based technique that implicitly reuses graph nodes to address large-scale problems [21], [22]. Although CGP can produce many-to-one genotype-to-phenotype mapping, it generates an arbitrary sequence of computer programs that can solve only a particular problem. It also creates very large solutions, making the learning intractable [23].

LCSs are more tractable as their GP-tree-like code fragment (CF)-based representation can encode complex knowledge in hierarchical problems with minimal bloat [24], [25]. However, these systems require a huge number of training instances, strict ordering of layered learning, and much human intervention. Moreover, these systems process everything at the same level of abstraction during the learning process, that is, the population of rules is still monolithic [25], [26]. Consequently, these systems do not have the ability to identify and learn higher order abstract relationship(s) between the features and knowledge of a complex problem.

A. Goals and Benefits

The overall goal of this work is to enable abstraction in learning by creating a novel lateralized EML system, inspired by the principles of biological intelligence. The system will have the ability to apply lateralization and modular learning at different levels of abstraction to solve complex problems. To achieve this goal, the following objectives have been set.

1) Create a lateralized system such that a single input can be processed at different levels of abstraction, that is, at the constituent level and/or the holistic level. Instead of mapping features to knowledge in a homogeneous manner that considers all input features equally, the problem will be split into two halves. One half will map subgroups of features to knowledge at a constituent level, whereas the other will map all features to knowledge at a holistic level.
2) Represent BBKs in a heterogeneous manner. Different sized blocks of knowledge can be recombined in a recursive manner, that is, a holistic block can be (re)used as a constituent block at a higher level of abstraction.
3) Identify and reuse the relevant BBKs to efficiently resolve complex problems, that is, those consisting of patterns within patterns.

In real life, we often face problems that have multiple parts and are also multifaceted, where it is useful to be able to classify objects at the lower level of detail and at higher levels of abstraction, for example, when we recognize a car, we can identify it as a vehicle for transportation, but we can also identify windscreen, wheels, etc., depending on the complex problem we are trying to solve.

II. Background

The goals of this section are threefold: first, to present an overview of the benchmark problems that will be used to evaluate the novel system; second, to introduce the relevant biological principles of vertebrate intelligence that will inform the work; and third, to describe the current state-of-the-art EC techniques, especially LCSs, that will provide a foundation upon which the novel lateralized system will be constructed.

A. Problem Domains

To test the effectiveness of the lateralized approach for classification problems well-known complex Boolean problems (multiplexer (Mux), parity, majority-on, and carry problems) will be used. These problems possess a number of features essential to evaluating the lateralized approach. First, they exhibit heterogeneity and epistasis, which are known to cause problems in classification techniques as these cannot make the linearly separable assumption. Second, they possess identifiable components of a solution that are transferable to other problems, making them useful for evaluating transfer learning abilities. Third, they have known exact solutions, so the correctness of the produced solution can be interrogated. And finally, Boolean problems have *measurable search space*, *dependency structure*, *multimodal solutions*, and *distributed niches (subsolutions)* [27].

Although Boolean problems are sometimes considered to be "toy" benchmark problems, these characteristics make them analogous to real-world problems, such as those found in finance, bioinformatics, and behavior modeling. Hence, many

[1] A niche is an area of the sample space where the neighboring instances share a common property.

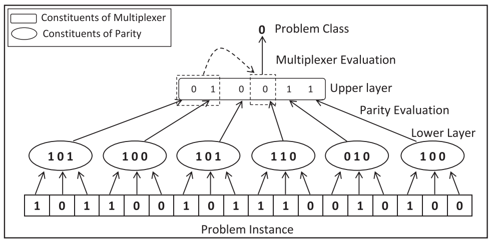

Fig. 2. Instance of the 18-bit hierarchical Mux problem. The lower layer consists of 3-bit parity problems, whereas the upper layer is a 6-bit Mux problem.

state-of-the-art systems have been evaluated by using Boolean problems [20], [24], [25], [28], [29]. The scalability of the lateralized approach will be investigated by utilizing more complex derived versions of these problems, such as hierarchical Mux problems and high-scale parity problems (i.e., beyond the common 7-bit problem).

Hierarchical problems are hard to resolve as they have low sparsity and hierarchical distribution of knowledge [24], [27]. Therefore, these problems are the best candidates with which to evaluate the effectiveness of the lateralized system. The hierarchical Boolean problems presented here consist of two layers. The lower layer is composed of multiple instances of a specific Boolean problem. The evaluation and integration of the lower layer generate an instance of the upper layer, which is another Boolean problem. For example, an 18-bit hierarchical problem may be composed of the following layers: 1) a lower layer based on a 3-bit parity problem and 2) an upper layer based on the 6-bit Mux problem. An example instance of an 18-bit hierarchical multiplexer (HMux) is shown in Fig. 2. A deeper introduction to Boolean problems, for readers who are unfamiliar, is presented in the supplementary material (see Section S-II).

B. Cognitive Architecture in Vertebrate Brains

Vertebrate brains have a functional architecture that allows them to abstract knowledge from simple and small-scale problems and then reuses it to solve complex problems. It is not the intention of this work to model the specific architecture of a specific species; rather to take inspiration from basic principles of functional organization that are fundamental to vertebrate intelligence. We focus here on two such principles: 1) lateralization and 2) semantic processing.

1) Lateralization: The propensity of a specific cognitive process to be performed more efficiently and precisely by one hemisphere as compared to the other is called hemispheric lateralization [30]. At the macrostructural view, the left and right hemispheres look alike. However, they have distinct neuroanatomy, neurochemistry, and functional architecture [30]. Three aspects of lateralization are relevant for applications to AI.

a) Representation and processing: Some functions are strictly lateralized to one hemisphere or the other. For example, each hemisphere receives sensory inputs from the opposite side of the body and controls the contralateral musculature. But, for higher order cognition, differences between hemispheres are more relative than absolute, with both hemispheres contributing to most tasks. Often these hemispheric differences concern the scale at which the same sensory inputs are represented for subsequent processing. For example, in visual perception, the left hemisphere processes information at a local (or constituent) level while the right hemisphere processes information at a more global (or holistic) level [31].

These fundamental differences in the representational scale may arise through filtering. For example, the double filtering by the frequency model proposes that the left hemisphere acts as a high pass filter, allowing it to represent detailed information that is available in high spatial or temporal frequencies. At the same time, the right hemisphere acts as a low pass filter, allowing it to represent global patterns that emerge in low spatial or temporal frequencies [32]. Such complementary forms of representation are not limited to sensory information, however. For example, in language processing, the left hemisphere may activate single, literal, meanings of words, or sentences, while the right hemisphere keeps alternative, metaphorical, or figurative meanings active [33]. This ability to represent and process the same problem instance at a local constituent level and a global holistic level will need to be incorporated in the work presented here [Objective 1)].

b) Coordination: Effective cognition requires that the computations carried out in opposite hemispheres be coordinated. Recognizing faces requires that we integrate individual features (left) with their configural arrangement (right) [34]; understanding a joke requires that we integrate the literal meanings of individual words (left) with their alternative subtext (right) [35]; and understanding a song requires that we integrate the lyrics (left) with the melody (right) [36]. It is the coordination between the left and right hemispheres that enables the transfer of critical information at different levels of abstraction. This coordination needs to be included in the modules created here [Objective 3)].

c) Goal-driven processing: Vertebrate brains have the ability to select the computations required to perform a specific task from the most suitable and relevant hemisphere. Goal-driven processes analyze the problem at hand and shift control to the superior and suitable hemisphere. For example, if the emotional state of a conversational partner is most relevant, outputs from right hemisphere speech processing systems will dominate; however, if the linguistic elements are of concern, then left hemisphere computations are prioritized [16], [17]. The connections between hemispheres in vertebrate brains can be excitatory or inhibitory, allowing for either integration or inhibition, as goals dictate [37]. The ability to identify which hemisphere is best matched to the task is important in practical situations. A strategy needs to be developed for the identification of the most suitable module with respect to the given problem instance [Objective 3)].

2) Semantic Processing: Semantic cognition is the competency to utilize, control, and generalize learned knowledge. Dominant models of semantic cognition include two interacting components—1) a representational system that abstracts information from the environment to form concepts and 2) a control system that selects and activates the appropriate features of a concept given current goals or contexts [38], [39].

a) Semantic representation: A *concept* is a coherent collection of knowledge about the world. According to the hub-and-spoke model [39], the features that comprise a concept are distributed throughout the brain in modal spokes (motor, auditory, color, shape, etc.) that are formed through sensory-motor experience and/or abstracted from statistical regularities in the environment. These modal spoke then connect to a cross-modal hub (anatomically located in the anterior temporal lobe in humans), in which related concepts are connected to each other. The hub is, therefore, able to represent generalized concepts that are independent of any specific instance. The activation of a concept (e.g., a hammer) then entails activation of its cross-modal hub, along with modality-specific activation of its features (how it is held, what it is used for, its visual form, etc.) [40]–[43]. Although the network is bilateral, left and right hubs display subtle asymmetries by virtue of their connections to lateralized perceptual and motor systems [38].

b) Semantic control: Not all features of a concept are relevant in all contexts. Anatomically distinct semantic control mechanisms (localized in the frontal cortex) activate the most relevant meanings with respect to the current task or subtask [39]. For example, when asked to describe the similarities between a fire extinguisher and a tomato, people show increased activation in color-processing areas of the brain. In contrast, reflecting on the similarity between hammers and hacksaws activates areas that generate motor movements. Controlled semantic cognition therefore allows for concepts to be constructed from BBKs as dictated by current needs [39], [40], [42], [43].

Concepts are therefore flexible; able to represent both generalizable and task-specific knowledge. This flexible use of concepts needs to be incorporated in this work to allow the representation of knowledge at different levels of abstraction through the activation of relevant BBKs. Both specific and generalized knowledge need to be utilized in concert to resolve heterogeneous problems. The cognitive concept hypothesis will be utilized to generate a block of knowledge (named *concept*) for each learned problem (see Section III). The system will need to have the ability to utilize a learned concept at a constituent level or holistic level depending on its occurrence in the problem instance [Objective 2)].

C. Evolutionary Computation

Early attempts to create artificial cognitive systems were inspired by evolutionary computation [44]. The relevant evolutionary computation approaches are described as follows.

1) Layered Learning GP: LLGP is a methodology for learning complex problems by (a human-in-the-loop) decomposing them into a hierarchy of subtasks [19]. These subtasks are separately learned in a sequence by utilizing suitable learning algorithms. The knowledge learned for a subtask at the lower layer of the hierarchy is utilized for the learning of a task at the upper layer [18], [45]. The layered learning approach is suitable for complex problems where direct learning is intractable and hierarchical decomposition is possible. The layered learning approach requires human intervention to decompose the complex problem and select a suitable algorithm for learning a subtask. Moreover, the layered learning methodology requires a strict sequence of learning in order to solve complex tasks [46]. In contrast, our novel system will have the ability to simultaneously analyze the complex problem at both constituent and holistic levels. Consequently, the novel system will identify and utilize relevant BBKs without human intervention.

Recently, a common subtree-based transfer learning technique has been introduced in GP. The resultant system has the ability to automatically find the relevant information that can be transferred between the problems. However, it cannot transfer information between different problem domains. Moreover, this system struggles to completely learn complex problems, for example, 7-bit parity problems [20]. The novel lateralized system will have the ability to handle heterogeneous and complex Boolean problems.

2) Cartesian GP: CGP is a methodology for developing graph-based programs to solve a problem. In CGP, a program is represented by a 2-D grid of nodes in the form of a graph. The program has the ability to implicitly reuse nodes in the graph. Consequently, the mapping from genotype to phenotype in such a program is many-to-one [21], [47]. The CGP technique generates an arbitrary sequence of computer programs that can solve only a particular problem; the learned solution cannot be reused to solve other complex problems. Moreover, the size of the solutions is very large for large-scale and complex problems. Consequently, the learning becomes intractable [23]. The novel system will have the ability to generate a compact solution by (re)using the learned BBKs at different levels of abstraction.

3) Learning Classifier Systems: LCSs are a methodology for developing rule-based machine learning by applying discovery algorithms and learning components [26]. Holland and Reitman developed the first LCS, naming it CS-1, that is, "Cognitive System One" [44]. An LCS-based artificial agent learns by taking the appropriate action that maximizes its current or future rewards [48]. LCSs apply genetic algorithms (GAs) to generate new rules and explore unique niches in the problem space. In this way, LCSs evolve a collection of classifier rules that collectively solve the given problem. These rules have fitness based on their contribution toward the solution. In the learning process, LCSs improve the fitness of good rules and produce new fitter rules. Moreover, to keep the population of rules within a bound, LCSs apply different condensation and compaction techniques to remove the weak, and redundant classifier rules from the population [49]. A brief introduction to GA and LCSs, for readers unfamiliar with these base techniques, is presented in the supplementary material (see Section S-I).

4) Code Fragments: CF representation is an important encoding scheme that has been introduced in LCSs to achieve high-level knowledge representation by storing and transferring learned knowledge [25]. A CF is a GP-like tree in which the internal nodes have operations that are selected from a predefined set, that is, AND, OR, NOT, NAND, and XOR. The leaf nodes, in contrast, can represent environment variables or previous CFs. The inclusion of CFs in LCSs resolves previously unresolvable complex problems (such as the 135-bit Mux problem, n-bit parity problem, and n-bit Mux problem), enables the transfer of learned knowledge, and generates compact rule sets for an optimal population [24], [25].

Existing CF-based systems struggle to solve complex problems, such as hierarchical Boolean problems (see Section IV) and have many constraints. For example, a solution proposed to resolve the n-bit Mux problem requires a strict ordering of layered learning, prior knowledge of problem decomposition, and much human intervention [24]. Similarly, the cyclic graphs-based technique (XCSSMA), which can solve n-bit parity problems, is not beneficial for Mux problems because the finite state machines used in the action are unable to capture patterns in the Mux search space [23]. Recently, CF-fitness-based techniques have been introduced to search for useful and relevant features for efficient learning. Nevertheless, these systems struggle to completely learn 18-bit HMux problems due to the homogeneous nature of their knowledge representation [50]. A novel CF-based technique will need to be developed to facilitate the efficient learning of complex BBKs at different levels of abstraction without human intervention.

III. Novel Lateralized System

This work develops a novel lateralized system for classification problems. We first introduce the novel techniques that are used to achieve heterogeneous knowledge representation at different levels of abstraction. Then, the overall strategy adopted by the novel system to resolve a given problem is presented. Finally, the learning methodology of the lateralized system is described.

A. Representing Heterogeneous Knowledge

The knowledge associated with a learned problem needs to be stored for future use. Initially, the CFs link represented environmental features through functional nodes. A disjunctive normal form of CFs constitutes a rule, which encapsulates how well CFs link together to classify the problem. These rules are then combined in a population, which enables specific niches of the problem to solve the problem domain.

The knowledge associated with a problem is stored as a BBK in the knowledge pool. This BBK, termed a *concept*, includes the set of rules, CFs, attributes (such as length of an instance, for example, 3-bit parity has length 3), and unique ID, applicable in a learned problem domain. At a constituent level, the conditions in the classifier are the encoded features of the input problem instance. Because concepts have unique IDs and may appear in the terminals of a CF, they can be (re)used in the conditions of the holistic-level decision-making process. Conventional LCSs have a set of rules (population) for each learned problem. In contrast, the lateralized system stores knowledge for all learned problems as a set of concepts in the knowledge pool. A concept has sufficient knowledge to solve any instance of its associated problem. A schematic illustration of concepts is shown in Fig. 3, with length, number of rules, and number of CFs (both in condition and action of the "if <conditions> then <action>" rules) attributes.

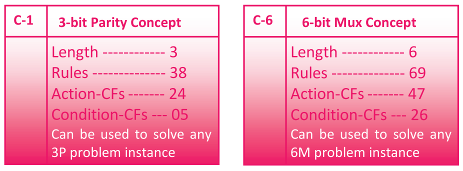

Fig. 3. Illustration of 3-bit parity (C_1) and 6-bit Mux (C_6) concepts.

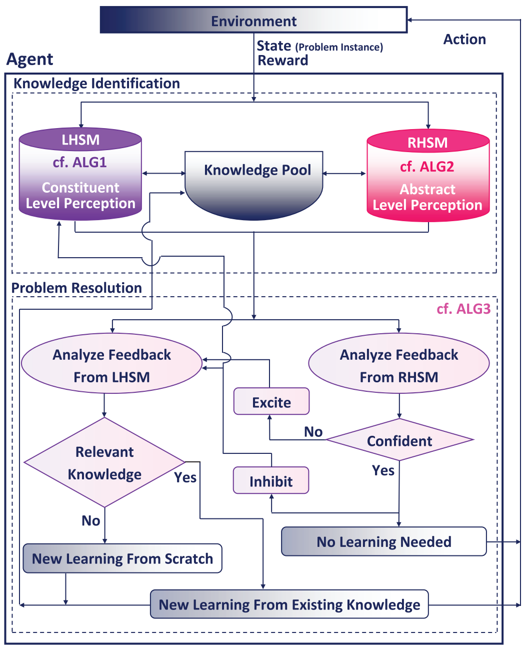

Fig. 4. Schematic illustration of the strategies developed to achieve cognitive inspired functionality in the lateralized system. RHSM and LHSM represent right hemispheric stratagem module and left hemispheric stratagem module, respectively.

B. Overall Strategy

A schematic illustration of the strategies developed for the novel lateralized system is shown in Fig. 4. The process by which the lateralized system addresses a problem can be divided into two main phases: 1) knowledge identification and 2) problem resolution.

The lateralized system needs to first determine if any knowledge has already been learned about the patterns in the input signal. This *knowledge identification* phase is critical as the system uses different learning methods depending on the quality (e.g., the accuracy of prediction) of existing knowledge to solve the problem. The novel system passes the input signal to the left half (to consider constituent patterns) and to the right half (to consider higher level holistic patterns) at the same time (see Fig. 4). The methods that determine the quality of the knowledge regarding the input signal are termed left-hemispheric stratagem module (LHSM) and right hemispheric stratagem module (RHSM), respectively.

The subsequent *problem resolution* phase evaluates the quality of the identified knowledge, that is, it can independently solve the problem (e.g., the accuracy of prediction is 100% or above a threshold), it can cooperate to solve the problem, or it is irrelevant to the problem. Consequently, action proceeds through one of three methods, that is, no learning needed, new learning from existing knowledge, or new learning from scratch. Despite this difference in methods used to determine the action, the overall state-action-reward scheme of the novel lateralized system is similar to standard reinforcement learning [51].

1) Knowledge Identification: Each environmental instance of the given problem is presented to both the LHSM and RHSM simultaneously in an online fashion. At the start of the learning process, the knowledge pool is empty. In such a scenario, both the LHSM and RHSM do not perform any computation and the system starts learning from a tabula rasa. The system is said to have learned a problem when its accuracy reaches 100% (or a given threshold). Hence, the associated concept is transferred to the knowledge pool.

The LHSM needs to identify those learned blocks of knowledge (concepts) that could be combined together to form the input instance. Concepts store their length, that is, the number of stored features. The LHSM iteratively searches through the knowledge pool to identify those concepts that have a length that is a factor of the length of the current instance. Repeating those concepts could therefore represent the complete instance. The identified concepts (termed constituent concepts) could form the lower layer of a solution to a hierarchical problem. For each constituent concept, the LHSM splits the given problem instance into parts such that the length of each part is equal to the length stored in the concept. Subsequently, each part is resolved by utilizing the constituent concept in the exploit mode (i.e., by selecting the best action). The resulting action bits are concatenated together to form an abstract layer, to be treated as a separate learning problem to be solved by the LHSM only.

Now, the LHSM again searches through the knowledge pool to identify those concepts that have a length equal to the length of the abstracted layer problem instance. The identified concepts are termed holistic concepts. The LHSM generates the candidate groups of concepts by pairing the constituent concept with each holistic concept. For example, the system is tasked to solve a problem instance of length 18 bits (features). Suppose that the lower layer constituent concept C_1 has a length 3 (the number of stored features). The resultant upper layer generated by utilizing C_1 has a length 6 (because C_1 is repeated for six parts). Furthermore, suppose that there are two learned concepts (C_5 and C_6) that have length 6. These concepts are selected as holistic concepts. The resultant candidate groups of knowledge are: 1) lower layer constituent concept C_1 and upper layer holistic concept C_5 and 2) lower layer constituent concept C_1 and upper layer holistic concept C_6. This process is repeated for each constituent concept.

The LHSM evaluates each candidate group by performing a preliminary test, for example, by utilizing 1000 environmental instances. During this evaluation process, the LHSM addresses a given problem instance by utilizing constituent concept and holistic concept in the exploit mode at the lower layer and higher layer of the problem instance, respectively.

Algorithm 1: Strategy Adopted by the LHSM to Identify the Candidate Groups of Concepts That Have the Potential to Efficiently Learn the Given Problem (cf. Fig. 4)

```
   Data: The environment and problem configurations
   Result: Relevant Groups of Concepts
1  Initialize the parameter settings and global variables;
2  Identify Constituent-Concepts();   % Generate a list of
   concepts that have a length (pre-determined value) which
   is a factor of the length of the current problem instance.
3  for Each Identified Constituent-Concept do
4      Split Problem();   % Split the problem instance
       into parts (sub-problems) such that each part has
       length equal to the length of the constituent-concept.
5      Solve Sub-Problems();   % Solve each part by
       utilizing the constituent-concept in exploit mode.
6      Get Higher Layer();   % The actions are
       concatenated to form a higher layer instance.
7      Identify Holistic-Concepts();   % Identify concepts
       that have a length equal to the length of the higher
       layer instance.
8      Generate Candidate Groups ();   % Group pairs of
       the constituent-concept with each holistic-concept.
9  end
10 Preliminary Test();   % Perform preliminary test for
   each candidate group by utilizing 1000 problem
   instances.
11 Mark Relevant Knowledge();   % Mark candidate
   groups of concepts that have a prediction accuracy equal
   to (or above) a given threshold.
12 Return Relevant Knowledge;
```

Subsequently, if the prediction accuracy is equal to (or above) a given threshold, the candidate group is marked (a flag is set) as relevant knowledge, otherwise, it will be discarded. Finally, the LHSM passes the relevant knowledge groups to the problem resolution method. The pseudocode of the strategy adopted by the LHSM for the identification of the relevant knowledge is given in Algorithm 1 and an illustrative walk-through is available in the supplementary material (Section S-III).

The RHSM needs to identify those learned blocks of knowledge that have sufficient knowledge to solve the problem independently (prediction accuracy is 100% or above a specified threshold). The concepts in the knowledge pool that have a length equal to the length of the given instance are termed candidate concepts. For example, the system is tasked to solve a problem instance of length 18 bits (features). Suppose that there are three learned concepts, C_1, C_6, and C_{10} that have lengths 3, 6, and 18, respectively. Only concept C_{10}, that has length 18, is selected as a candidate concept that may independently solve the given problem.

The RHSM computes confidence for each candidate concept. *Confidence* is the prediction accuracy of the RHSM in exploit mode. The RHSM computes confidence in two steps. Initially, the RHSM performs a preliminary test for each candidate concept. During the preliminary test, the RHSM solves

1000 problem instances utilizing the candidate concept in the exploit mode and computes confidence. If the confidence is below a specific threshold, for example, 90%, the candidate concept is removed from the candidate list. This process is repeated for each candidate concept.

In the next stage, the RHSM performs an intermediate level test for each candidate concept remaining in the list. During this stage, the RHSM is tasked to solve more problem instances, for example, 5000×length of the given problem instance. The RHSM solves these problem instances by utilizing the candidate concept in the exploit mode and computes the confidence again. If a confidence score above a defined threshold is produced, the RHSM concludes that it has seen this problem before. Otherwise, the RHSM removes this candidate concept from the candidate list. This process is repeated for each candidate concept. Finally, the RHSM shares the list of candidate concepts, along with their confidence values, with the problem resolution module (see Algorithm 2).

2) Problem Resolution: The problem resolution module receives feedback about the candidate concepts and candidate knowledge groups from RHSM and LHSM, respectively. During the analysis of the feedback from RHSM, the problem resolution module iteratively searches through the list of candidate concepts to find a concept that has a confidence value 100% (or above a given threshold). If such a concept is found, the problem resolution module marks the given problem as an already learned problem. Consequently, the system runs in exploit mode and utilizes that concept to solve any future problem instances given by the environment. Moreover, the mode is set as *no learning is needed*. The problem resolution module generates an inhibit signal so that the LHSM ceases and further analysis on the feedback from the LHSM is stopped. Else, the problem resolution module generates an excite signal to the LHSM to continue to identify relevant knowledge.

During the analysis of the feedback from the LHSM, if the relevant knowledge groups exist in the list. The problem resolution module iteratively searches through all the relevant knowledge groups and forms a list of all the concepts that are present in those groups. Subsequently, the problem resolution module marks all those concepts and their associated CFs (stored with those concepts) as relevant knowledge. The mode is set as *learning from existing knowledge*. Consequently, the system learns the given problem by utilizing all the relevant knowledge, that is, automatically using CFs and concepts in the classifiers. However, if the problem resolution module does not find any relevant knowledge groups in the feedback from LHSM, the mode is set as *new learning from scratch*. In such a scenario, the system behaves as an ordinary LCS and learns the given problem from a tabula rasa. The overall pseudocode of the novel strategy adopted by the lateralized system to solve a problem is given in Algorithm 3.

It is important to note that as each subsequent problem is learned, the concept, length (number of unique features addressed), and associated CFs are stored in the knowledge pool. The associated CFs can be used separately from the concepts to seed feature learning. A CF stores the relationship between features and targets. The individual CFs are used together in the condition and action of rules (see Section IV-E). Consequently, niches can form, allowing for heterogeneous representation of knowledge. These rules are encapsulated in concepts, which store the solution for a part of the problem or the whole problem. Moreover, a concept may represent a higher level of knowledge w.r.t. one problem, while also representing a lower level of knowledge w.r.t. another problem.

Algorithm 2: Strategy Adopted by the RHSM System to Identify the Candidate Concepts That Can Confidently Solve Problem (cf. Fig. 4)

```
Data: The environment and problem configurations
Result: The candidate concepts that can confidently solve
        the given problem
1  Identify candidate concepts();    % Generate a list of
   concepts that have a length equal to the length of the
   current problem instance.
2  for Each Identified Concept do
3      Preliminary Test();    % Perform preliminary test
       for each candidate concept by utilizing 1000 problem
       instances.
4      Compute Confidence();    % Confidence
       (prediction accuracy) of each concept to solve the
       problem.
5      if Concept Confidence < Threshold then
6              % The confidence is below 90%.
7          Remove Candidate Concept();    % Remove the
           candidate concept from the candidate list.
8      end
9  end
10 for Each Candidate Concept do
11     Intermediate Test();    % Perform intermediate
       level test for each candidate concept by utilizing
       (5000×length of the given problem instance) problem
       instances.
12     Compute Confidence();    % Confidence of each
       concept to solve the problem.
13     if Concept Confidence > Threshold then
14             % The confidence is above 90%.
15         Save Confidence Value();    % Save the
           confidence value for the candidate concept.
16     else
17         Remove Candidate Concept();    % Remove the
           candidate concept from the candidate list.
18     end
19 end
20 Return Candidate Concepts And Confidence Values();
```

C. Learning Methodology

Lateralization could be implemented by utilizing any EML-based system, but we have chosen LCS due to their niche-based algorithms and built-in support for heterogeneity (i.e., different rules can co-exist in the same population). These two features make LCSs an ideal candidate to support lateralization, extending the concepts of niches and heterogeneity to

Algorithm 3: Overall Strategy Adopted by the Lateralized System to Solve Problem (cf. Fig. 4)

```
Data: The environment and problem configurations.
Result: The given problem is identified as no learning
        needed, new learning from existing knowledge,
        or new learning from scratch.
1  Initialize the parameter settings and global variables;
2  Knowledge Identification
3      RHSM();      %Algorithm 1.
4      LHSM();      %Algorithm 2.
5  Problem Resolution
6      Analyze RHSM Output();      %Iteratively search
       through the list of candidate concepts and check their
       confidence values.
7      if Concept Confidence = 100% then
8          %Confidence value 100 (or above a given
           threshhold).
9          Run in Exploit Mode Only();
10         Generate Inhibit Signal();      %Generates an
           inhibit signal so that LHSM cease and further
           analysis on the feedback from LHSM is stopped.
11         Set No learning is Needed();
12     else
13         Generate Excite Signal();
14     end
15     Analyze LHSM Output();
16     if Relevant Knowledge Groups Exist then
17         %Iteratively search through all the relevant
           knowledge groups.
18         Form List of Concepts();      %Forms a list of all
           the concepts that are present in the relevant
           knowledge groups.
19         Mark Relevant Knowledge();      %Marks all
           those concepts and their associated CFs (stored
           with those concepts) as a relevant knowledge.
20         Set Learning From Existing Knowledge();
21     else
22         Set Learning From Scratch();
23     end
```

the different levels of a problem's composition. The learning methodology of the lateralized system is developed by utilizing the framework of Wilson's XCS [52], an accuracy-based LCSs. The ability of XCS to generate a complete and accurate solution enables the lateralized system to capture all the BBKs required for a concept. The learned concepts and associated CFs can then be used at different levels of abstraction. Moreover, the use of CFs provides improved expressivity and scalability as compared to the ternary alphabet [25]. These characteristics of CFs enable the lateralized system to efficiently handle diverse knowledge.

The lateralized system enhances the state-of-the-art CF-based XCS [25] in the following ways.

1) Condition and Action of Classifier: State-of-the-art CF-based systems have CFs either in the condition or in the action of the classifier [25], [53]. The novel lateralized system has CFs in both the condition and the action of the classifier. To the best of our knowledge, this is the first time that CFs have been included in both. The inclusion of concepts in the CFs and then CFs in both condition and action of the classifier enables the lateralized system to represent a complex and heterogeneous knowledge, reduce search space, generate compact rules, and have a lateralized population of rules.

2) Code Fragment Enhancement: The lateralized system enhances CFs such that the leaf nodes have the ability to randomly select concepts or other CFs or environment variables. This inclusion of concepts in the CF enables the classifier to independently resolve a large part of the problem.

3) Lateralized Population: Two contributions are inherent in the population of rules evolved by the lateralized system. First, CFs include concepts from different levels and domains in the knowledge pool. Second, the set of rules that are evolved to solve a problem, therefore, have diverse CFs from the learned BBKs. Consequently, the system incorporates diverse knowledge that enables the inclusion of a population of rules at different levels of abstraction. Thus, the population of rules is termed a lateralized population.

IV. Experimental Work

A. Learning Order

The order with which subproblems are presented to an EML system plays an important role in learning [24], [25]. In the start, the lateralized system could be trained with a diverse set of subproblems. The learned knowledge is stored in the knowledge pool. The lateralized system can automatically (without human involvement) identify the relevant required BBK from the knowledge pool (if they exist). Instead of trying each block of learned knowledge, the lateralized system adopts an efficient strategy to identify the relevant BBKs from the knowledge pool. To identify higher level relevant knowledge, the RHSM only evaluates the learned concepts that have the same length as that of the given problem instance. If the higher knowledge is not present, the LHSM analyses only those concepts that could be repeated to form a complete problem instance. Moreover, if the required knowledge is completely absent, the system considers it a novel problem and learns from a tabula rasa. In this case, the lateralized system behaves similarly to a conventional LCS.

For all the experiments, the lateralized system and CF-based LCS (conventional LCS does not utilize subproblems) are trained with the same set of relevant subproblems in the same order. A separate set of experiments for the lateralized system is also conducted to calculate the overhead due to extraneous subproblems (see Section IV-D).

B. Experimental Setup

The lateralized system uses the LCS parameter values that have been commonly used in [25], [54], and [55]. These values are: learning rate $\beta = 0.2$; prediction error threshold $\varepsilon_0 = 10$; fitness fall-off rate $\alpha = 0.1$; fitness exponent $\nu = 5$; GA threshold $\theta_{\text{GA}} = 25$; crossover probability $\chi = 0.8$; mutation probability $\mu = 0.04$; deletion threshold $\theta_{\text{del}} = 20$; deletion fraction $\delta = 0.1$; subsumption threshold $\theta_{\text{sub}} = 20$; don't care probability = 0.33; fitness reduction = 0.1; prediction error reduction = 0.25; and prediction reward = 1000. Additional parameter settings for XCS with the Bayesian optimization algorithm (XCS/BOA) [56] are: error BOA 400; $\theta_{\mathrm{BOA}} = 20$; maximum parents 4; minimum population 0; local population 10; and MCMC updates 18. All the experiments have been repeated 30 times and the values presented here are obtained by averaging the results of those experiments. The population size used for each experiment is presented in the supplementary material (see Section S-IV)

C. Experiments

The lateralized system is bootstrapped with five hard-coded Boolean functions, that is, AND, OR, NAND, NOT, and NOR. The system does not learn these basic functions. These functions are given to the lateralized system as they are commonly supplied to EML systems in [25]. Initially, the lateralized system learns simple problems, such as the 3-bit parity problem, 3-bit carry problem, and 6-bit Mux problem. Subsequently, the system is trained with higher level problems, such as 6-bit carry, 11-bit Mux, and 18-bit HMux problems. During the learning of these problems, the system is tasked to identify and utilize the relevant BBKs from previously learned knowledge. Consequently, the system efficiently learns new problems and acquires the ability to resolve higher level complex problems. The experimental results generated by the lateralized system (LateralXCS) are compared with the results generated by the conventional LCS (XCS), CF-based LCS (XCSCFC) [25], XCS/BOA [56], conventional GP, LLGP [18], and CGP [21] techniques.

1) Multiplexer Problems: The first set of experiments was conducted by using Mux problems to obtain a proof of concept of the lateralized system. All the LCS systems (XCS, XCSCFC, XCS/BOA, and LateralXCS) managed to learn 6-bit, 11-bit, and 20-bit Mux problems. For these problems, the learning pace of XCS/BOA is better than all other systems. The experimental results of 6-bit, 11-bit, and 20-bit Mux problems are presented in the supplementary material (see Section S-III-A). However, both XCS and XCS/BOA were unable to learn 37-bit Mux problems, whereas, XCSCFC and LateralXCS successfully learned the problem by utilizing 200 000 problem instances, see Fig. 5. The learning pace of the LateralXCS is slower than XCSCFC from 30 000 to 120 000 problem instances. This occurs because the LateralXCS has to identify suitable building blocks from the pool of learned knowledge. The XCSCFC swiftly achieved 99.8% accuracy but struggled to learn the problem completely. However, the LateralXCS achieved 100% accuracy by utilizing 300 000 problem instances. The average accuracy and standard deviation, for the last hundred runs, of LateralXCS and XCSCFC are 100 ± 0.00 and $99.99 \pm 9.52985e^{-05}$, respectively.

2) Parity Problems: All the LCS systems (XCS, XCSCFC, XCS/BOA, and LateralXCS) managed to learn 2-bit, 3-bit, and 4-bit parity problems. For these problems, the learning pace of XCS/BOA is better than all other systems. However, for 5-bit, 6-bit, and 7-bit parity problems, XCS and XCS/BOA are not able to learn the problems completely, whereas, XCSCFC

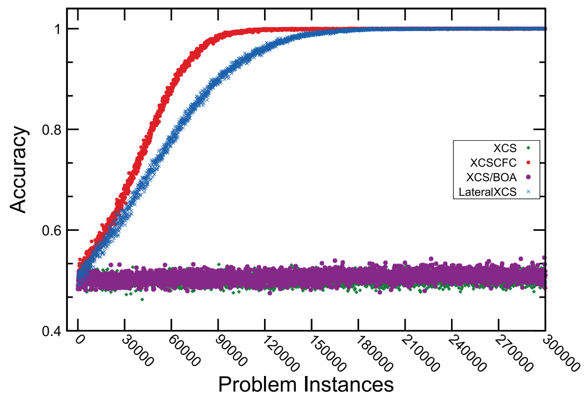

Fig. 5. Experimental results of 37-bit Mux problem using conventional LCS (XCS), CF-based XCS (XCSCFC), XCS/BOA, and Lateralized XCS (LateralXCS).

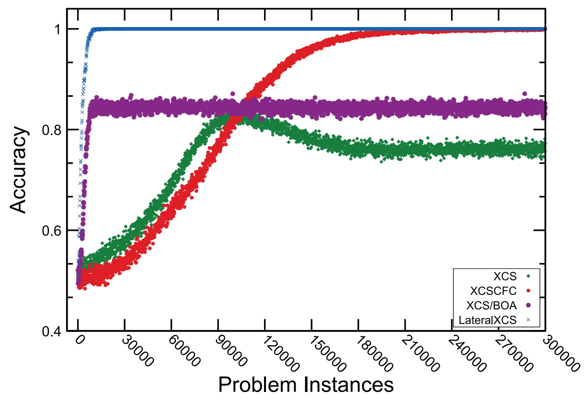

Fig. 6. Experimental results of 7-bit parity problem using XCS, XCSCFC, XCS/BOA, and LateralXCS.

lags behind the LateralXCS. The experimental results of 2-bit to 6-bit parity problems are presented in the supplementary material (see Section S-III-B).

The experimental results of the 7-bit parity problem are shown in Fig. 6. The LateralXCS successfully identified and reused the relevant building blocks of learned knowledge. Consequently, the system is able to efficiently learn the 7-bit parity problem by utilizing only 15 000 problem instances. In contrast, XCS, XCS/BOA, and XCSCFC learned 70%, 83%, and 100% 7-bit parity problem by utilizing 250 000 problem instances, respectively. The interpretation of rules generated by LateralXCS shows that the lateralized system automatically considers a higher level parity problem as a two-bit parity problem. Three features of LateralXCS play a critical role in this regard: 1) the ability to identify and utilize relevant learned *concepts*; 2) the ability to address a problem at different levels of abstraction; and 3) the ability to apply CFs to the condition as well as the action of classifiers.

A Wilcoxon signed-rank test was applied to statistically compare LateralXCS with XCSCFC (see Table I). The test was conducted on the results of the 100 problem instances after the lateralized system achieved a performance accuracy of 100%. The second and third columns contain the average performance along with the standard deviation. All *P*-values are less than 0.00001, which is evidence that the performance improvement of LateralXCS is statistically significant.

TABLE I
WILCOXON SIGNED-RANK TEST

WILCOXON SIGNED-RANK TEST

Problem Domain	XCSCFC	LateralXCS	Z-Value	P-Value
7-bit Parity	65.71 ± 1.38	100.00 ± 0.00	−8.6818	<0.00001
18-bit hierarchical Mux	96.30 ± 3.40	100.00 ± 0.00	−8.6818	<0.00001

TABLE II
PERFORMANCE COMPARISON WITH DIFFERENT GP SYSTEMS

Problem Domain	GP	LLGP	CGP	Lateralized XCS
7-bit Parity	60.09	76.31	52.47	100.00
18-bit hierarchical Mux	89.09	88.89	54.28	100.00

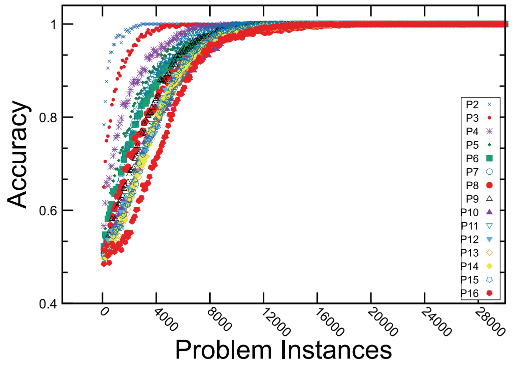

Fig. 7. Experimental results of 2-bit–16-bit parity problems using LateralXCS.

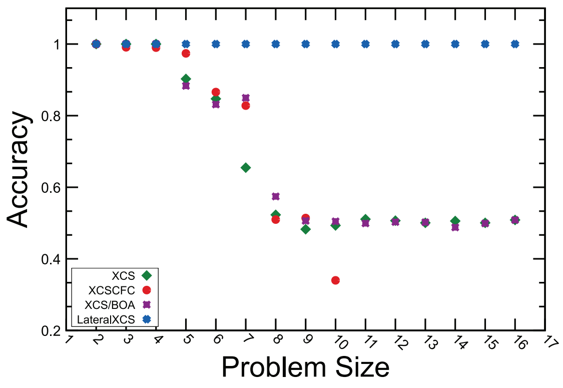

Fig. 8. Accuracy of XCS, XCSCFC, XCS/BOA, and LateralXCS after utilizing 50 000 problem instances of 2-bit–16-bit parity problems.

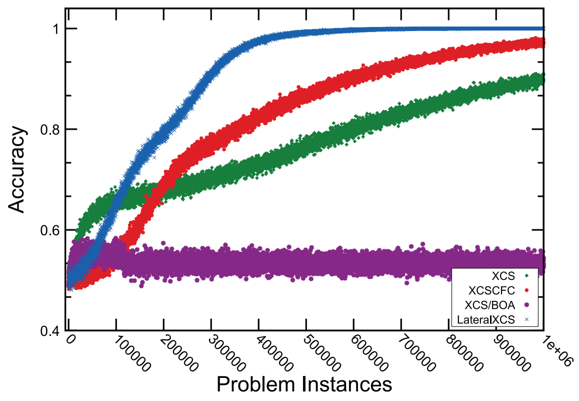

Fig. 9. Experimental results of 18-bit hierarchical Mux problem using XCS, XCSCFC, XCS/BOA, and LateralXCS.

The XCS and GP systems are based on different evolutionary techniques and apply different strategies to address problems. The main objective of the work presented here is not to develop a system that outperforms the GP-based system. However, their performance is compared to show the effectiveness of the lateralized approach. The same set of parity problem experiments was conducted by using GP, LLGP, and CGP systems. Each experiment was repeated 30 times with 50 generations and 1024 population. The experimental results of the 7-bit parity problem are shown in Table II. None of the GP-based systems were able to completely learn the 7-bit parity problem, whereas the LateralXCS achieved 100% accuracy using the same number of problem instances.

The effectiveness of the LateralXCS was tested by utilizing higher level parity problems. The experimental results of 2-bit to 16-bit parity problems using LateralXCS are shown in Fig. 7. The LateralXCS efficiently achieved a performance accuracy of 100% by utilizing less than 20 000 instances per problem. The performance scalability of the LateralXCS is explained in Section IV-E.

The accuracy achieved by XCS, XCS/BOA, XCSCFC, and LateralXCS after utilizing 50 000 problem instances during the learning of 2-bit to 16-bit parity problems is shown Fig. 8. XCS and XCS/BOA could not usefully learn beyond the 7-bit parity problem due to the lack of generalization in the ternary alphabet. The accuracy of XCSCFC gradually decreases from 100% (for 2-bit parity problems) to 34% (for 10-bit parity problems). Due to such a poor accuracy, XCSCFC stopped generating fit rules to match the test instances. Consequently, XCSCFC could not be applied to the remaining higher level parity problems due to its dependency on previously learned fitter rules. In contrast, the LateralXCS achieved 100% accuracy for all the given 2-bit–16-bit parity problems.

3) Hierarchical Problems: Two sets of experiments for hierarchical problems were conducted: 1) hierarchical Mux problems and 2) hierarchical carry problems. In 18-bit HMux problems, the lower layer consists of 3-bit parity problems and the upper layer consists of a 6-bit Mux problem. Initially, both XCSCFC and LateralXCS were trained with 3-bit parity and 6-bit Mux problems. Subsequently, all systems were tasked to learn the 18-bit HMux problem. The experimental results of the 18-bit HMux problem are shown in Fig. 9. Both the XCS and state-of-the-art XCSCFC were unable to efficiently learn the hidden patterns in the given problem, and XCS/BOA could not usefully learn the hierarchical problem (although it can learn these problems by using a large population). In contrast, the LateralXCS learned the hidden patterns and was able to organize the relevant BBKs in a useful way. Consequently, the LateralXCS learned the required rules to solve the problem, achieving performance accuracy of 100%. The number of instances ($\approx$ 600 000) utilized by the LateralXCS to learn the 18-bit HMux problem is low with respect to the complexity of the problem, that is, low sparsity and hierarchical distribution of knowledge.

The Wilcoxon signed-rank test was applied in a similar way as for parity problem experiments (see Table I). The

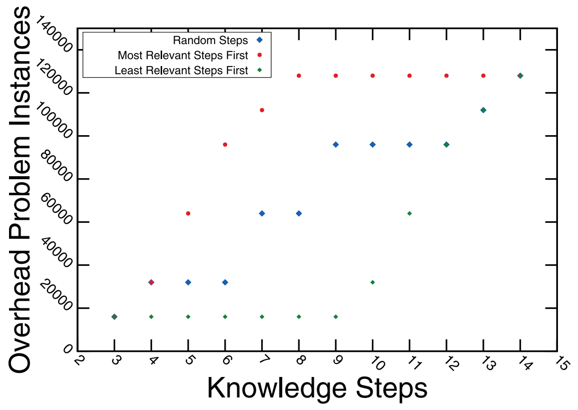

Fig. 10. Overhead problem instances utilized by LateralXCS during the learning of 18-bit hierarchical Mux problems.

P-values are less than 0.00001. The experimental results of the 18-bit HMux problem for GP-based systems are shown in Table II. It shows that none of the GP-based systems achieved the performance accuracy of 100%, whereas the LateralXCS achieved 100% performance accuracy by utilizing an equivalent number of problem instances. In 18-bit hierarchical carry problems, the lower layer consists of 3-bit majority-on problems and the upper layer consists of a 6-bit carry problem. All systems managed to learn the 18-bit hierarchical carry problem. The experimental results are presented in the supplementary material (see Section S-III-C).

D. Overhead of Irrelevant Subproblems to LateralXCS

The overhead (extra) problem instances utilized by the LateralXCS to identify the relevant BBKs during the learning of 18-bit HMux problems are presented in Fig. 10. The overhead cost depends on the relevancy of the learned problem concerning the given problem. For example, both 6-bit Mux and 6-bit parity problems are potentially relevant subproblems for learning the 18-bit HMux problem (the sequence of knowledge steps is presented in the supplementary material, see Section S-V). After utilizing preliminary problem instances, the novel system identified 6-bit Mux as relevant and 6-bit parity as irrelevant subproblems. LateralXCS also considers 5-bit MajorityOn and 5-bit parity as irrelevant subproblems without utilizing any problem instances during the learning of the 18-bit HMux problem. The number of combinations tried by the LateralXCS with the addition of a new problem along with the overhead cost is presented in the supplementary material (see S-Table II). At worst 128 000 problem instances are needed to find the relevant BBKs for 18-bit HMux when the system has learned 14 diverse knowledge steps that form 16 candidate combinations.

E. Interpretation of Decisions

The decision-making process of the LateralXCS is interpretable. Close observation of the rules generated by the LateralXCS reveals that the system successfully identified and efficiently utilized the relevant BBKs from the pool of learned knowledge.

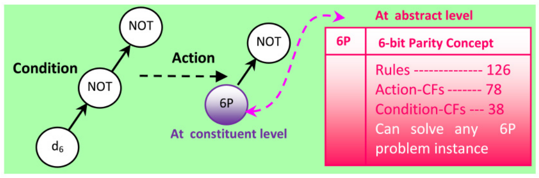

Fig. 11. Example rule (R1) from 7-bit parity problem (d_6 represents condition bit #6 and 6P represents 6-bit parity concept). Numerosity 25, Experience: 121465, Accuracy 1, Prediction Error: 0, Prediction: 1000, Fitness: 0.095, and Specificness 1. R1 is represented as $CF(\sim(\sim d_6)) : CF(\sim(6P))$.

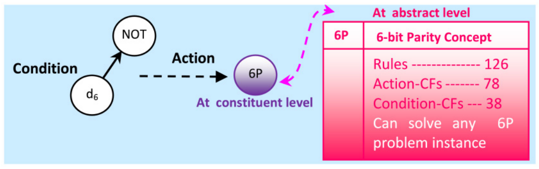

Fig. 12. Example rule (R2) from 7-bit parity problem (d_6 represents condition bit #6 and 6P represents 6-bit parity concept). Numerosity 6, Experience: 111038, Accuracy 1, Prediction Error: 0, Prediction: 1000, Fitness: 0.023, and Specificness 1. R2 is represented as $CF(\sim d_6) : CF(6P)$.

The LateralXCS efficiently learned the 7-bit parity problem by utilizing a small number of problem instances as compared to other EML systems, see Fig. 6. The learned concept of the 7-bit parity problem consists of 203 rules, 69 condition-CFs, and 116 action-CFs. An example rule from the learned concept of the 7-bit parity problem is shown in Fig. 11. This is the most experienced (iterations 121465) and accurate rule with high numerosity (25) and low specificity (1). The only CF in the condition has three elements, that is, condition bit #6, operator NOT, operator NOT. The condition CF is "not of not of condition bit #6," that is, NOT(NOT(D_6)). The CF in the action part has two elements, that is, the 6-bit parity concept and operator NOT. The action CF is "NOT of 6-bit parity." A rule matches with the given problem instance if all of the CFs in its condition generate value "1." According to this principle, the above-mentioned rule matches all the problem instances that have value 1 at the 7th bit (i.e., D_6). The resultant response of the system is the opposite of the 6-bit parity concept.

Another important rule from the 7-bit parity problem is shown in Fig. 12. This is also an experienced (iteration 111038) and accurate rule with numerosity 6, and specificity 1. The only CF in the condition part has two elements, that is, condition bit #6 and NOT operator. This CF is "not of condition bit #6," that is, NOT(D_6). The CF in the action part has only one element which is "6-bit parity." This rule matches all the problem instances that have the value "0" at the 7th bit. The resultant response of the system is the same as that of the 6-bit parity concept. These two rules cover all the instances of the 7-bit parity problem. By evolving these compact rules, the lateralized system effectively converted the 7-bit parity problem into a two-bit parity problem, as shown in Fig. 13. Consequently, the LateralXCS efficiently learned the 7-bit parity problem by utilizing a very small number of problem instances. Therefore, it is plausible that LaterXCS can

D_0	D_1	D_2	D_3	D_4	D_5	D_6	P6	P7	Rule1 if ~(~D_6) -> P7=~P6		Rule2 If ~D_6 -> P7 = P6	
									Applicable	Output	Applicable	Output
0	0	0	0	0	0	0	0	0	×	-	✓	0
0	0	0	0	0	0	1	0	1	✓	1	×	-
0	0	0	0	0	1	0	1	1	×	-	✓	1
0	0	0	0	0	1	1	1	0	✓	0	×	-
0	0	0	0	1	0	0	1	1	×	-	✓	1
-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-
1	1	1	1	1	1	0	0	0	×	-	✓	0
1	1	1	1	1	1	1	0	1	✓	1	×	-

Fig. 13. Logical interpretation of two experienced and accurate 7-bit parity problem rules (see rule R1 Fig. 11 and rule R2 Fig. 12).

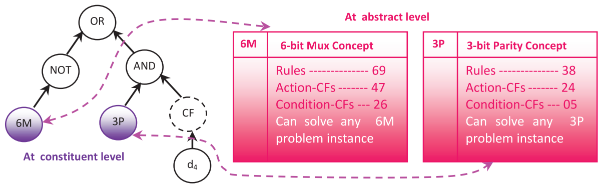

Fig. 14. Tree representation for CF Id: 1218D. 6M, 3P, and d_4 represent 6-bit Mux concept, 3-bit parity concept, and condition bit #4, respectively.

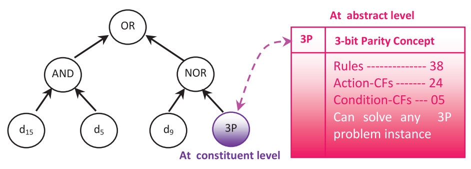

Fig. 15. Tree representation for action CF Id: 08D36. 3P, and d_5, d_9, d_{15} represent 3-bit parity concept, and condition bits #5, #9, #15, respectively.

solve any scale n-bit parity problem given successively scaled problems.

Hierarchical Boolean problems are challenging due to an additional layer of complexity, low sparsity, and hierarchical distribution of knowledge. It is necessary to apply heterogeneous BBKs to resolve a hierarchical Mux problem. The learned concept of 18-bit HMux problem consists of 3202 rules, 2347 condition-CFs, and 679 action-CFs. An example rule from 18-bit HMux is shown in the supplementary material (see S-Fig. 14). This is an experienced (iterations 7700) and accurate rule with numerosity 12, and specificity 6. There are 6 CFs in the condition of the rule. The tree representations for condition CF (Id: 1218D) and action CF (Id: 08D36) are shown in Figs. 14 and 15, respectively. A close observation of this rule reveals that, along with other BBKs, it has 6-bit Mux and 3-bit parity concepts in its condition and 3-bit parity in its action. These concepts act at the constituent level when they are inside the CFs to resolve hidden problem instances. In contrast, they act at the holistic level when tasked to independently resolve any respective problem instance of 6-bit Mux problem or 3-bit parity problem. This shows that the system has the ability to identify and utilize the critical BBKs, from the learned knowledge pool, to resolve the hidden layers of the problem.

V. Discussion

This work is designed to solve problems where a presented instance of a problem can be constructed from instances of subproblems. The system considers (addresses) the problem at two different levels (constituent level and holistic level) simultaneously. As the problem scales and the knowledge pool grows, there will be more and more candidate constituent subproblems, which does slow down the behavior. However, the system has the ability to identify (without a human-in-the-loop) the relevant subproblems for a large problem that has constituents parts. Consequently, it scales much more quickly than systems that do not consider subproblems.

It is noted that a concept requires a large number of online instances to form, relative to what might be required in supervised learning. However, the novel system uses reinforcement learning to support the plausibility of applying the lateralized approach to multistep problems, for example, path planning.

In complex problems, the additional processing of the lateralized system is justified by its ability to automatically (without a human-in-the-loop) identify and use the relevant BBKs to efficiently solve a given problem. Furthermore, the novel system can be qualitatively evaluated on the transparency and understandability of the evolved solution. As the decision-making process of the novel system is interpretable (see Section IV-E), the system is a step toward explainable artificial intelligence.

Alternative systems that are able to use learned subcomponents fail to achieve the final step of efficiently integrating knowledge that is represented at different levels. For example, cooperative coevolution methods link subpopulations of learned subcomponents but do not allow mating between individuals in heterogeneous subpopulations [57]. Consequently, individual parts of the system cannot efficiently solve the whole problem because they cannot mate across levels of abstraction.

The novel lateralized system is essentially different from the ensemble approach. The majority of ensemble systems obtain an accurate prediction, by using multiple constituent learning algorithms, which can be obtained by utilizing the single best constituent algorithm [58]–[60]. It is important to note that there is no guarantee that the performance accuracy of an ensemble system is always better than the performance accuracy of the best constituent algorithm [61]. On the other hand, if a lateralized system finds a holistic-level BBK that has the ability to independently resolve the given problem, it generates an inhibit signal to stop further processing of other system components and utilizes only that holistic BBK to solve the future problem instances. Consequently, the performance accuracy of a lateralized system is always better than or equal to the performance accuracy of an individual BBK.

Another similar methodology is Granular computing that has been applied to recognize regularities, present at different levels of abstraction, in the data [62], [63]. This approach makes an effective use of the granular structure to represent the same problem at different levels of abstraction and from different viewpoints [64]. In granular computing, a level of abstraction consists of granules that have similar nature or granularity; it is essential to identify an effective level of granularity concerning the given problem *a priori*. The hierarchical representation can only be used for the intended problem and cannot be used for other problems [65]. The novel lateralized approach, in contrast, allows the integration of heterogeneous

BBKs to solve one specific level of abstraction. It also has the ability to consider a given problem at multiple levels of abstraction, without human-in-the-loop. Furthermore, it has the ability to avoid extraneous computations by generating inhibit and excite signals.

The majority of deep learning-based systems store knowledge in multiple layers such that all features are treated equally in each layer. These systems generate a homogeneous knowledge representation that cannot be reused elsewhere in the system [2]. Consequently, these systems generate a huge/deep network of homogeneous knowledge in order to learn complex (hierarchical) problems and do not take advantage of the potential to transfer knowledge between levels in the hierarchy. It is important to note that transfer learning takes whole layer sections from one system and applies them to another system. But that is not the same as (re)using information multiple times within the same system. In contrast, a lateralized system generates a heterogeneous knowledge representation at different levels of abstraction. It has the ability to automatically identify relevant BBKs from the heterogeneous knowledge pool and (re)utilize them at the constituent level or holistic level. This means that a holistic-level BBK learned for a lower level problem could be used/reused as a constituent level BBK for a higher level problem, either within the same system or across different systems [9], [10], [53]. For example, a lateralized system could consider eye color at the same time as the shape of the triangle formed between the eyes and nose to recognize a face in a nonlinear manner [9].

Formal concept analysis (FCA) [66], [67] is another approach that may resemble "concepts" in lateralized learning. FCA generates a hierarchy of concepts by utilizing a collection of objects and their attributes. In the FCA hierarchy, a concept consists of objects that share a set of attributes. Thus, a concept here represents the relationship between a set of objects and a set of attributes. In FCA, it is necessary to segregate attributes from objects and cast them on a conceptual scale to generate concepts [66]. However, the novel lateralized concepts are empirically (automatically) extracted from data rather than explicitly defining them, which would require a human-in-the-loop.

The lateralized approach does have limitations. It may not work well for optimization problems, because once the agent has solved an optimization problem it is rarely used as a constituent part for another problem. This includes bilevel optimization problems where the value of the subproblem solution depends on the higher level problem. Moreover, in order to take advantage of the lateralized approach, it is necessary to first have constituent knowledge. For example, when dealing with images the system needs to recognize salient objects at the level of constituent parts (e.g., eyes in a face or headlights in a car) to efficiently classify a face or a car. However, once a class is learned, the lateralized system can reutilize it, for example, for the identification of crowd versus motorway congestion. We believe that the inclusion of lateralization may open a new door for EML systems that can efficiently learn real-world classification problems.

VI. Conclusion

The novel system successfully applied lateralization and modular learning at different levels of abstraction to resolve complex Boolean problems. Considering the same problem at different levels of abstraction (i.e., at a constituent level and a holistic level) enables the novel system to reformulate a complex problem as a simple problem and efficiently resolve it. For example, the novel system addressed the n-bit parity problem as a two-bit problem by utilizing the learned concept of the (n-1)-bit parity problem and the one additional condition bit #n. Experimental results demonstrated that the lateralized system has the ability to identify and utilize the relevant BBKs to efficiently learn the distribution of knowledge in hierarchical Boolean problems.

This work makes the following major contributions to the field of machine learning.

1) Lateralization, enabling modular learning, is successfully applied for the first time to resolve single-step, scalable, and complex problems.
2) Novel abstraction methods are developed to represent the same problem at different levels of abstraction (i.e., constituent level and holistic level).
3) A heterogeneous knowledge representation is enabled such that different sized blocks of knowledge can be automatically (without human-in-the-loop) recombined in a recursive manner, that is, a holistic block can be (re)used as a constituent block at a higher level of abstraction. Conventional homogeneous representations struggle to achieve this feat.

Future tests will include noisy, redundant, and irrelevant information that were not included in the test domain in this article as this would have added uncertainty to the analysis. In further work, the lateralized approach will also be tested to address perceptual aliasing in multistep decision-making tasks.

Acknowledgment

The authors wish to acknowledge the use of New Zealand eScience Infrastructure (NeSI) high performance computing facilities to run our experimental work.

References

[1] D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap, "Measuring abstract reasoning in neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 511–520.

[2] M. Shanahan, K. Nikiforou, A. Creswell, C. Kaplanis, D. Barrett, and M. Garnelo, "An explicitly relational neural network architecture," 2019, *arXiv:1905.10307*.

[3] Z. Ji, H. Wang, J. Han, and Y. Pang, "SMAN: Stacked multimodal attention network for cross-modal image–text retrieval," *IEEE Trans. Cybern.*, vol. 52, no. 2, pp. 1086–1097, Feb. 2022.

[4] X. Chen, J. Yu, and Z. Wu, "Temporally identity-aware SSD with attentional LSTM," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2674–2686, Jun. 2020.

[5] A. Vaswani *et al.*, "Attention is all you need," 2017, *arXiv:1706.03762*.

[6] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3856–3866.

[7] T. Wang, A. Bezerianos, A. Cichocki, and J. Li, "Multikernel capsule network for schizophrenia identification," *IEEE Trans. Cybern.*, early access, Dec. 1, 2020, doi: 10.1109/TCYB.2020.3035282.

[8] G. Konidaris, "On the necessity of abstraction," *Current Opinion Behav. Sci.*, vol. 29, pp. 1–7, Oct. 2019.

[9] A. Siddique, W. N. Browne, and G. M. Grimshaw, "Lateralized learning for robustness against adversarial attacks in a visual classification system," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 395–403.

[10] H. A. Shehu, A. Siddique, W. N. Browne, and H. Eisenbarth, "Lateralized approach for robustness against attacks in emotion categorization from images," in *Proc. Int. Conf. Appl. Evol. Comput. (Part EvoStar)*, 2021, pp. 469–485.

[11] A. S. Alexander and D. A. Nitz, "Retrosplenial cortex maps the conjunction of internal and external spaces," *Nat. Neurosci.*, vol. 18, no. 8, pp. 1143–1151, 2015.

[12] D. A. Nitz, "Spaces within spaces: Rat parietal cortex neurons register position across three reference frames," *Nat. Neurosci.*, vol. 15, no. 10, pp. 1365–1367, 2012.

[13] M. C. Corballis, "The evolution of lateralized brain circuits," *Front. Psychol.*, vol. 8, p. 1021, Jun. 2017.

[14] J. L. Krichmar, "The neuromodulatory system: A framework for survival and adaptive behavior in a challenging world," *Adapt. Behav.*, vol. 16, no. 6, pp. 385–399, 2008.

[15] W. H. Gaddes and D. Edgell, *Learning Disabilities and Brain Function: A Neuropsychological Approach*. New York, NY, USA: Springer, 2013.

[16] M. Dharmaretnam and L. J. Rogers, "Hemispheric specialization and dual processing in strongly versus weakly lateralized chicks," *Behav. Brain Res.*, vol. 162, no. 1, pp. 62–70, 2005.

[17] L. J. Rogers, P. Zucca, and G. Vallortigara, "Advantages of having a lateralized brain," *Proc. Roy. Soc. London B, Biol. Sci.*, vol. 271, no. S6, pp. S420–S422, 2004.

[18] D. Jackson and A. P. Gibbons, "Layered learning in Boolean GP problems," in *Proc. Eur. Conf. Genet. Program.*, 2007, pp. 148–159.

[19] P. Stone and M. Veloso, "Layered learning," in *Proc. Eur. Conf. Mach. Learn.*, 2000, pp. 369–381.

[20] D. O'Neill, H. Al-Sahaf, B. Xue, and M. Zhang, "Common subtrees in related problems: A novel transfer learning approach for genetic programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2017, pp. 1287–1294.

[21] J. F. Miller, "Cartesian genetic programming," in *Proc. Cartesian Genet. Program.*, 2011, pp. 17–34.

[22] J. F. Miller and S. L. Smith, "Redundancy and computational efficiency in cartesian genetic programming," *IEEE Trans. Evol. Comput.*, vol. 10, no. 2, pp. 167–174, Apr. 2006.

[23] M. Iqbal, W. N. Browne, and M. Zhang, "Extending learning classifier system with cyclic graphs for scalability on complex, large-scale boolean problems," in *Proc. Annu.Conf. Genet. Evol. Comput.*, 2013, pp. 1045–1052.

[24] I. M. Alvarez, W. N. Browne, and M. Zhang, "Human-inspired scaling in learning classifier systems: Case study on the n-bit multiplexer problem set," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 429–436.

[25] M. Iqbal, W. N. Browne, and M. Zhang, "Reusing building blocks of extracted knowledge to solve complex, large-scale Boolean problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 465–480, Aug. 2014.

[26] M. V. Butz and W. Stolzmann, "An algorithmic description of ACS2," in *Proc. Int. Workshop Learn. Classifier Syst.*, 2001, pp. 211–229.

[27] M. V. Butz, *Rule-Based Evolutionary Online Learning Systems*. Berlin, Germany: Springer, 2006.

[28] M. Horiuchi and M. Nakata, "Self-adaptation of XCS learning parameters based on learning theory," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 342–349.

[29] R. Chandra, A. Gupta, Y.-S. Ong, and C.-K. Goh, "Evolutionary multitask learning for modular knowledge representation in neural networks," *Neural Process. Lett.*, vol. 47, no. 3, pp. 993–1009, 2018.

[30] M. T. Banich and R. Compton, *Cognitive Neuroscience*. Cambrige, U.K.: Nelson Educ., 2010.

[31] L. C. Robertson and M. R. Lamb, "Neuropsychological contributions to theories of part/whole organization," *Cogn. Psychol.*, vol. 23, no. 2, pp. 299–330, 1991.

[32] A. V. Flevaris and L. C. Robertson, "Spatial frequency selection and integration of global and local information in visual processing: A selective review and tribute to Shlomo Bentin," *Neuropsychologia*, vol. 83, pp. 192–200, Mar. 2016.

[33] M. Faust and Y. N. Kenett, "Rigidity, chaos and integration: Hemispheric interaction and individual differences in metaphor comprehension," *Front. Human Neurosci.*, vol. 8, p. 511, Jul. 2014.

[34] S. Frässle, F. M. Paulus, S. Krach, S. R. Schweinberger, K. E. Stephan, and A. Jansen, "Mechanisms of hemispheric lateralization: Asymmetric interhemispheric recruitment in the face perception network," *Neuroimage*, vol. 124, pp. 977–988, Jan. 2016.

[35] Y.-C. Chan *et al.*, "Towards a neural circuit model of verbal humor processing: An fMRI study of the neural substrates of incongruity detection and resolution," *Neuroimage*, vol. 66, pp. 169–176, Feb. 2013.

[36] S. K. Yelle and G. M. Grimshaw, "Hemispheric specialization for linguistic processing of sung speech," *Perceptual Motor Skills*, vol. 108, no. 1, pp. 219–228, 2009.

[37] D. E. Stark *et al.*, "Regional variation in interhemispheric coordination of intrinsic hemodynamic fluctuations," *J. Neurosci.*, vol. 28, no. 51, pp. 13754–13764, 2008.

[38] P. Hoffman, J. L. McClelland, and M. A. L. Ralph, "Concepts, control, and context: A connectionist account of normal and disordered semantic cognition," *Psychol. Rev.*, vol. 125, no. 3, pp. 293–328, 2018.

[39] M. A. L. Ralph, E. Jefferies, K. Patterson, and T. T. Rogers, "The neural and computational bases of semantic cognition," *Nat. Rev. Neurosci.*, vol. 18, no. 1, pp. 42–55, 2017.

[40] A. Martin, "GRAPES—Grounding representations in action, perception, and emotion systems: How object properties and categories are represented in the human brain," *Psychonomic Bull. Rev.*, vol. 23, no. 4, pp. 979–990, 2016.

[41] M. A. L. Ralph, "Neurocognitive insights on conceptual knowledge and its breakdown," *Philos. Trans. Roy. Soc. London B, Biol. Sci.*, vol. 369, no. 1634, 2014, Art. no. 20120392.

[42] T. T. Rogers *et al.*, "Structure and deterioration of semantic memory: A neuropsychological and computational investigation," *Psychol. Rev.*, vol. 111, no. 1, pp. 205–235, 2004.

[43] T. T. Rogers and J. L. McClelland, *Semantic Cognition: A Parallel Distributed Processing Approach*. Cambridge, MA, USA: MIT Press, 2004.

[44] J. Holland and J. Reitman, *Cognitive Systems Based on Adaptive Algorithms Reprinted In: Evolutionary Computation. The Fossil Record.* New York, NY, USA: IEEE Press, 1998.

[45] T. H. Hoang, R. I. McKay, D. Essam, and N. X. Hoai, "On synergistic interactions between evolution, development and layered learning," *IEEE Trans. Evol. Comput.*, vol. 15, no. 3, pp. 287–312, Jun. 2011.

[46] L. Rodriguez-Coayahuitl, A. Morales-Reyes, and H. J. Escalante, "Structurally layered representation learning: Towards deep learning through genetic programming," in *Proc. Eur. Conf. Genet. Program.*, 2018, pp. 271–288.

[47] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Proc. Eur. Conf. Genet. Program.*, 2000, pp. 121–132.

[48] L. Bull and T. Kovacs, "Foundations of learning classifier systems: An introduction," in *Foundations of Learning Classifier Systems*. Berlin, Germany: Springer, 2005, pp. 1–17.

[49] T. Kovacs, "Evolving optimal populations with XCS classifier systems," M.S. thesis, Dept. School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 1996.

[50] T. B. Nguyen, W. N. Browne, and M. Zhang, "Improvement of code fragment fitness to guide feature construction in XCS," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 428–436.

[51] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, MA, USA: MIT Press, 1998.

[52] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.

[53] A. Siddique, W. N. Browne, and G. M. Grimshaw, "Frames-of-reference-based learning: Overcoming perceptual aliasing in multistep decision-making tasks," *IEEE Trans. Evol. Comput.*, vol. 26, no. 1, pp. 174–187, Feb. 2022.

[54] A. Siddique, W. N. Browne, and G. M. Grimshaw, "Learning classifier systems: Appreciating the lateralized approach," in *Proc. Genet. Evol. Comput. Conf. Comput.*, 2020, pp. 1807–1815.

[55] A. Siddique, M. Iqbal, and W. N. Browne, "A comprehensive strategy for mammogram image classification using learning classifier systems," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 2201–2208.

[56] M. V. Butz, M. Pelikan, X. Llorà, and D. E. Goldberg, "Automated global structure extraction for effective local building block processing in XCS," *Evol. Comput.*, vol. 14, no. 3, pp. 345–380, 2006.

[57] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.

[58] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, Jul. 1999.

[59] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, 3rd Quart., 2006.

[60] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 1–39, 2010.

[61] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 942–956, Jun. 2005.
[62] Y. Yao, "Artificial intelligence perspectives on granular computing," in *Granular Computing and Intelligent Systems*. Berlin, Germany: Springer, 2011, pp. 17–34.
[63] W. Xu and W. Li, "Granular computing approach to two-way learning based on formal concept analysis in fuzzy datasets," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 366–379, Feb. 2016.
[64] J. T. Yao, A. V. Vasilakos, and W. Pedrycz, "Granular computing: Perspectives and challenges," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1977–1989, Dec. 2013.
[65] Y. Yao, "Three-way decision and granular computing," *Int. J. Approx. Reason.*, vol. 103, pp. 107–123, Dec. 2018.
[66] A. K. Sarmah, S. M. Hazarika, and S. K. Sinha, "Formal concept analysis: Current trends and directions," *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 47–86, 2015.
[67] S. Yevtushenko, "Computing and visualizing concept lattices," Ph.D. dissertation, Dept. Fachbereich Informatik, Technische Universität, Darmstadt, Germany, 2004.

Abubakar Siddique (Member, IEEE) received the B.Sc. degree in computer science from Quaid-i-Azam University, Islamabad, Pakistan, in 2003, the M.Sc. degree in computer engineering from U.E.T Taxila, Taxila, Pakistan, in 2008, and the Ph.D. degree in computer engineering from the Victoria University of Wellington, Wellington, New Zealand, in 2021.

His undergraduate senior project was conducted in an internship at Ultimus, where his work was deployed at the company's Workflow product. He spent nine years with Elixir Pakistan, Islamabad, a California-based software company. His last designation was a Principal Software Engineer where he led a team of software developers. He developed enterprise-level software for customers, such as Xerox, IBM, and Finis. He is a Postdoctoral Research Fellow with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His main research lies in lateralized systems based on artificially intelligent techniques, particularly evolutionary computation, to provide efficient solutions for challenging and complex problems in different domains, such as Boolean, computer vision, and navigation. Lateralization, inspired by the principles of biological intelligence, is an advanced form of transfer learning which utilizes feature extraction, feature construction, and dimensionality reduction. The decision-making process of a novel lateralized system is interpretable which is a step toward explainable/trustworthy AI.

Dr. Siddique was the recipient of VUWSA Gold Award and the "Student of the Session" Award during the Ph.D. and bachelor studies, respectively.

Will N. Browne (Member, IEEE) received the Doctorate degree in industrial learning classifier systems (LCSs) from the University of Wales, Cardiff, U.K., in 1999.

His postdoctoral work was with the University of Leicester, Leicester, U.K., followed by eight years lecturing in Cybernetics with the University of Reading, Reading, U.K. He led the LCS theme with the Evolutionary Computation Research Group, Victoria University of Wellington, Wellington, New Zealand, from 2009 to 2021. He has recently coauthored the first textbook titled *LCSs Introduction to Learning Classifier Systems* (Springer, 2017). His research focuses on applied cognitive systems, including cognitive robotics, learning classifier systems, and modern heuristics. He currently holds the Chair with the Queensland University of Technology, Brisbane, QLD, Australia.

Prof. Browne has been the Co-Track Chair for the Evolutionary Machine Learning Tracks and provided tutorials on rule-based machine learning in ACM and IEEE conferences.

Gina M. Grimshaw received the B.Sc. degree in biochemistry from the University of Toronto, Toronto, ON, Canada, in 1987, and the Ph.D. degree in cognitive psychology from the University of Waterloo, Waterloo, ON, Canada, in 1996.

She was a Postdoctoral Fellow with the Department of Cognitive Science, University of California at San Diego, La Jolla, CA, USA, from 1996 to 1997 before taking up an academic position with California State University San Marcos, San Marcos, CA, USA. Since 2007, she has been with the Victoria University of Wellington, where she is the Director of the Cognitive and Affective Neuroscience Lab. Her research has been funded by the National Institute of Mental Health (U.S.) and the Royal Society of New Zealand Marsden Fund. She has authored over 50 refereed journal publications. Her research explores the cognitive and neural mechanisms that support cognition and emotion, with a particular focus on hemispheric specialization and interaction.

Dr. Grimshaw has been the Editor of *Laterality: Asymmetries of Body, Brain, and Cognition* since 2016. She is the Secretary of the Australasian Society for Experimental Psychology, and chaired the Society's Experimental Psychology Conference in 2011 and 2019.