Frames-of-Reference-Based Learning: Overcoming Perceptual Aliasing in Multistep Decision-Making Tasks

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

***Abstract*—Perceptual aliasing challenges reinforcement learning agents. They struggle to learn stable policies by failing to identify and disambiguate perceptually identical states in the environment that require different actions to reach a goal. As the agent often has only a local frame of reference, it cannot represent the global environment. Frame-of-reference-based learning is a feature of vertebrate intelligence that allows multiple simultaneous representations of an environment at different levels of abstraction. This enables the resolution of patterns that are made up of patterns that are made up of features. The evolutionary computation technique of learning classifier systems has shown promise in learning nested patterns in single-step domains. This work uses the frame-of-reference concept within a learning classifier system to learn stable policies in non-Markov multistep domains. Considering aliased states at a constituent level enables the system to place them appropriately in holistic-level policies. Instead of enumerating a huge search space, evolution computation empowers the novel system to evolve fitter rules and policies. The experimental results show that the novel system effectively solves complex aliasing patterns in non-Markov environments that have been challenging to artificial agents. For example, the novel system utilizes only 6.5, 3.71, and 3.22 steps to resolve Maze10, Littman57, and Woods102, respectively.**

***Index Terms*—Building blocks, cognitive neuroscience, frame of reference, learning classifier systems (LCSs), non-Markov mazes, perceptual aliasing.**

I. Introduction

PERCEPTUAL aliasing is a long-standing problem for artificial agents in applying reinforcement learning (RL) to many multistep tasks [1]–[7]. Here, the agent is assumed to perceive its local environment without access to the global worldview, which it can only construct through interaction with the environment. Aliasing occurs when the agent's internal representation confounds the external world states, i.e., when the agent's current perception is unable to discriminate environmental states, which appear identical but require different actions [1]. In such a scenario, the reinforcement for the environment instructs the agent to take a specific action in a given state. Unfortunately, when the agent encounters an aliased state, it persists in taking the same action again, which will now be reinforced differently. This inconsistency prevents the learning of stable policies, especially for multistep tasks [8]. Perceptual aliasing, therefore, diminishes the effectiveness of RL [2] and hinders its application to real-world problems [6].

RL agents can handle simple environments that do not have aliased states, e.g., Markov environments; however, they struggle in environments that have aliased states, e.g., non-Markov environments. A sample non-Markov maze environment is shown in Fig. 1. The states A and B are two aliased states. In these states, the agent's immediate sensation provides the same input signal, i.e., 00100010. But the agent needs to take different actions to optimally reach the goal state (see Section S-II in the supplementary material). Perceptual aliasing leads to non-Markov environments. A large number of approaches have been investigated to handle these perceptual aliasing problems [1], [3]–[5], [7], [9]–[20]. These techniques can solve simple non-Markov environments but cannot optimally resolve the majority of complex non-Markov environments. The limitations of these alternative techniques are presented in Section II-A.

One factor that may have hampered learning in non-Markov environments is the reliance on only a local frame-of-reference (FoR)-based on an agent's immediate perception (local viewpoint (LV)). Hence, the agent cannot apprehend the environment at the higher (big-picture) level of abstraction, which would allow it to uniquely identify aliased states. Consequently, aliased steps in a policy[1] are stored with the same weight as nonaliased steps. An aliased state is a small pattern that gets repeated in an environment, which makes it difficult to identify where it occurs locally.

These patterns can be combined with other patterns (aliased or not) to form higher level patterns and so forth. Eventually,

Manuscript received November 12, 2020; revised March 23, 2021 and June 4, 2021; accepted July 19, 2021. Date of publication August 4, 2021; date of current version January 31, 2022. This work was supported by the Science for Technological Innovation National Science Challenge, New Zealand. *(Corresponding author: Abubakar Siddique.)*

Abubakar Siddique is with the Evolutionary Computation Research Group, 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/TEVC.2021.3102241.

Digital Object Identifier 10.1109/TEVC.2021.3102241

[1]A policy, like a route, can be considered as a large pattern prescribing transitions from a starting state to the goal state, (see Section III-B).

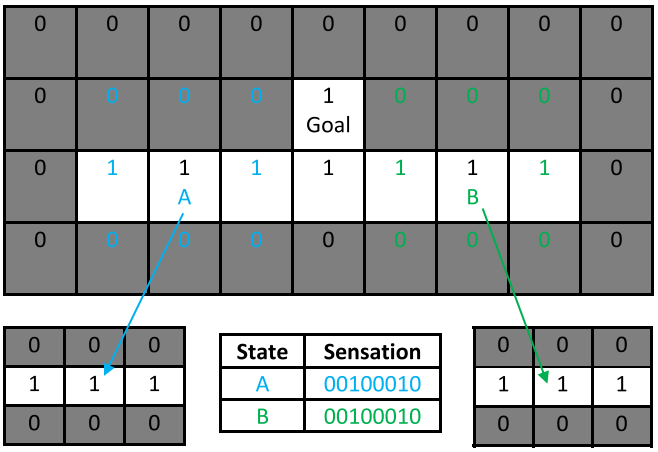

Fig. 1. Sample non-Markov maze with two aliased states.

each pattern, either in itself or as part of a higher level pattern, is unique. Thus, non-Markov environments entail patterns that may form hierarchical patterns, such as multiple aliased states at different positions in the environment. These aliased states can be identified uniquely (i.e., turned to nonaliased states) by considering the environment at different levels of abstraction. Conventional RL systems struggle to capture such complex structures.

Another problem faced by an artificial agent in multistep environments is the huge search space to discover good policies. There are many building blocks of knowledge (BBKs) that could be used to provide LVs. It is not feasible to enumerate them all. Therefore, evolutionary computation (EC) is needed to find the best BBKs that provide the fittest rules. These learned BBKs also need to be placed appropriately in holistic-level policies. A large number of policies can be created due to the possibility of nonoptimum, e.g., cyclical, routes. Again, enumerating such a search space is not feasible: EC is needed to find the best policies (combinations of BBKs) without doing an enumerated search.

A hypothesized solution is to develop an EC-based learning system inspired by the biological FoRs. In biological intelligence, a FoR is used to represent an environment from a specific viewpoint, e.g., LV (from the agent's perspective) or world viewpoint (WV) (a "birds-eye" view of the complete map) [21], [22] (see Section II-B). FoRs enable vertebrate (and many invertebrate) brains to process the same information at multiple levels of abstraction [23], [24]. Moreover, FoRs are utilized to generate a grid map of the environment [25]. This grid map is utilized by the brain's coordinate mechanism for spatial navigation. An artificial agent localizes itself by utilizing a local FoR. If it fails to uniquely identify its state (i.e., it is in an aliased state), it will utilize a higher level FoR. The agent will keep expanding to its FoRs until it disambiguates the aliased states. This nonaliased representation is then considered the WV.

Thus, a system is needed that can store representations of a state at different levels of abstraction and learn how these can be formed into a hierarchy to describe the patterns in a problem. An evolutionary machine learning (EML) system is capable of detailed learning of individual features, and abstract learning of the patterns of features [26], [27]. Moreover, an EML system has the ability to identify and reuse a learned building block in similar parts of an environment. However, existing EML systems do not have the ability to distinguish these learned building blocks to uniquely identify similar parts of an environment located at different positions. This ability is critical for an EML system to reduce the enumeration of the search space in a multistep environment.

We hypothesize that incorporating FoRs into an EML system could allow it to overcome current limitations and learn to solve problems in non-Markov environments. In contrast to conventional systems that do not differentiate between detailed and abstract learning, our novel EML system is anticipated to solve problems in non-Markov environments by representing knowledge in both constituent and holistic frames of reference. In order to create the novel EML system, the learning agent first needs to automatically identify the level of abstraction that is required to successfully turn an aliased state into a nonaliased state. Then, inspired by the brains' grid map, an adjacent states map (ASM) needs to be created for an environment. This ASM will be utilized by the agent to differentiate aliased states based on the neighboring states.

The EML technique to be used is learning classifier systems (LCSs)[2] as they store learned knowledge in *if* <*state*> *then* <*action*> rules. The states need to be stored in a format that links them through actions, which are termed code paths (CPs) here. CPs form BBK,[3] which are useful in themselves, but crucially can be constructed together to form higher level (more abstract) BBKs. This enables the system to function at appropriate levels of abstractions. The rules can provide elementary knowledge (or LV), which is needed to form the constituent level blocks of knowledge (CPs). Simultaneously, abstract knowledge (or the WV) can be formed by combining these CPs into holistic blocks of knowledge ((sub)policies of differing length, similar to routes). A CP will have the ability to accurately handle a nonaliased state as in an ordinary rule. Multiple CPs, policies, and the ASM will have the ability to provide a WV at higher levels of abstraction, which can be used to handle an aliased state. This process will essentially turn an aliased state at the constituent level into a nonaliased state at the holistic level. The agent can then create the optimal policy to reach the goal.

A schematic illustration of a conventional approach and novel (FoRs-based) approach is shown in Fig. 2. Each state is represented by a colored circle. The multiple instances of the same color represent aliased versions of a state. The policies are represented by ellipses. A conventional EML approach (left-hand side) relies only on local FoRs (LV) and considers individual features and niches in a homogeneous manner, i.e., all the states are treated the same, hence, it does not generate unique patterns. The novel approach (right-hand side) utilizes multiple FoRs and splits a complex problem into constituent

[2]LCSs have been used as a preferred research tool to evolve solutions for a wide range of maze problems for the last 30 years [3], [5], [13], [28]–[30].

[3]A building block of knowledge is a unit of knowledge that is transferable and can be used or reused to solve a part of a problem or the whole problem.

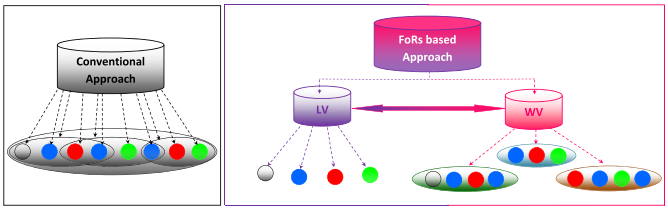

Fig. 2. Schematic illustration of a conventional approach and novel (FoRs-based) approach.

and holistic knowledge. The LV identifies states at the constituent level; the WV places them appropriately in policies at a holistic level to generate unique patterns.

A. Goals

This work aims to create a novel FoRs-based system, inspired by the principles of animals' navigation, for decision making in state-transition learning. The system is to provide optimal solutions for non-Markov environments by utilizing FoRs that enable heterogeneous knowledge representations at different levels of abstraction. To achieve this goal, we set the following objectives.

1) Create a novel FoRs-based system that has the ability to process a single input at different levels of abstraction to represent an LV (constituent knowledge) and a WV (holistic knowledge to keep parallel construction with constituent knowledge) of the same state.
2) Create a heterogeneous representation of knowledge using CPs and policies. This knowledge will be utilized or reutilized at different levels of abstraction to generate constituent representation and holistic representation, which will allow interpretation of learned policies (see Section IV-C).
3) Integrate blocks of knowledge (CPs) at different levels of abstraction to generate an unambiguous representation. The resultant knowledge will be used to disambiguate complex patterns of aliased states, which will enable the learning of stable policies.
4) Create a strategy to activate/deactivate (sub)policies such that the agent can reach the goal state by using the minimum number of steps.

The remainder of this article is organized as follows. Section II discusses the state-of-the-art techniques that have been developed to address perceptual aliasing problems. Moreover, it presents the required background knowledge from cognitive neuroscience and LCSs. Section III describes how a FoRs-based system can be created. It explains the critical components and architecture of the novel system. Deterministic and non-Markov environments are then used to evaluate the developed system. The effectiveness of the novel approach to optimally solve non-Markov environments is presented in Section IV. Section V highlights the strength, drawbacks, and limitations of the novel approach. Finally, the conclusion and future work are described in Section VI.

II. Background

The goals of this section are threefold: first, to highlight the strengths and limitations of the relevant techniques that have been developed to handle perceptual aliasing problems; second, to introduce the relevant principles of biological intelligence that will inform this work; and third, to present the relevant knowledge of LCSs that provide a foundation upon which the novel system will be developed.

A. Relevant Approaches

A stochastic environment can be considered Markov if the agent's immediate perception provides all the necessary information to *decide* the best action in all situations/states. Such decision processes are called Markov decision processes (MDPs). An environment can be considered non-Markov if the agent's immediate perception does not provide all the necessary information to decide the best action in all situations/states. Such hidden states can be *aliased states*, which are only partially observable and the agent needs more information to take the best action. Such decision processes are called partially observable MDPs (POMDPs) [31]. The majority of RL agents can easily learn Markov environments, but they struggle to learn non-Markov environments. Many techniques have been developed to address perceptual aliasing in non-Markov environments. These techniques are discussed as follows.

Initial attempts to cope with perceptual aliasing by additionally learning an immediate perception could not solve the majority of non-Markov environments because of impractical and strict assumptions such as noiseless sensing, deterministic actions, and incomplete perception [1], [3], [19], [20]. Further attempts to utilize belief state methods to address perceptual aliasing failed due to two factors: 1) difficulty in calculating value functions and 2) updating belief states. These two factors make the belief state-based strategies intractable for large and complex non-Markov environments [14], [15], [32].

Another way to handle perceptual aliasing is to have a message list where states and actions are stored and passed to future decisions but this is inconvenient because of the storage size and contents. It is hard to determine the optimum memory relevant to the decisions. For example, an agent may store neutral messages and/or incorrect messages, making it hard to search for only the appropriate messages. The resultant systems could not evolve optimal rules and so were unable to solve the majority of non-Markov environments [18], [33]–[35].

Internal memory-based systems successfully solve simple non-Markov environments; however, they struggle to disambiguate aliased states in complex non-Markov environments. This is due to the likelihood of path entrainment, i.e., becoming stuck in a local optimum as once a route to the goal has been discovered, it could be exploited such that the exploration of potentially better routes does not occur. Moreover, multiple versions of the same route may be stored due to a lack of generalization. Consequently, internal memory-based systems are neither robust nor able to achieve optimal performance for complex problems [13]. Thus, these systems provide only ad hoc solutions [3], [18]. Similarly, internal action-table-based systems struggle to obtain an optimal policy for complex non-Markov environments. Moreover, they require extra computations to find an appropriate policy [10], [11].

Artificial agents based on the psychological principle of *anticipatory behavioral control* provide a partial solution for non-Markov environments [4], [16]. Latent learning-based anticipatory classifier systems (ACSs), and the more advanced ACS2, predict the next state of the environment, but still struggle to learn complex aliasing patterns. The methodology adopted by ACSs fails to identify the aliased states in the majority of cases. Consequently, the agent is unable to take the required best action [4], [13], [16], [17]. Even ACS2, with behavioral sequences, fails to efficiently address non-Markov environments. For example, they struggle to address environments with loops in states and environments that have the same aliased states on one path [16]; see discussion about solving loops of states in Section III. Recently, Orhand *et al.* [28] presented a study that implemented behavior sequences in ACS2. The resultant systems, BACS2 and BACS3, performed well on the majority of mazes; however, these systems were unable to counteract all of ACS2's inherent issues to effectively resolve complex environments, such as Maze10, Littman57, and Woods102 (see Section IV).

Zatuchna and Bagnall applied the idea of imprinting (adopted from psychology and ethology) to resolve non-Markov environments. They incorporated memory mechanisms and associative perception techniques in a single system to address aliased states. Moreover, they modified the evolutionary and reinforcement mechanisms of the standard LCSs. The resultant system, named *AgentP*, has a complex architecture that relies on tailored methods, such as memory and imprinting. AgentP observes and stores many states as it progresses, which eventually renders the learning intractable. Moreover, it inherits the above-mentioned limitations and drawbacks of ACS2 and memory-based systems. Consequently, it is still unable to completely learn complex non-Markov environments, such as Maze10 [5], [13].

Deep RL is another approach to address the perceptual aliasing problem. Deep RL agents require high computational resources to handle perceptual aliasing. This becomes worse in complex non-Markov environments, e.g., Minecraft [7], in which many states have common visual features. Interactive machine learning (IML)-based techniques have been proposed to improve the performance and reduce the extraneous computations utilized by the RL agents to handle perceptual aliasing in non-Markov environments [36]. However, these techniques could not be generalized and required a human-in-the-loop to assist the agent. Moreover, it is hard to decide *when*, e.g., triggering based on low confidence and action advice/critique, and *how much*, e.g., how often and level of expertise, human intervention is appropriate [7], [9], [12].

Another approach to handle the perceptual aliasing problem is the use of subgoals [6], [37], [38]. However, it is hard to select an appropriate number of subgoals and define the complexity of the subgoals. Consequently, the resultant systems have poor learning efficiency. Recently, a genetic algorithm (GA)-based strategy has been developed to find appropriate subgoals [6]. However, this technique requires extraneous computations and cannot find appropriate subgoals if the system fails to achieve the task at the initial population generation.

In sum, although methods to address perceptual aliasing do exist, they either fail under challenging circumstances or entail unreasonable computational overhead. What is needed is a novel FoRs-based system that can identify aliased states at a constituent level and place them uniquely in holistic level policies.

B. Functional Organization in Vertebrate Brains

Vertebrate brains efficiently solve complex problems due to functional architecture that allows them to represent sensory information at different levels of abstraction. It is not the aim of this work to simulate the specific architecture of a specific species; rather to take inspiration from basic principles of vertebrate intelligence. We focus here on the functional organization in vertebrate brains, especially with respect to navigation. The aspects of vertebrate intelligence that are relevant to this work are explained as follows.

1) Representation and Processing: In vertebrate brains, the same sensory information is represented and processed by different regions at different scales, i.e., constituent level or holistic level. For example, in animals' navigation, an *egocentric* FoR represents an LV, whereas *allocentric* and *route-centric* FoRs represent the WV of the environment [21], [22], [39], [40]. Information related to the animal's head direction as well as motor and sensory actions associated with navigation are logged via an egocentric FoR; information related to external cues and boundaries of the environment is logged via an allocentric FoR; information related to the route states (spaces), distances between those states, and a series of actions on the planned route is logged via a route-centric FoR [39], [41]–[43]. This ability to represent the same environmental state at different levels of abstraction will be incorporated in this work (Obj. 1) and 2)).

2) Knowledge Integration: Most animals learn and store a route (*policy*) while navigating an environment. A specific path, or route, can be described as a sequence of turns connected by straight segments of different lengths. Animals follow a specific route while moving from one position to another in an environment. The knowledge of a specific route allows animals to make correct decisions during navigation. A route has a particular shape and different routes may have similar/different shapes at different scales. Different cortical and subcortical structures of the brain utilize FoRs to log information that is used to represent these routes. For example, the position of an animal within a route is represented by utilizing egocentric and route-centric FoRs. This information is logged by the posterior parietal cortex and retrosplenial cortex neurons [21], [40], [44]. The specific location of a route within an environment is represented by utilizing allocentric FoR. This information is logged by the hippocampal place cells [45]–[47]. Similarly, a grid map of an environment is represented by utilizing all three FoRs, i.e., egocentric, route centric, and allocentric. This information is logged by the grid cells of the entorhinal cortex [25]. This map is used by the brain's coordinate mechanism for spatial navigation. Different brain regions coordinate with each other to integrate the learned knowledge and assist the animal to exhibit

intelligent behavior during spatial navigation [21], [40], [44], [47]–[49]. This integration of knowledge and the route-based strategy for navigation will be applied in the CPs and policies created here (Obj. 2) and 3)).

3) Goal-Driven Processing: Vertebrate brains have the ability to activate the most suitable and relevant region to perform computations required for a specific task. This activation is controlled by the goal-driven processes that analyze the problem and switch control to the appropriate region. For example, egocentric and allocentric processing have been associated with right and left posterior cortex, respectively, [50]; dorsal stream and frontal areas are active during egocentric coding, whereas dorsal and ventral regions are involved during allocentric coding [51], [52]; similarly, sequential organization of consecutive choices is managed by the left hippocampus, whereas allocentric or map-based navigation is handled by the right hippocampus [53]. This ability to determine which computational structure is the most suitable with respect to the task is important for real-life problems. A strategy will be developed for the activation/deactivation of the most suitable and relevant policy with respect to the situation (Obj. 4)).

C. Learning Classifier Systems

LCSs are an EC methodology for developing rule-based solutions to complex problems. Instead of enumerating a search space, LCSs learn heterogeneous patterns (niches) in the search space by applying learning components and discovery algorithms. The first LCS was developed by Holland and Reitman [54], named "Cognitive System One" to acknowledge its foundation in cognitive neuroscience. An LCS-based artificial agent endeavors to identify innate patterns in the given environmental data. It learns by applying an adaptive strategy to take the best action that maximizes its current or future rewards [55], [56].

The reward prediction p for a classifier rule in single-step problems is updated as follows:

$$p_{i+1} \longleftarrow p_i + \beta(r_{i+1} - p_i) \text{ where } 0 \leq \beta \leq 1 \tag{1}$$

where r is a reward returned from the environment. Q-learning has been used as a standard RL method for the propagation of reward in multistep RL problems. It is computed as follows:

$$\mathcal{Q}(s_t, a_t) \leftarrow \mathcal{Q}(s_t, a_t) + \beta\Big(r_{i+1} + \gamma.\max_a \mathcal{Q}(s_{t+1}, a) - \mathcal{Q}(s_t, a_t)\Big) \tag{2}$$

where s, a, t, γ, and $\mathcal{Q}$ represent state, action, time, discount, and value of an action, respectively. This equation propagates the reward from the goal state back down the chain of classifiers to the originating state. In multistep problems, the final goal state may not be reached in the next step. Therefore, it is necessary to include the proportion of the reward obtained on the next iteration. Thus, the single-step reward prediction (1) is modified for multistep problems as follows:

$$p_{i+1} \leftarrow p_i + \beta\Big(r_{i+1} + \gamma.\max_a p_{[A]_{+1}} - p_i\Big) \text{ where } 0 \leq \beta \leq 1 \tag{3}$$

where $[A]$ is an action set. This equation can be implemented programmatically by saving the previous action set $[A]_{-1}$ such that its updates are conducted by utilizing the current prediction array as the future values. Moreover, for the goal states, the updates occur in $[A]$ (for details see [31], [57]).

LCSs apply GAs to generate new rules and explore unique niches in the problem space. LCSs evolve a collection of rules that collectively provide knowledge of the environment. The agent applies this knowledge in a piecewise manner to make predictions [58]. A classifier's rule has a fitness value based on its contribution toward the solution. As learning progresses, LCSs produce fitter rules, improve the fitness of good rules, and remove the weak and redundant rules. A brief introduction of LCSs and GA, for readers unfamiliar with these base techniques, is presented in the supplementary material (see Section S-I).

1) Code Fragments: A code fragment (CF) is an encoding technique that has been introduced in LCSs to achieve high-level knowledge representation. It facilitates LCSs in creating transferable BBKs [59]. A CF is a genetic programming (GP)-like tree that has operations in the internal nodes and environmental variables or other CFs in the leaf nodes. CF-based LCSs can solve complex problems, e.g., 135-bit and n-bit multiplexer (Mux) problem [59]–[61]. A novel CP (similar to CF)-based technique, for multistep rather than single-step domains, will be developed to disambiguate aliased states.

III. Frames-of-Reference-Based System

This work develops a FoRs-based system for decision making to resolve non-Markov environments. We first introduce two novel components that are used to achieve heterogeneous knowledge representation at different levels of abstraction, i.e., the CP (constituent knowledge), and the policy of CPs (holistic knowledge). These techniques are assisted by a novel ASM strategy that provides a snapshot of the environment. Subsequently, the utilization of CPs and their policies for the identification and disambiguation of aliased states is described. Finally, the overall strategy adopted by the novel system to resolve non-Markov environments is presented.

A. Code Paths

A CP is a GP-like tree (similar to a CF [59]) that encodes state–action–state sequences. Its format is a binary tree with depth up to two. Consequently, a CP can have a maximum of seven nodes, i.e., four states linked by three actions (see Fig. 3). This limit is set to keep the tree size bounded to avoid intractable learning problems. A CP acts as a constituent level BBK such that a single-step CP provides an egocentric (local) viewpoint, whereas a multistep CP or multiple CPs provide an allocentric viewpoint of the environment.

A state is an environmental input instance and a version is its unique identity. All states have a default version of 0. The agent disambiguates aliased states (S) by assigning them different versions (V). For example, the states S_1 and S_2 are two different egocentric states (i.e., different observations on the local scale). S_{1,V_0}, S_{2,V_1}, and S_{2,V_2} are three different allocentric states (i.e., different from a WV); V_0 indicates a state

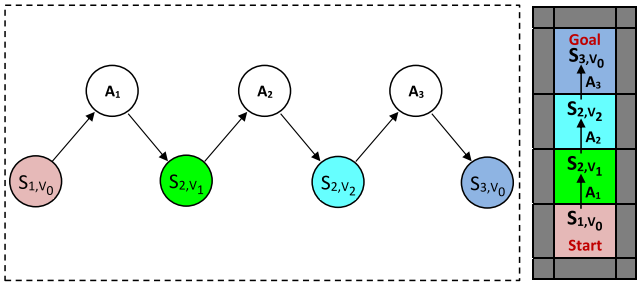

Fig. 3. Sample maze (right) and the corresponding code path (left) when an agent moves from the start state (10000000 encoded—conventionally clockwise in a grid from the top; with 1 open, 0 blocked) to the goal state (00001000). With LCS perceived states S_{2,V_1} (10001000, 1) and S_{2,V_2} (10001000, 2) are nonaliased versions of the aliased state S_{2,V_0} (10001000).

with no known aliasing; and $V_1, V_2, \ldots, V_n$ indicate aliased versions of the state S_i.

Let $\mathcal{S}$ be a set of N_s states, $\mathcal{S} = \{S_i\}_{i=1}^{N_s}$, then we use "$S_{i,V_j}$" to refer to the jth version (V_j) of the ith state (i.e., S_i). Thus, the pool of states with their versions is represented as $\mathcal{SV} = \{S_{i,V_j}\}_{\forall i,j}$. Now, let $\mathcal{A}$ be a set of N_A actions, $\mathcal{A} = \{A_k\}_{k=1}^{N_A}$. Then, a CP can be defined as a state–action function that walks through states using actions

$$\text{CP} : \mathcal{SV} \times \mathcal{A} \longrightarrow \mathcal{SV}. \tag{4}$$

In practice, a CP is an alternate sequence of states and actions starting from a start state (say S_{l,V_m}) and ends at an end state (say S_{p,V_q}). For example, the CP in Fig. 3 can be represented by the sequence

$\text{CP} = <S_{1,V_0}, A_1, S_{2,V_1}, A_2, S_{2,V_2}, A_3, S_{3,V_0}>$.

Whenever an artificial agent moves from one state to another state during training, three nodes of the corresponding CP are created/updated. For example, when an agent currently in the state S_{1,V_0} executes an action A_1 and moves to another state S_{2,V_1}, the corresponding CP will insert/update these three entries in its nodes, respectively. A sample maze and the corresponding CP representing an agent that moves upward from the start state to the goal state are shown in Fig. 3.

B. Policies

A policy, similar to a route in animals' navigation, is a pattern that prescribes state transitions from a start state to the goal state. Here, a policy is comprised of multiple CPs that are used by the agent while moving from a start position to the goal. Moreover, a policy has two associated attributes—1) the number of steps used to reach the goal state (or steps) and 2) the number of times the policy successfully guided the agent to the goal state (or experience). Fig. 4 illustrates two policies (red and yellow dotted lines), and corresponding CPs. The first policy P_r (red) consists of CP-1 and CP-3, which can lead the agent to the goal state by using four steps. The second policy P_y (yellow) consists of CP-2 and CP-3, which can lead the agent to the goal state by using five steps.

Let $\mathcal{CP}$ be a set of N_{cp} code paths, $\mathcal{CP} = \{CP_k\}_{k=1}^{N_{cp}}$, then the policy P can be defined as a subset of $\mathcal{CP}$. For example, in Fig. 4, there are three code paths, $\mathcal{CP} = \{CP_1, CP_2, CP_3\}$. The policy P_r consists of two CPs and it is represented as $P_r = \{CP_1, CP_2\}$. Similarly, the policy P_y is represented as $P_y = \{CP_2, CP_3\}$. In general, the set of all N_p possible polices is represented as $\mathcal{P} = \{P_l\}_{l=1}^{N_p}$, where the goal is to find the optimal policy $P_{\hat{l}}$, which has the lowest cost, i.e., $\text{cost}(P_{\hat{l}}) \leq P_l \ \forall l = 1, \ldots N_p$. Such a cost is computed based on two associated attributes, i.e., the number of steps and experience.

The policy is a holistic level knowledge representation, which provides an allocentric viewpoint (WV) of an environment. It is created during the explore mode in two situations: 1) when an agent successfully reaches the goal or 2) when an evolutionary process is triggered. In the first scenario, the agent logs the path that is used to reach the goal. Loops are removed from the path before creating a policy. In order to remove a loop, the whole path is traversed such that if the current state (and version) already exists in the path, the horizontal and vertical distances are computed to determine whether it is the same state or a new aliased state (distance computation is presented in Section III-D case-5). If the resultant horizontal and vertical distances are zero, then it is the same state, that is, the agent has completed a loop, and so the states visited by the agent during this loop are removed from the path.

In the second scenario 2), an evolutionary process is triggered when the average number of time steps since the last crossover is greater than θ_{GA}. Two new policies are created by applying a crossover technique on two existing policies such that both policies have a common state (with the same version) in their paths. The first policy is randomly selected from the pool of existing policies. A second policy is randomly selected until one with a common state with the first policy is found. If compatible policies are found, the common state between these policies acts as a crossover point such that the new child policies are created by combining the opposite halves from the parent policies, otherwise, no new policies are created.

C. Adjacent States Map

The ASM contains information about the neighboring states within the environment. It is a sparse, dynamic map that is developed/updated at runtime while the agent explores the environment. The number of rows of the ASM is equal to the number of states visited by the agent. The columns of the ASM are equal to the number of possible actions that can be taken by the agent, plus the last column (ID) that is reserved for naming each state. Each nonheading cell contains a state-version tuple. For this work, as is common in maze navigation tasks, an agent can execute eight actions (separated by 45°) to move to a neighboring state where the action is without noise. Whenever an agent moves from one state to another, all the corresponding entries (adjacent states) in the ASM are created/updated. For example, an agent moves from a state S_{1,V_1} to another state S_{2,V_2} by executing an action A_1. Two entries in the ASM are created representing: 1) state S_{2,V_2} at column A_1 in a row against state-id S_{1,V_1} and 2) state S_{1,V_1} at column A_5 in a row against state-id S_{2,V_2}. It is important to note that the agent can move back to the original state by executing a flipped (opposite) action, e.g., A_5 is a flipped action of A_1. A section of the ASM, corresponding to this movement, is shown in Fig. 5.

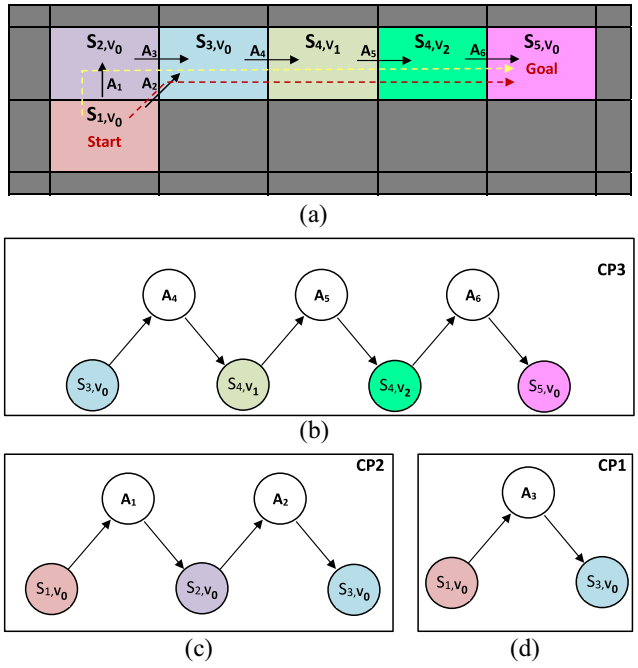

Fig. 4. (a) Sample maze and two policies. The first policy (red dotted line) consists of CP-1 and CP-3. The second policy (yellow dotted line) consists of CP-2 and CP-3. (b) Three-step CP (CP-3) from S_{3,V_0} to S_{5,V_0}. (c) Two-step CP (CP-2) from S_{1,V_0} to S_{3,V_0}. (d) Single-step CP (CP-1) from S_{1,V_0} to S_{3,V_0}. A_i can take any action value between 0 and 8.

A_0		A_1		A_2		A_3		A_4		A_5		A_6		A_7		ID	
0	0	S_2	V_2	0	0	0	0	0	0	0	0	0	0	0	0	S_1	V_1
0	0	0	0	0	0	0	0	0	0	S_1	V_1	0	0	0	0	S_2	V_2

Fig. 5. Example section of an ASM.

D. Aliasing Identification and Disambiguation

The most challenging step in learning a non-Markov environment is identifying and disambiguating the aliased states, as local knowledge provides conflicting information. Learning is achieved by comparing and integrating different levels of CPs. The single-step CPs represent the egocentric viewpoint, whereas multistep CPs represent the allocentric viewpoint. Multistep CPs and policies are used, along with the ASM, to generate the WV (complete map) of the environment. During this process, new CPs are created and incompatible CPs are updated or deleted. There are five distinct cases that can lead to the identification of an aliased state.

1) Case-1: In a specific state, the agent effects a previous action but *transitions to an unexpected state*. For example, the agent moves from state S_{1,V_0} to state S_{3,V_0} by executing an action A_1, but another CP already exists, moving from state S_{1,V_0} to state S_{2,V_0} by executing the same action A_1. Thus, state S_{1,V_0} is marked in the ASM as an aliased state and is disambiguated into two states, here, termed state S_{1,V_1} and state S_{1,V_2}. Consequently, the agent is only confident about its transitions due to the actions A_1 and flipped-A_1 (i.e., A_5). The transitions due to all other actions now become ambiguous because the agent cannot discern the correct nonaliased version (S_{1,V_1} or S_{1,V_2}).

The agent updates the unambiguous knowledge and deletes the ambiguous knowledge. For this purpose, all the CPs with entries "$S_{1,V_0} \xrightarrow{\text{A1}} S_{2,V_0}$" are updated with "$S_{1,V_1} \xrightarrow{\text{A1}} S_{2,V_0}$." Similarly, the CPs with flipped action (A_5) entries "$S_{2,V_0} \xrightarrow{\text{A5}} S_{1,V_0}$" are updated with "$S_{2,V_0} \xrightarrow{\text{A5}} S_{1,V_1}$." The corresponding entries in the ASM and multistep CPs are also updated. Moreover, all the entries in CPs from S_{1,V_0} that transition to another state by executing an action other than A_1 are deleted as they are no longer reliable. Similarly, the flipped entries from another state to S_{1,V_0} with an action other than A_5 are deleted. The corresponding information is also deleted from the ASM and multistep CPs. Consequently, the agent updates learning and, most importantly, learns to forget. Finally, two new CPs for the move "$S_{1,V_2} \xrightarrow{\text{A1}} S_{3,V_0}$" and "$S_{3,V_0} \xrightarrow{\text{A5}} S_{1,V_2}$" are created. For this newly disambiguated aliased state S_{1,V_2}, a new row in the ASM is created and corresponding information is added to that row.

2) Case-2: The agent ends in the same state from the same action, but *transitioned from an unexpected starting state*. For example, the agent moves from a state S_{3,V_0} to another state S_{2,V_0} by executing an action A_2 but another CP already exists that transitions from state S_{1,V_0} to state S_{2,V_0} by executing the same action A_2. The state S_{2,V_0} is marked as an aliased state and is disambiguated into two states, here, termed state S_{2,V_1} and state S_{2,V_2}. Subsequently, all the CPs with entries "$S_{1,V_0} \xrightarrow{\text{A2}} S_{2,V_0}$" are updated with "$S_{1,V_0} \xrightarrow{\text{A2}} S_{2,V_1}$." Similarly, the CPs with flipped action "$S_{2,V_0} \xrightarrow{\text{A6}} S_{1,V_0}$" are updated with "$S_{2,V_1} \xrightarrow{\text{A6}} S_{1,V_0}$." The corresponding entries in the ASM and multistep CPs are also updated. Moreover, all the entries in CPs from any state to S_{2,V_0} by executing an action other than A_2 are deleted as unreliable. Similarly, the flipped entries from S_{2,V_0} to another state with an action other than A_6 are deleted. The corresponding information is also deleted from the ASM and multistep CPs. Consequently, the agent updates learning and learns to forget. Finally, two new CPs for the move "$S_{3,V_0} \xrightarrow{\text{A2}} S_{2,V_2}$" and "$S_{2,V_2} \xrightarrow{\text{A6}} S_{3,V_0}$" are created. For this newly disambiguated aliased state S_{2,V_2}, a new row in the ASM is created and corresponding information is added to that row.

3) Case-3: The agent does not effect the previous action, but *a new action that transitions to the same state*. For example, the agent moves from state S_{2,V_0} to another state S_{1,V_0} by executing an action A_3 but another CP already exists from state S_{2,V_0} to state S_{1,V_0} by executing a different action $\neq A_3$. The state S_{1,V_0} is marked as an aliased state and is disambiguated into two states, here, termed state S_{1,V_1} and state S_{1,V_2}. Subsequently, all the CPs with entries "$S_{2,V_0} \xrightarrow{\neq A3} S_{1,V_0}$" are updated with "$S_{2,V_0} \xrightarrow{\neq A3} S_{1,V_1}$." Similarly, the CPs with flipped action "$S_{1,V_0} \xrightarrow{\neq A7} S_{2,V_0}$" are updated with "$S_{1,V_1} \xrightarrow{\neq A7} S_{2,V_0}$." The corresponding entries in the ASM and multistep CPs are also updated. Moreover, all the entries in CPs from S_{1,V_0} to another state by executing an action other than $\neq A7$ are deleted. Similarly, the flipped entries from another state to S_{1,V_0} with an action other than $\neq A3$ are

	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
Angle	0	45	90	135	180	225	270	315
X-Axis	0	1	1	1	0	-1	-1	-1
Y-Axis	1	1	0	-1	-1	-1	0	1

Fig. 6. Horizontal and vertical distances against actions. The action A_0 is at 0° and each subsequent action is separated by 45°.

deleted. The corresponding information is also deleted from the ASM and multistep CPs. Consequently, the agent updates learning and learns to forget. Finally, two new CPs for the move "$S_{2,V_0} \xrightarrow{\text{A3}} S_{1,V_2}$" and "$S_{1,V_2} \xrightarrow{\text{A7}} S_{2,V_0}$" are created. For this newly disambiguated aliased state S_{1,V_2}, a new row in the ASM is created and corresponding information is added to that row.

4) Case-4: The agent effects an action and transitions to a new state; however, a CP already exists that *effects a flipped action from the new state that transitions to a different state*. For example, the agent moves from state S_{1,V_0} to another state S_{2,V_0} by executing an action A_4 such that no other CP already exists from state S_{1,V_0} to any other state by executing the same action A_4. Subsequently, two CPs for this move need to be created, i.e., "$S_{1,V_0} \xrightarrow{\text{A4}} S_{2,V_0}$" and "$S_{2,V_0} \xrightarrow{\text{A0}} S_{1,V_0}$." During the creation of CP with flipped action, i.e., "$S_{2,V_0} \xrightarrow{\text{A0}} S_{1,V_0}$," it is found that a CP already exists from state S_{2,V_0} to another state S_{3,V_0} by executing the same action A_0. The state S_{2,V_0} is marked as an aliased state and is disambiguated into two states, here, termed state S_{2,V_1} and state S_{2,V_2}. Subsequently, all the CPs with entries "$S_{2,V_0} \xrightarrow{\text{A0}} S_{3,V_0}$" are updated with "$S_{2,V_2} \xrightarrow{\text{A0}} S_{3,V_0}$." Similarly, the CPs with flipped action "$S_{3,V_0} \xrightarrow{\text{A4}} S_{2,V_0}$" are updated with "$S_{3,V_0} \xrightarrow{\text{A4}} S_{2,V_1}$." The corresponding entries in the ASM and multistep CPs are also updated. Moreover, all the entries in CPs from S_{2,V_0} to another state by executing an action other than A_0 are deleted. Similarly, the flipped entries from other states to S_{2,V_0} with an action other than A_4 are deleted. The corresponding information is also deleted from the ASM and multistep CPs. Consequently, the agent updates learning and learns to forget. Finally, two new CPs for the move "$S_{1,V_0} \xrightarrow{\text{A4}} S_{2,V_2}$" and "$S_{2,V_2} \xrightarrow{\text{A0}} S_{1,V_0}$" are created. For this newly disambiguated aliased state S_{2,V_2}, a new row in the ASM is created and corresponding information is added to that row.

5) Case-5: An aliased state already exists in a path. This case is executed only if the agent cannot identify an aliased state by utilizing any of the above-mentioned cases. The agent logs its path during the navigation. If the current state (and version) already exists in the path, the agent computes the horizontal and vertical distance to determine whether it is in the same state (i.e., identifies a loop) or a new aliased state. For this work, the action A_0 is at 0° and each subsequent action is separated by 45°. The distance values against all the actions are presented in Fig. 6. To compute the distance, the following procedure is applied. For each action executed by the agent, it moves to a new state, a value ($+1$, -1 or 0) is added to the horizontal and vertical distances. For example, against action A_0 the values 0 and 1 are added to the x-axis and y-axis, respectively. These values are added for all the states between the current state and the already visited state by traversing through the path. Finally, if the agent determines that it has traveled a nonzero horizontal or vertical distance, it marks its state as a new aliased state and disambiguates it by assigning a new version.

E. Predict Aliased Version

The prediction of the correct aliased version (V) of a state (S) is an important step in resolving non-Markov environments. The agent disambiguates aliased states by assigning them unique versions, as an aliased state with a unique version behaves as a nonaliased state. The agent is confident about its position in the environment if it is in a nonaliased state or an aliased state with the correct version. Consequently, the agent moves to the next state by executing the appropriate action that leads to the optimal path to the goal. The techniques adopted by the agent to predict the aliased version are explained as follows.

1) Case-1: The *agent is confident about its position in the environment*, i.e., either it is in a nonaliased state or an aliased state with the correct version. When the agent moves from a confirmed state to an aliased state, it extracts the correct version of the aliased state from the CPs. For this purpose, if a CP exists that has the same initial $state - version \xrightarrow{\text{action}} state$ entries, the version of the current aliased state is set from that CP.

2) Case-2: The *agent is not confident about its position in the environment*, i.e., either a random starting point is an aliased state or there exist multiple CPs with this same state but different versions. Consequently, the agent is in an aliased state with a default version 0. When the agent moves from such a state to another aliased state, it applies one of the following techniques to predict the aliased version.

First, if the current path of the agent has more than two states, it finds matching multistep CPs by comparing only the states (not versions) and actions with the last three moves (because a CP can support at most three moves; this number could be extended at the cost of additional computations). If such a CP exists, the agent considers the aliased state version of that CP as a candidate version. To verify that candidate version, the agent searches for the flipped CP, i.e., from the current state to the previous state. Subsequently, the agent counts all the CPs that have the same flipped action but result in different states. If this number is equal to the total number of aliased versions for that state, the agent flags the prediction as correct. Otherwise, the candidate version is discarded.

If the first technique does not find the aliased version, the agent identifies the flipped CPs from the current state to the previous state without comparing the versions. If the candidate CPs have multiple versions for the current aliased state, they are discarded. Otherwise, the agent finds all the CPs that have the same flipped action but different states. If the number of CPs found is equal to the total number of aliased versions for that state, the agent marks the prediction as correct. Otherwise, it is discarded. If the agent fails to make a correct prediction, the default version 0 is used, which means that the agent is not yet confident about its position in the environment.

3) Case-3: This case is executed if the *agent cannot predict an aliased state version* by utilizing the above-mentioned cases. The agent attempts to predict the aliased version by utilizing the information from the ASM. The agent makes a list of the current states (without comparing the versions) that have the previous state in their neighbors at the flipped action. These are considered candidate states. Subsequently, the agent compares the shared neighborhood of each candidate state with the previous state. If a state has the same neighboring states as the shared neighborhood, then that state is flagged as a final candidate state. At the end of this process, if there is only one final candidate state, the version of that state is considered a correct aliased version. Otherwise, the agent is not confident about its position in the environment, and the default version 0 is used.

F. Overall Strategy

EC generates the constituent rules using CFs to encode the conditions of the state and actions. Many encodable rules are not valuable, e.g., rules where actions collide with walls, so EC search improves efficiency and reduces storage. Moreover, EC is used to utilize these constituent rules to evolve CP-based policies in the form of state–action–state tuples. Again this removes redundancy/irrelevant conditions that would be kept by exhaustive search. A schematic depiction of the overall strategy developed to achieve the cognitive inspired functionality in the FoRs-based system is shown in Fig. 7. The general state–action–reward scheme of the novel system is similar to the standard multistep RL scheme in LCSs [4], [18]. The learning methodology of the novel FoRs system is developed by utilizing the framework of accuracy-based LCSs, i.e., Wilson's XCS [62]. CFs assist the novel system to link the environmental features through functional nodes. A disjunctive normal form of CFs constitute a rule, which encapsulates how well CFs link together to provide an egocentric viewpoint of the environment. As in the standard LCSs, here the rules are created by three methods, i.e., covering, crossover, and mutation. These rules are combined in a population, which enables specific niches of the problem to be combined together to solve different parts of the problem domain. The FoRs system departs from a conventional LCS in the novel use of state versions within the conditions of the classifier rules are enhanced with a state-version encoding. In order to appear in the match set, the rules need to match the condition, including the version, of the state.

At the start of the learning process, all the states (aliased and nonaliased) have version 0. The agent is randomly placed in a state in the environment. The agent obtains the version of the current state by applying methods presented in Section III-E. Subsequently, the agent determines whether it is an aliased state or not. If it is an aliased state, the agent disambiguates the aliased states by assigning them unique versions. The methods adopted by the agent for the identification and disambiguation of aliased states are presented in Section III-D.

The learning process of the agent is divided into explore and exploit modes, similar to the standard LCS process. The agent logs the path while navigating through the environment.

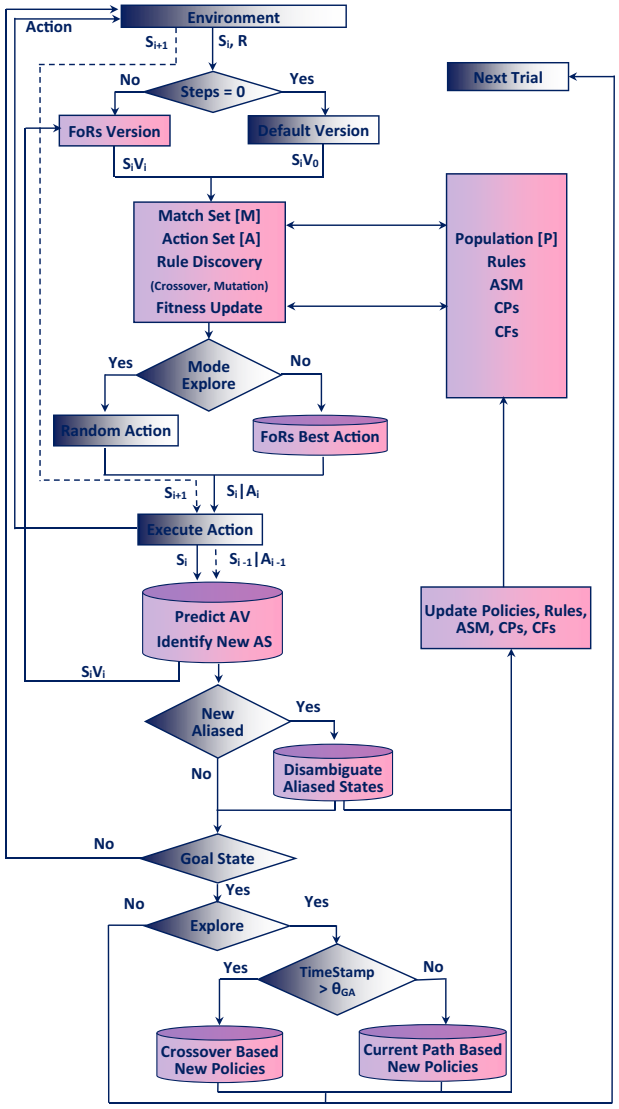

Fig. 7. Schematic depiction of the overall strategy developed to achieve cognitive-inspired functionality. The FoRs-based system identifies and disambiguates aliased states by utilizing LV (single-step CPs) and WV (multistep CPs, policies, and the ASM). The action is obtained from rules with state-version-action, and policies.

During the explore mode, the system attempts to create a new policy if one does not exist for the current path. A new policy can be created in two situations: 1) when an agent successfully reaches the goal or 2) when an evolutionary process is triggered. The method to create a new policy in the first situation is presented below, whereas the conditions to trigger the evolutionary process to subsequently create new policies are presented in Section III-B. A new policy is created if the agent successfully reaches the goal state before utilizing the maximum allowed steps, which is set with domain knowledge. For this purpose, the loops are removed from the path. Subsequently, the path is virtually traversed in reverse order, i.e., from the goal state to the start state. For each step, the ambiguous aliased versions (i.e., 0) are updated with the correct versions. New CPs are created if they do not exist, by applying the strategies explained in Sections III-D and III-E. New aliased versions may be created to disambiguate the

aliased states from the path. During this process, if no new aliased versions are created and no ambiguous version is left in the path, the new policy is created if it does not already exist. Moreover, the experience attribute of the new policy is initialized to zero and incremented by one each time the policy assists the agent to successfully reach the goal state.

The reverse traversing of the path enables the novel system to connect the isolated blocks of aliased states. If the agent cannot establish a connection (by finding or creating CPs) between the states of the path by utilizing the above-mentioned methods, the agent attempts to find blocks of aliased states that match the path at the point of missing connection (i.e., when no CP exists). Utilizing the ASM, such blocks are identified through analyzing the multistep CPs. If a block is found that matches the current path and is isolated from other neighboring states, then it is connected with the broken state by creating a new CP. An example of such a scenario is presented in the supplementary material (see Section S-III).

During exploit mode, the system activates a policy that assists the agent to take the best action. In order to select the most appropriate policy, the system must identify the correct version of the state. Therefore, for aliased states, the agent predicts the most likely aliased version by utilizing the strategies explained in Section III-E. However, if the agent is unable to predict the version, the system randomly selects one. Subsequently, the agent tries to identify the best policy for the selected state version. Each state may have more than one policy. The novel system selects a valid policy that has the smallest value for the step attribute. That policy can lead the agent to the goal state by utilizing the minimum number of steps. Finally, the agent activates the selected policy (cf. roll out).

The agent determines the best action using two different strategies: 1) the action set and 2) the active policy. If two actions are the same, the policy is marked as true by setting a flag (termed "cognate"), which can take value (true, false). The agent is confident about its decision and executes the action. However, if the actions are different, then the agent prefers the action provided by the policy and marks the cognate flag for the policy as false. Consequently, if the agent moves to a different state predicted by the policy or is unable to move, the policy is marked as malign by setting a flag (termed "malign"), which can take value (true, false), and the agent picks another best policy with respect to the current state. The malign policies will not be selected again for the current multistep run. At the end of the multistep run, all the cognate and malign flags are cleared. A walk through of this novel approach is presented in the supplementary material (see Section S-III).

IV. Experimental Work

This work seeks to demonstrate the effectiveness of the FoRs-based approach to address the perceptual aliasing problem in RL agents while solving multistep tasks in non-Markov environments. Maze problems are used as a test paradigm to investigate how agents learn state–action transitions in multistep environments. Mazes have been used in a wide variety of navigation-based research from cognitive neuroscience to artificial intelligence [4], [13], [16], [21], [45], [57], [63]–[66], as they approximately simulate real-world navigation problems. Mazes have a structure that allows experimenters to easily control and trace the behavior of an agent during the learning process. They offer a wide range of complex environments that artificial agents struggle to solve. This includes complex non-Markov mazes that are characterized by heterogeneity in action probability in a given state and clusters of such aliased states. As these characteristics make maze problems effective to evaluate the novel approach, a wide range of mazes are used as the test domain.[4] These mazes include both deterministic and non-Markov environments, all aliasing types (I, II, and III), and a broad range of complexity (1–251) [13], [28]. The majority of these mazes and the related woods environments have been used in state-of-the-art studies [4], [13], [18], [28], [67].

A. Experimental Setup

The novel system uses the XCS configuration settings that have been commonly used in XCS studies [4], [18]. The parameter values are: crossover probability $\chi = 0.8$; GA threshold $\theta_{GA} = 25$; mutation probability $\mu = 0.04$; learning rate $\beta = 0.2$; deletion fraction $\delta = 0.1$; deletion threshold $\theta_{del} = 20$; prediction error threshold $\varepsilon_0 = 10$; fitness exponent $\nu = 5$; fitness fall-off rate $\alpha = 0.1$; and fitness reduction $= 0.1$; do not care probability $= 0.33$; subsumption threshold $\theta_{sub} = 20$; prediction reward $= 1000$; prediction error reduction $= 0.25$. The agent is randomly placed at an empty position within an environment at the start of a trial. The agent is allowed to reach the goal by utilizing a maximum of 50 steps. All the results presented here are taken from the average of 30 runs.

The experimental results of the novel system (FoRsXCS) are compared with the experimental results of seven well-known benchmark systems, i.e., BACS2 [28], BACS3 [28], XCSLib [68], ACS2 [29], XCSM [18], AgentP [13], and deep recurrent Q-network (DRQN) [69], [70]. DRQN is a well-known deep learning-based system that applies a connectionist strategy to solve POMDPs environments. We were able to reproduce the experimental performance for XCSLib, ACS2, and DRQN. The results for BACS2 and BACS3 have been kindly shared by the authors. However, for the other systems (XCSM and AgentP), we have used the results reported in the respective studies.

B. Experiments

The first set of experiments was conducted for deterministic mazes (Maze5 and Maze6) to provide proof of concept for the developed FoRsXCS system. For the Maze5, ACS2 and DRQN outperformed FoRsXCS by 0.18 and 0.13 steps, respectively. But for Maze6, ACS2 outperformed FoRsXCS by 0.25 steps but the performance of FoRsXCS is better than all other systems including DRQN. It is noted that the learning

[4]A brief introduction of a maze environment and the mazes that are used in this work, for readers unfamiliar with these environments, are presented in the supplementary material (see Section S-II).

TABLE I
PERFORMANCE ACCURACY OF LCS AGENTS
LOWER IS BETTER (BEST PERFORMANCE IS IN BOLD)

	XCSM	AgentP	BACS2	BACS3	XCSLib	ACS2	DRQN	FoRsXCS
Maze5	-	-	-	-	5.18 ± 0.31	**4.61 ± 0.06**	4.66 ± 0.18	4.79 ± 0.07
Maze6	-	-	-	-	36.48 ± 0.87	**5.20 ± 0.05**	8.07 ± 1.12	5.45 ± 0.06
Maze7	10.0	**4.3**	4.34 ± 0.07	4.32 ± 0.06	31.73 ± 0.75	26.20 ± 1.30	4.45 ± 0.18	4.33 ± 0.05
Maze10	35.0	7.87 ± 7.43	26.09 ± 17.50	8.17 ± 1.36	40.44 ± 0.64	36.27 ± 0.81	9.75 ± 0.75	**6.51 ± 0.21**
MazeB	-	-	4.0 ± 0.17	4.09 ± 0.17	4.31 ± 0.05	4.73 ± 0.19	3.80 ± 0.13	**3.51 ± 0.04**
MazeD	-	-	3.0± 0.20	2.87± 0.11	2.88 ± 0.03	2.77 ± 0.03	2.83 ± 0.15	**2.74 ± 0.03**
MazeF1	-	-	-	-	1.80 ± 0.0	1.80 ± 0.02	1.80 ± 0.08	**1.79 ± 0.02**
MazeF2	-	-	-	-	2.50 ± 0.0	**2.49 ± 3.37**	3.01 ± 0.13	**2.49 ± 0.03**
MazeF3	-	-	-	-	**3.37 ± 0.01**	**3.37 ± 0.04**	3.62 ± 0.14	**3.37 ± 0.04**
MazeF4	-	4.5	4.51 ± 0.09	**4.49 ± 0.09**	33.63 ± 0.84	28.16 ± 1.12	5.06 ± 0.16	**4.49 ± 0.06**
Littman57	-	4.82 ± 5.41	4.52 ± 0.33	4.79 ± 0.49	11.95 ± 0.74	15.84 ± 1.35	5.60 ± 0.34	**3.71 ± 0.08**
Littman89	-	3.8	4.29 ± 0.21	4.29 ± 0.19	6.07 ± 0.25	8.34 ± 0.60	6.13 ± 0.29	**3.78 ± 0.05**
Woods101	3.2	**2.9**	3.03 ± 0.08	3.02 ± 0.07	8.04 ± 0.56	8.48 ± 0.35	3.31 ± 0.13	**2.90 ± 0.02**
Woods101-$\frac{1}{2}$	-	**3.1**	3.23 ± 0.20	3.20 ± 0.15	29.94 ± 0.60	22.95 ± 0.75	3.73 ± 0.15	**3.10 ± 0.02**
Woods102	3.5	3.3	3.73 ± 0.24	3.85 ± 0.23	28.97 ± 0.98	13.07 ± 0.50	3.49 ± 0.11	**3.22 ± 0.03**

rate of the novel system is slower than ACS2 (see Section S-IV in the supplementary material). This is understandable because the novel system has to create heterogeneous BBKs to create the local and WVs of the environment. Thus, the FoRXCS may be computationally inefficient for solving simple deterministic mazes.

The second set of experiments was conducted for non-Markov mazes to evaluate the effectiveness of the novel approach. A comparison of experimental results with different state-of-the-art studies is presented in Table I. The experimental results show that the novel FoRsXCS system either outperformed all other systems or showed similar behavior in solving all the non-Markov mazes except Maze7. For Maze7, AgentP and BACS3 outperformed the novel system by 0.03 and 0.01 steps, respectively.

The experimental results show that the novel FoRsXCS system effectively solves complex aliasing patterns in mazes that have been most challenging to artificial agents. For example, the novel system utilizes 6.51, 3.71, and 3.22 steps to resolve Maze10, Littman57, and Woods102, respectively. In contrast, none of the existing systems behave effectively in all these mazes. The minimum steps required by the best existing systems are 7.87 (AgentP), 4.52 (BACS2), and 3.30 (AgentP) to solve Maze10, Littman57, and Woods102, respectively. It is important to note that the well-known DRQN successfully solves deterministic and non-Markov mazes but the novel FoRsXCS outperformed DRQN in all the mazes. The reasons for the performance efficiency of the novel FoRsXCS are explained in Section IV-C. The varied learning pace of the novel system for different mazes is presented in the supplementary material (see Section S-IV).

The Wilcoxon signed-rank test was applied to statistically compare FoRsXCS with DRQN (see Table II). The test was conducted on the results of the last 100 trials. The second and third columns contain the average performance along with standard deviation. FoRXCS statistically outperformed DRQN on all mazes, all $ps < 0.00001$, which is evidence that the performance advantage of FoRsXCS is statistically significant.

The novel FoRsXCS can utilize multiple viewpoints at different levels of abstraction, depending on the complexity of aliased patterns in the environment. This functionality adds

TABLE II
WILCOXON SIGNED-RANK TEST

Problem Domain	DRQN	FoRsXCS	Z-Value	P-Value
Littman57	5.60 ± 0.34	3.71 ± 0.08	-5.5109	<0.00001
Maze10	9.75 ± 0.75	6.51 ± 0.21	-5.5109	<0.00001
Woods102	3.49 ± 0.11	3.22 ± 0.03	-5.5109	<0.00001

extra computational cost. Although it is not straightforward to compare the computational cost for different systems due to operating system constraints, these systems can be compared based on the average processing time required for an agent to take a step in an environment. The average single-step processing times, computed by using Maze7, for FoRsXCS and XCSLib are 326.37 and 74.56 μs, respectively. The processing time for the novel FoRsXCS is 4.4 times longer than the XCSLib. However, this cost is justified because the FoRsXCS needs on average 4.3 steps to successfully reach the goal in Maze7, whereas the XCSLib needs 31 steps. Thus, XCSLib utilizes 7.2 times more steps as compared to FoRsXCS. This shows that the overall computational cost of the XCSLib is greater than that of the FoRsXCS. Furthermore, the FoRsXCS-based agent successfully reached the goal in all trials, whereas the XCSLib-based agent did not always reach the goal.

C. Interpretation of Learning

The learning process of the FoRs-based system is interpretable. Close observation of CPs and the ASM reveals that the novel system successfully identified and disambiguated complex patterns of aliased states by evolving the relevant BBKs at different levels of abstraction. Consequently, the novel system translated a non-Markov environment into a deterministic environment.

Littman57 provides a non-Markov environment with moderate aliasing complexity. The maze and its aliased states (each state has a unique color) are shown in Fig. 8. Because FoRsXCS is a transparent technique, it is possible to walk through the learning sequence of the algorithm to resolve Littman57. This process demonstrates how new knowledge is learned, existing knowledge is updated, and learned knowledge is forgotten. For example, the agent was randomly

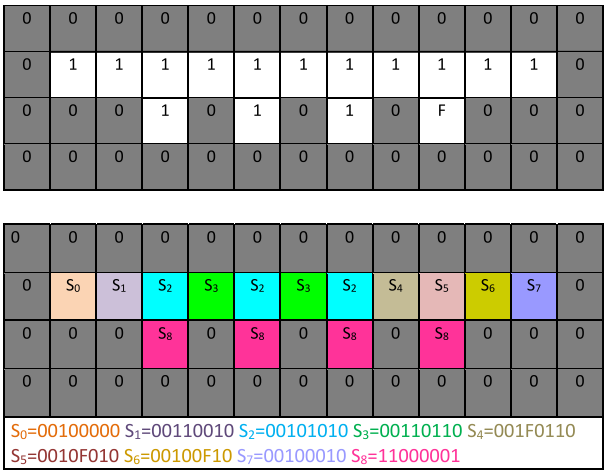

Fig. 8. Maze Littman57, 1 empty, 0 blocked, and F food/goal.

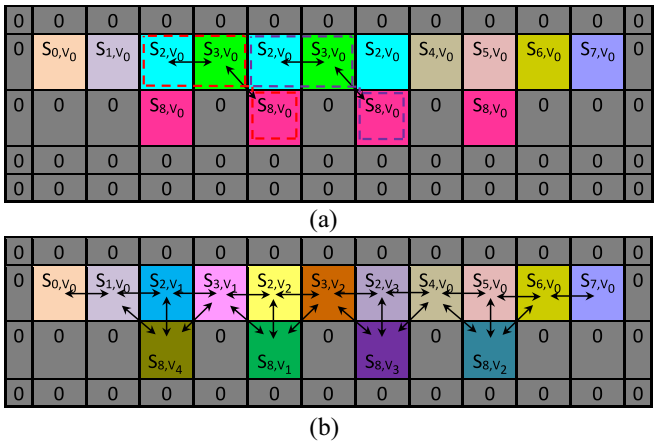

Fig. 9. FoRs-based learning for maze Littman57, here, S_{i,V_j} are colored differently. (a) Environment after creating connections among S2, S3, and S8. (b) Environment after creating connections among all the states.

placed in state S_8. It executed action $A7$ to transit to state S_3. Consequently, the agent created CPs, which provide connections between "S_{8,V_0} & S_{3,V_0}," i.e., $S_{8,V_0} \xrightarrow{\text{A7}} S_{3,V_0}$, $S_{3,V_0} \xrightarrow{\text{A3}} S_{8,V_0}$. Subsequently, the agent executed action $A6$ to transit to state S_2. Consequently, the agent created CPs, which provide connections between "S_{3,V_0} & S_{2,V_0}," i.e., $S_{3,V_0} \xrightarrow{\text{A6}} S_{2,V_0}$, $S_{2,V_0} \xrightarrow{\text{A2}} S_{3,V_0}$. These CPs form constituent BBKs and provide the egocentric view. Subsequently, the agent created two holistic level CPs, which provide connections among "S_{8,V_0}, S_{3,V_0}, and S_{2,V_0}," i.e., $S_{8,V_0} \xrightarrow{\text{A7}} S_{3,V_0} \xrightarrow{\text{A6}} S_{2,V_0}$ and $S_{2,V_0} \xrightarrow{\text{A2}} S_{3,V_0} \xrightarrow{\text{A3}} S_{8,V_0}$. These CPs form holistic BBKs and provide an abstract view. However, there are two such blocks in the maze, represented by purple- and red-dotted lines. The agent could not differentiate between these blocks. The relevant information is updated in the ASM (see Fig. 9(a)).

The agent continues to traverse the maze such that new knowledge (CPs, policies, and ASM) is created and ambiguous knowledge is deleted/updated. Ultimately, the agent successfully transformed the non-Markov environment into a deterministic environment. The resultant map of the final environment (without any ambiguous aliased states, i.e., a deterministic environment) is shown in Fig. 9(b). A complete and step-by-step walk through of the learning sequence, for the interested readers, is presented in the supplementary material (see Section S-III).

V. Discussion

The FoRs-based system is designed to address the perceptual aliasing problem in non-Markov environments. The novel system simultaneously considers the environmental instance from an LV (single-step CPs, egocentric FoR) and WV (multistep CPs, policies; allocentric and route-centric FoRs). Consequently, the learning speed of the novel system is slower than that of other state-of-the-art systems for large-scale deterministic mazes. However, the ability to consider the same problem instance at multiple viewpoints empowers the novel system to efficiently learn the complex patterns of aliased states that characterize non-Markov environments. As the problem scales in size and complexity, there will be more and more constituent-level and holistic-level BBKs, which will slow down learning. Nevertheless, the novel system has the ability to identify and disambiguate the clusters of aliased states by utilizing the BBKs at different levels of abstraction. Consequently, the novel system efficiently solves complex non-Markov mazes that homogeneous systems struggle to solve.

For this work, it is assumed that the actions always correctly affect the environment. If not, LCSs do have an error threshold that can be used to handle noise. This problem is beyond the scope of this work, but is the subject of future work. Moreover, the assumption that the agent has the freedom to explore, (e.g., flipped actions), may not be reasonable in practical situations, e.g., driving on one-way roads. The FoRs-based approach may not work well for problems, such as numerical optimization, in which constituent knowledge cannot be used or reused to solve higher level problem components.

Although it is expensive to learn constituent-level BBKs, once learned, they can be used or reused to form holistic level BBKs. The novel system applies these learned BBKs at different levels of abstraction to solve heterogeneous patterns in complex problems.

VI. Conclusion

The novel system successfully applied FoRs to learn stable policies for multistep tasks in non-Markov environments. The ability to represent the same environmental instance from different viewpoints, i.e., LV (single-step CPs, egocentric FoR) and WV (multistep CPs, policies; allocentric and route-centric FoRs), empowers the novel system to successfully address perceptual aliasing problems by identifying and disambiguating aliasing patterns. Consequently, the novel system transforms a non-Markov environment into a deterministic environment. EC played a critical role by enabling the novel system to evolve fitter rules at a constituent level and optimal policies at a holistic level. Otherwise, it was not practical to enumerate the huge search space of a complex non-Markov environment. The experiments demonstrate that the novel system has the ability to utilize or reutilize relevant learned BBKs at different levels of abstraction to learn aliasing patterns that are made up of patterns that are made up of features. The novel system effectively

solves complex aliasing patterns in the environments that have previously been challenging to artificial agents. For example, the novel system utilizes only 6.5, 3.71, and 3.22 steps to resolve Maze10, Littman57, and Woods102, respectively.

The novel system is robust against aliasing states because of its focus on the appropriate parts of the reward signal to achieve a necessary level of abstraction. Aliasing challenges existing evolutionary computing systems across a wide range of problem domains. How this approach functions with dynamic states, especially in domains with little information to start forming code paths, can now be investigated.

In further work, the novel approach will be applied to similar problem domains, such as multiclass visual classification, to prevent adversarial attacks causing misclassification. Here, constituent information, such as nose, mouth, and so forth, could be combined to holistically classify objects. The ability to consider an object at different levels of abstraction should invoke classification robustness.

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] S. D. Whitehead and D. H. Ballard, "Learning to perceive and act by trial and error," *Mach. Learn.*, vol. 7, no. 1, pp. 45–83, 1991.

[2] L. Chrisman, "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach," in *Proc. AAAI*, vol. 1992, 1992, pp. 183–188.

[3] P. L. Lanzi, "Adaptive agents with reinforcement learning and internal memory," in *Proc. 6th Int. Conf. Simulat. Adapt. Behav. (SAB)*, 2000, pp. 333–342.

[4] M. V. Butz, *Anticipatory Learning Classifier Systems*, vol. 4. New York, NY, USA: Springer, 2002,

[5] Z. V. Zatuchna and A. J. Bagnall, "AgentP classifier system: Self-adjusting vs. gradual approach," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. 2005, pp. 1196–1203.

[6] K. Suzuki and S. Kato, "Hierarchical reinforcement learning introducing genetic algorithm for POMDPs environments," in *Proc. ICAART*, 2019, pp. 318–327.

[7] S. Frazier and M. Riedl, "Improving deep reinforcement learning in minecraft with action advice," in *Proc. AAAI Conf. Art. Intell. Interact. Digit. Entertain.*, vol. 15, 2019, pp. 146–152.

[8] P. Crook and G. Hayes, "Learning in a state of confusion: Perceptual aliasing in grid world navigation," in *Proc. Towards Intell. Mobile Robots*, vol. 4, 2003.

[9] S. Krening and K. M. Feigh, "Newtonian action advice: Integrating human verbal instruction with reinforcement learning," in *Proc. 18th Int. Conf. Auton. Agents Multi Agent Sys. Syst.*. 2019, pp. 720–727.

[10] T. Hayashida, I. Nishizaki, S. Sekizaki, and H. Takeuchi, "Improved anticipatory classifier system with internal memory for POMDPs with aliased states," *Procedia Comput. Sci.*, vol. 112, pp. 215–224, Dec. 2017.

[11] T. Hayashida, I. Nishizaki, and K. Moriwake, "XCS with an internal action table for non-Markov environments," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 6, pp. 162–172, 2014.

[12] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2013, pp. 2625–2633.

[13] Z. V. Zatuchna and A. Bagnall, "Learning mazes with aliasing states: An LCS algorithm with associative perception," *Adapt. Behav.*, vol. 17, no. 1, pp. 28–57, 2009.

[14] N. Roy, G. Gordon, and S. Thrun, "Finding approximate POMDP solutions through belief compression," *J. Artif. Intell. Res.*, vol. 23, no. 1, pp. 1–40, 2005.

[15] M. W. Mitchell, "Using Markov-k memory for problems with hidden-state," in *Proc. Mach. Learn. Models Technol. Appl.*, 2003, pp. 242–248.

[16] M. Métivier and C. Lattaud, "Anticipatory classifier system using behavioral sequences in non-Markov environments," in *Proc. Int. Workshop Learn. Classifier Syst.*, 2002, pp. 143–162.

[17] W. Stolzmann, "Latent learning in Khepera robots with anticipatory classifier systems," in *Proc. 2nd Int. Workshop Learn. Classifier Syst.*, Orlando, FL, USA, 1999, pp. 290–297.

[18] P. L. Lanzi, "An analysis of the memory mechanism of XCSM," in *Proc. Genet. Program.*, vol. 98, 1998, pp. 643–651.

[19] L. Chrisman, R. Caruana, and W. Carriker, "Intelligent agent design issues: Internal agent state and incomplete perception," in *Proc. AAAI Fall Symp. Sens. Aspects Robot. Intell.*. 1991, pp. 18–25.

[20] M. Tan, "Cost-sensitive reinforcement learning for adaptive classification and control," in *Proc. AAAI*, 1991, pp. 774–780.

[21] 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.

[22] 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.

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

[24] 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.

[25] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser, "Microstructure of a spatial map in the entorhinal cortex," *Nature*, vol. 436, no. 7052, pp. 801–806, 2005.

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

[27] 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.

[28] R. Orhand, A. Jeannin-Girardon, P. Parrend, and P. Collet, "BACS: A thorough study of using behavioral sequences in ACS2," in *Proc. Int. Conf. Parall. Problem Solving Nat.*, 2020, pp. 524–538.

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

[30] W. Stolzmann, "An introduction to anticipatory classifier systems," in *Proc. Int. Workshop Learn. Classifier Syst.*, 1999, pp. 175–194.

[31] R. J. Urbanowicz and W. N. Browne, *Introduction to Learning Classifier Systems*. Heidelberg, Germany: Springer, 2017.

[32] W. S. Lovejoy, "A survey of algorithmic methods for partially observed Markov decision processes," *Ann. Oper. Res.*, vol. 28, no. 1, pp. 47–65, 1991.

[33] S. W. Wilson, "ZCS: A zeroth level classifier system," *Evol. Comput.*, vol. 2, no. 1, pp. 1–18, 1994.

[34] G. G. Robertson and R. L. Riolo, "A tale of two classifier systems," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 139–159, 1988.

[35] R. E. Smith, "Memory exploitation in learning classifier systems," *Evol. Comput.*, vol. 2, no. 3, pp. 199–220, 1994.

[36] J. A. Fails and D. R. Olsen, Jr., "Interactive machine learning," in *Proc. 8th Int. Conf. Intell. User Interfaces*, 2003, pp. 39–45.

[37] M. Wiering and J. Schmidhuber, "HQ-learning," *Adapt. Behav.*, vol. 6, no. 2, pp. 219–246, 1997.

[38] K. Suzuki and S. Kato, "Hybrid learning using profit sharing and genetic algorithm for partially observable Markov decision processes," in *Proc. Int. Conf. Network-Based Inf. Syst.*, 2017, pp. 463–475.

[39] M. V. Chafee, B. B. Averbeck, and D. A. Crowe, "Representing spatial relationships in posterior parietal cortex: Single neurons code object-referenced position," *Cerebral Cortex*, vol. 17, no. 12, pp. 2914–2932, 2007.

[40] D. A. Nitz, "Parietal cortex, navigation, and the construction of arbitrary reference frames for spatial information," *Neurobiol. Learn. Memory*, vol. 91, no. 2, pp. 179–185, 2009.

[41] C. R. Olson and S. N. Gettner, "Object-centered direction selectivity in the macaque supplementary eye field," *Science*, vol. 269, no. 5226, pp. 985–988, 1995.

[42] R. G. M. Morris, P. Garrud, J. N. P. Rawlins, and J. O'Keefe, "Place navigation impaired in rats with hippocampal lesions," *Nature*, vol. 297, no. 5868, pp. 681–683, 1982.

[43] J. O'Keefe and L. Nadel, *The Hippocampus as a Cognitive Map*. Oxford, U.K.: Clarendon Press, 1978.

[44] J. Cho and P. E. Sharp, "Head direction, place, and movement correlates for cells in the rat retrosplenial cortex," *Behav. Neurosci.*, vol. 115, no. 1, p. 3, 2001.
[45] J. O'Keefe and J. Dostrovsky, "The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat," *Brain Res.*, vol. 34, no. 1, pp. 171–175, 1971.
[46] J. O'Keefe and N. Burgess, "Geometric determinants of the place fields of hippocampal neurons," *Nature*, vol. 381, no. 6581, p. 425, 1996.
[47] E. R. Wood, P. A. Dudchenko, R. J. Robitsek, and H. Eichenbaum, "Hippocampal neurons encode information about different types of memory episodes occurring in the same location," *Neuron*, vol. 27, no. 3, pp. 623–633, 2000.
[48] L. M. Frank, E. N. Brown, and M. Wilson, "Trajectory encoding in the hippocampus and entorhinal cortex," *Neuron*, vol. 27, no. 1, pp. 169–178, 2000.
[49] D. A. Nitz, "Tracking route progression in the posterior parietal cortex," *Neuron*, vol. 49, no. 5, pp. 747–756, 2006.
[50] T. Iachini, G. Ruggiero, M. Conson, and L. Trojano, "Lateralization of egocentric and allocentric spatial processing after parietal brain lesions," *Brain Cogn.*, vol. 69, no. 3, pp. 514–520, 2009.
[51] G. Committeri, G. Galati, A.-L. Paradis, L. Pizzamiglio, A. Berthoz, and D. LeBihan, "Reference frames for spatial cognition: Different brain areas are involved in viewer-, object-, and landmark-centered judgments about object location," *J. Cogn. Neurosci.*, vol. 16, no. 9, pp. 1517–1535, 2004.
[52] T. Zaehle, K. Jordan, T. Wüstenberg, J. Baudewig, P. Dechent, and F. W. Mast, "The neural basis of the egocentric and allocentric spatial frame of reference," *Brain Res.*, vol. 1137, pp. 92–103, Mar. 2007.
[53] K. Iglói, C. F. Doeller, A. Berthoz, L. Rondi-Reig, and N. Burgess, "Lateralized human hippocampal activity predicts navigation based on sequence or place memory," *Proc. Nat. Acad. Sci.*, vol. 107, no. 32, pp. 14466–14471, 2010.
[54] J. Holland and J. Reitman, "Cognitive systems based on adaptive algorithms," in *Evolutionary Computation: The Fossil Record*. New York, NY, USA: IEEE Press, 1998.
[55] L. Bull and T. Kovacs, "Foundations of learning classifier systems: An introduction," in *Foundations of Learning Classifier Systems*. Heidelberg, Germany: Springer, 2005, p. 913.
[56] J. H. Holland *et al.*, "What is a learning classifier system?" in *Proc. Int. Workshop Learn. Classifier Syst.*, 1999, pp. 3–32.
[57] M. V. Butz, *Rule-Based Evolutionary Online Learning Systems*. Heidelberg, Germany: Springer, 2006.
[58] R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: A complete introduction, review, and roadmap," *J. Artif. Evol. Appl.*, vol. 2009, p. 1, Sep. 2009.
[59] 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.
[60] 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.
[61] M. Iqbal, W. N. Browne, and M. Zhang, "Extracting and using building blocks of knowledge in learning classifier systems," in *Proc. Annu. Conf. Genet. Evol. Comput.*, 2012, pp. 863–870.
[62] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.
[63] A. S. Alexander and D. A. Nitz, "Spatially periodic activation patterns of retrosplenial cortex encode route sub-spaces and distance traveled," *Current Biol.*, vol. 27, no. 11, pp. 1551–1560, 2017.
[64] D. A. Nitz, "Path shape impacts the extent of CA1 pattern recurrence both within and across environments," *J. Neurophysiol.*, vol. 105, no. 4, pp. 1815–1824, 2011.
[65] T. Oess, J. L. Krichmar, and F. Röhrbein, "A computational model for spatial navigation based on reference frames in the hippocampus, retrosplenial cortex, and posterior parietal cortex," *Front. Neurorobot.*, vol. 11, p. 4, Feb. 2017.
[66] P. L. Lanzi and S. W. Wilson, "Toward optimal classifier system performance in non-Markov environments," *Evol. Comput.*, vol. 8, no. 4, pp. 393–418, 2000.
[67] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Proc. Mach. Learn.*, 1995, pp. 362–370.
[68] *C++ Library for XCS*. Accessed: Mar. 26, 2020. [Online]. Available: http://xcslib.sourceforge.net/
[69] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPS," 2015. [Online]. Available: arXiv:1507.06527.
[70] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. AAAI Conf. Art. Intell.*, vol. 31, 2017, pp. 2140–2146.

Abubakar Siddique (Graduate Student Member, IEEE) received the B.Sc. degree in computer science from Quaid-i-Azam University, Islamabad, Pakistan, in 2003, and the M.Eng. degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2008. He is currently pursuing the Ph.D. degree with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand.

His undergraduate senior project was conducted in an internship with Ultimus Pakistan, Rawalpindi, Pakistan, where his work was deployed at the company's Workflow product. He spent nine years at Elixir, Islamabad, Pakistan, 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. Mr. Siddique's 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 detection, feature extraction, and feature construction. The decision-making process of a novel lateralized system is interpretable which is a step toward explainable/trustworthy AI. He is also interested in evolutionary deep learning, reinforcement, supervised, and unsupervised learning, image analysis, programming models, algorithms, runtime systems, and applications.

Mr. Siddique was a recipient of the VUWSA Gold Award and the Student Of The Session Award during his Ph.D. and bachelor's studies, respectively.

Will N. Browne (Member, IEEE) received the B.Eng. degree (Hons.) in mechanical engineering from the University of Bath, Bath, U.K., in 1993, and the M.Sc. degree in energy and the Eng.D. degree (Engineering Doctorate Scheme) from the University of Wales, Cardiff, U.K., in 1994 and 1999, respectively.

After lecturing for eight years with the Department of Cybernetics, University of Reading, Reading, U.K. He became a Professor with the Evolutionary Computation Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand, prior moving to Queensland University of Technology, Brisbane, QLD, Australia, in 2021. He has published over 120 academic papers in books, refereed international journals, and conferences. His research interests are in developing artificial cognitive systems.

Prof. Browne received the two best paper awards at the ACM Genetic and Evolutionary Computation Conference, where he has also served as the track chair on four occasions in the EML Track (or equivalent). He serves on the Editorial Board of Applied Soft Computing Journal. Together with Dr. Ryan Urbanowicz, he has authored the textbook Introduction to Learning Classifier Systems, Springer, 2017. He was the Co-Local Chair for the IEEE Congress Evolutionary Computation, Wellington, 2019.

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 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, Wellington, New Zealand, where she is an Associate Professor of Psychology and 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, and is an Editor of Laterality: Asymmetries of Brain, Cognition, and Behavior (2016 present). 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 won the university awards for Teaching Excellence, Research Excellence, and Contributions to Equity and Diversity. She has supervised over 20 postgraduate students in Psychology, Cognitive and Behavioral Neuroscience, and Engineering. She is the Secretary of the Australasian Society for Experimental Psychology, and chaired the Societys Experimental Psychology Conference (EPC) in 2011 and 2019.