

RAVEN: Extending VEsNA with Region-Agnostic Reinforcement Learning for Interpretable BDI Navigation

Zahra Daoui^{1,*†}, Andrea Gatti^{1,‡}, Angelo Ferrando^{2,§} and Viviana Mascardi^{1,§}

¹Department of Computer Science, Bioengineering, Robotics and Systems Engineering, University of Genoa, Genoa, Italy

²Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Modena, Italy

Abstract

In many real-world applications – smart buildings, service robotics, virtual environments – a cognitive agent must navigate a structured space toward its goals while justifying its decisions. Belief-Desire-Intention (BDI) agents are well suited to the second requirement because every decision rests on declarative beliefs and explicit plans; their limitation is that the plan library is fixed at design time, leaving the agent unable to adapt to environments not anticipated by the programmer.

We introduce **RAVEN** (Region-Agnostic VEsNA ENhancement), a neuro-symbolic extension of the VEsNA toolkit in which a learned navigation policy supplants manually engineered BDI plans. VEsNA stands for Virtual Environments via Natural language Agents and provides tools and libraries for connecting the brain of agents implemented in the AgentSpeak(L) language for BDI agents, with their bodies immersed in virtual environments like Unity and Godot. In RAVEN, the symbolic brain implemented in Jason, a popular interpreter for AgentSpeak(L), informs and constrains the reinforcement learning process through spatial relations encoded in its belief base.

Spatial relations are modeled in the Region Connection Calculus, RCC-8. For the experiments described in this paper, we only used the adjacency relation.

The symbolic layer preserves full control over goal specification, admissible state transitions, and the associated reward machine. Moreover, the learning problem is itself represented symbolically within the belief base, and a conventional learning algorithm is invoked as an external service, thereby ensuring that no symbolic information is directly propagated across the neural interface.

Experiments in three office environments implemented in Godot, with 11, 50, and 103 regions each, train a single policy per graph that handles all start-goal pairs without per-goal retraining, consistently reaching targets via the shortest path. Though graph-specific, the architectural pattern scales to all three sizes without changing the sub-symbolic layer. In contrast, an identical architecture without the RCC-8 symbolic mask – namely, a restriction of the set of actions available to the policy at each decision step – fails at the two larger scales. Because no symbolic content crosses the neural interface, every symbolic justification the BDI agent produces about a decision remains available unchanged.

Keywords

BDI agent, MAS, VEsNA, Neuro-Symbolic, Action Masking, Goal-conditioned Reinforcement Learning, RCC-8, Interpretability

1. Introduction

Autonomous agent navigation has long been approached through two competing paradigms, each capturing a property the other lacks.

The Belief-Desire-Intention (BDI) framework introduced by [1], realized through platforms such as Jason [2], produces agents whose behavior is expressed as explicit logical plans in AgentSpeak(L) [3]:

WOA 2026: 27th Workshop “From Objects to Agents”, June 2026, Salerno, Italy

*Corresponding author.

† This author is a PhD student; she contributed to the design of the RAVEN architecture presented in the paper, she designed and run the experiments, she drafted the first version of this document, and she contributed to its revision.

‡ This author is a PhD student; he contributed to the design of the RAVEN architecture and to the revision of this document.

§ These authors contributed equally to the design of the RAVEN architecture and to the revision of this document.

✉ zahra.daoui@edu.unige.it (Z. Daoui); andrea.gatti@edu.unige.it (A. Gatti); angelo.ferrando@unimore.it (A. Ferrando); viviana.mascardi@unige.it (V. Mascardi)

0009-0000-7217-2010 (Z. Daoui); 0009-0003-0992-4058 (A. Gatti); 0000-0002-8711-4670 (A. Ferrando); 0000-0002-2261-9926 (V. Mascardi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

every decision can be explained given the sequence of states of the belief and plan base before the decision was taken [4]. Drawing on foundational insights about what makes explanations useful to human stakeholders [5], the BDI community has progressively built a body of work on agent explainability. Early contributions demonstrated that declarative agent behavior can indeed be made transparent to non-expert users [6, 7]. Subsequent work introduced specific explanation mechanisms: valuing-based accounts [8], dialogue [9], and argumentation [10]. Contrastive explanations, shown to support better human decision-making in AI systems generally [11], have attracted growing BDI-specific attention [12, 13]. Most recently, design patterns and multi-level engineering frameworks have made explainability a first-class concern in BDI development [14, 15, 16].

A principal limitation of this approach is its brittleness. Because the plan library is specified and fixed at design time, the agent possesses minimal capacity to adapt when the environment changes or when it encounters previously unanticipated situations. Works that try to overcome this limitation exist, but they are still few [17, 18].

Deep reinforcement learning, from the Deep Q-Network [19] and its double-network successor [20] onward, offers the complementary property: the ability to discover effective policies from experience alone, with demonstrated success across domains from board games [21] to robotic manipulation [22]. The cost is opacity. A trained policy typically exhibits no readily interpretable internal structure, affords no formal guarantees regarding its behavior, and does not provide a tractable avenue for symbolic analysis or verification.

The research question we tackle in this paper is: *‘Can a BDI agent take high level control of the learning process by encoding goals, enforcing spatial constraints, and computing rewards, so that the neural component is solely responsible for policy optimization, without sacrificing the interpretability that declarative programming provides?’*

In order to take an initial step towards a positive answer to such an ambitious question, in this paper we present **RAVEN (Region-Agnostic VEsNA ENhancement)**, a neuro-symbolic navigation framework that extends VEsNA [23, 24, 25], a general-purpose agent-based framework for managing virtual environments through natural-language. VEsNA integrates a natural language interface for agent to agent and agent to human interaction (ChatBDI [26, 27]), a dynamic virtual environment supporting agents’ perception and action (Godot [28] or Unity [29]), and a cognitive framework that enables symbolic reasoning on agents’ goals, beliefs, and plans (Jason [2] or JaCaMo [30]).

We developed many examples of multiagent systems (MAS) and environments in VEsNA where cognitive agents navigate a three-dimensional environment modeled as a discrete graph of named regions, such as rooms, corridors, and common spaces, connected by qualitative spatial relations drawn from the Region Connection Calculus, RCC-8 [31]. RCC is very suitable to model geometric and spatial relations in a symbolic way, as discussed in [32]. Navigation between regions is performed entirely through predefined AgentSpeak(L) plans, leaving the agent incapable of adapting to configurations not anticipated at design time. RAVEN replaces these hand-crafted navigation plans with a learned policy while preserving the symbolic layer’s full authority over goal specification, constraint enforcement, and reward computation.

RAVEN represents one concrete instance of the NOVAE VEsNAe learning vision, presented in the recent paper “The Secret Life of Traces: A MAS Engineering Perspective” [33]. There, (Deep) Reinforcement Learning is seen as a Service, as shown in Figure 1. **NOVAE** stands for **N**euro-symb**O**lic agents that **V**erify, **l**e**A**rn and **E**xplain. In NOVAE, traces of states associated with external or internal actions and events (that we name “changers”), are the common denominator for verifying, learning, explaining, in a coherent and uniform architecture.

To enable systematic evaluation across environments of increasing complexity, RAVEN operates directly on the abstract navigation graph. Experiments have been run showing that RAVEN scales from 11 to 50 and 103 regions. Because the architectural separation that RAVEN enforces leaves every symbolic justification the agent can produce about a decision available unchanged, the design directly supports the interpretability, accountability, and trustworthiness requirements that make BDI valuable in safety-critical and human-facing settings.

Previous attempts to integrate BDI agents with reinforcement learning have predominantly focused

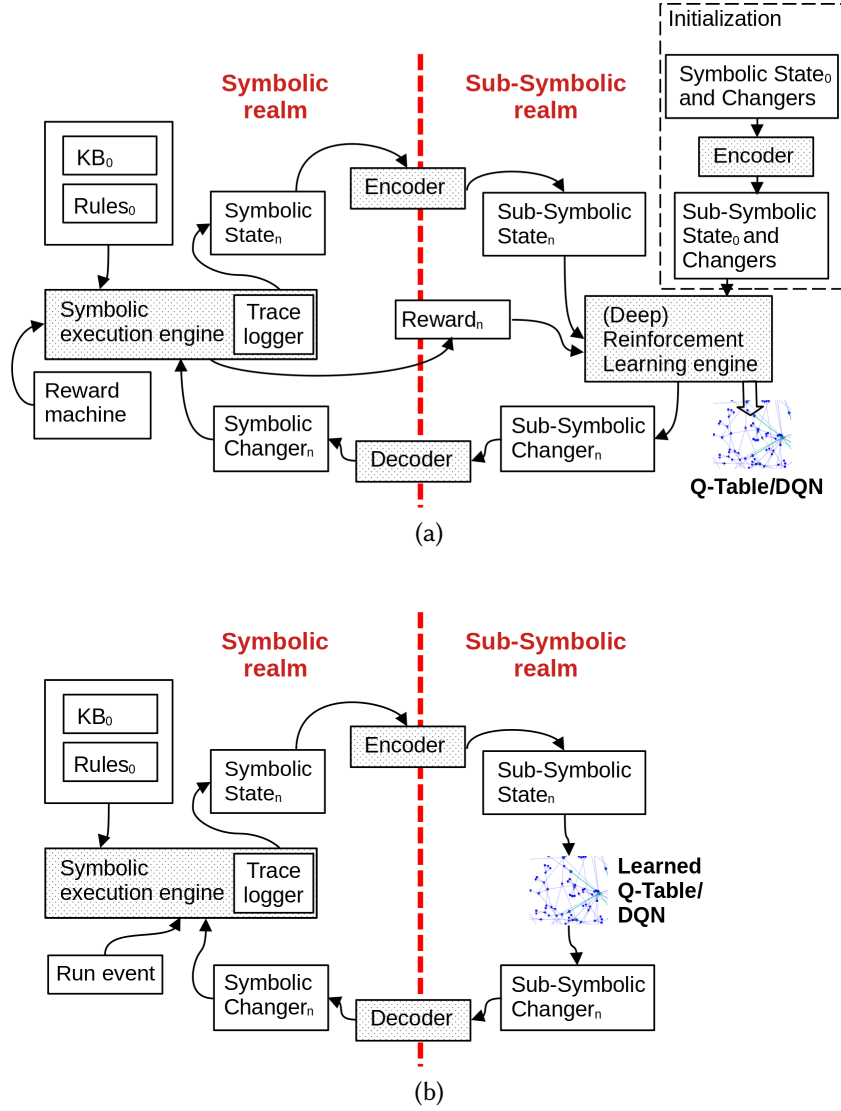


Figure 1: (Deep) Reinforcement Learning as a Service in NOVAE: (a) **exploration**, the agent is enriched with a reward machine; its states are encoded and fed to a (D)RL engine, properly initialized; the (D)RL engine outputs a changer that is decoded, and executed by the agent; a reward on the changer is computed; (b) **exploitation**, learned Q-table or Deep Q-network is used by the agent to decide the changer to execute given the current state (from “The Secret Life of Traces: A MAS Engineering Perspective” paper, 2026).

on unifying the two paradigms at the level of the agents’ internal reasoning mechanisms. In contrast, RAVEN adopts a layered architecture that assigns responsibilities in a principled manner, with a symbolic layer handling high-level deliberation and a sub-symbolic layer managing low-level learning and adaptation.

The symbolic layer, implemented in Jason, addresses goal-conditioned state representation within a unified framework, wherein a single policy is explicitly conditioned on both the agent’s current spatial region and its designated navigation target, enabling generalization across all start-goal pairs without retraining [34, 35]. It also enforces spatial constraints through action masking, restricting the set of admissible moves to those permitted by RCC-8 relations in the agent’s belief base at each step, a mechanism shown to scale substantially better than penalty-based approaches as the invalid action space grows [36, 37]. Finally, it computes the reward through a declarative reward machine embedded in the BDI belief base, so that every learning signal has a symbolic justification that can be inspected and modified without touching the neural component. Symbolic constraints do not merely ensure valid transitions; they actively guide the policy toward the shortest paths. The sub-symbolic layer,

a Deep Q-Network policy server, is responsible exclusively for policy optimization. The two layers communicate through a lightweight interface; the sub-symbolic layer never accesses goal symbols, plan names, or reward logic.

Contributions. We introduce RAVEN, a neuro-symbolic AI system in which the action mask for a deep reinforcement learning (RL) policy is derived exclusively from RCC-8 spatial-relational facts stored in a Jason belief base, without any hand-crafted, simulator-specific rules. The architecture enforces a strict separation of concerns: goal representations, admissible action sets, and reward signals are all specified exclusively at the symbolic level and are made accessible to the learning component only through this representational layer. This strict interface ensures that every justification the agent can produce about a decision remains accessible and human-readable, so that the symbolic justifications the BDI agent can already produce about its decisions remain accessible without post-hoc reconstruction.

The remainder of this paper is structured as follows. Section 2 provides a survey of related work. Section 3 describes the RAVEN architecture in detail. Section 4 reports the results of the experimental evaluation. Finally, Section 5 offers concluding remarks.

2. Related Work

2.1. BDI agent programming, VEsNA, and reinforcement learning

VEsNA provides a bridge between the BDI architecture – specifically the Jason interpreter – and situated virtual environments like Godot, enabling agents to interact via natural language. This framework has been extended in several directions: VEsNA-Pro introduces propensity-based goal and plan selection to drive emergent narrative in socially-aware agents [38], VEsNA-Unity [39] integrates VEsNA and JaCa-Unity, a toolkit bridging JaCaMo and Unity, for implementing behaviorally realistic simulations in virtual reality, while other developments have enabled agents to interact with spatial and grabbable artifacts, supporting automatic workspace switching [40]. In [41] unmodified Jason/AgentSpeak(L) agents, embodied through VEsNA in a Godot-based Formula One simulation, are shown to achieve competitive multi-car racing behavior at speed using an event-driven symbolic perception-action loop.

Moreover, many researchers have explored the integration of BDI agents and reinforcement learning, RL-BDI. A survey of this field is provided in [42]. We also conducted a systematic comparison in our recent ICAART 2025 paper [43], where we introduced SafeRLJ and evaluated it against RL-BDI research from the past decade. Unlike the analysis in [42], our review focuses on implementation details, supported by direct collaboration with the authors of key articles [44, 45, 46, 47]. The idea of integrating RL and BDI dates to the early 2000s. [48] was among the first to embed the SARSA algorithm – for State–Action–Reward–State–Action, based on Bellman equation [49] – into BDI agents using JACK Intelligent Agents, aiming to approximate human expert decision-making. Shortly after, [50, 51] proposed a hybrid BDI-RL model for vessel berthing, and [52] extended BDI agents with learning via Binary Decision Trees. Further works followed [53, 54, 55], until [44] introduced approaches closer to our SafeRLJ.

Among existing BDI-RL systems we identify three clusters relevant to RAVEN. The *first cluster* [45, 56] implements Q-learning directly as Jason plans on small grid examples (4×3 to 4×4 with four actions). The *second cluster* [57, 47] moves the learning algorithm outside the BDI interpreter: Bosello and Ricci ran exhaustive experiments across Grid World, CartPole, MountainCar and Car Intersection (a 1080-float state reduced to 20 via pre-processing, with three actions), while Ichida and Meneguzzi used Python-AgentSpeak for a travel-planning chatbot with 276-dimensional action embeddings and 100-dimensional state embeddings; both works, however, modify the standard BDI interpreter and define the reward externally. Our own work on SafeRLJ [43] can be seen as a *third cluster*. It also implements Q-learning as Jason plans, in the same way as the first cluster, but adds a *teacher-hint* mechanism for safe RL: a separate teacher agent sends the learner pre-set Q-values for specific (s, a) pairs, encoded as `ext_stact` beliefs, which override the Bellman update at runtime so that the learner avoids exploratory actions known to be unsafe.

2.2. Deep reinforcement learning for navigation

Reinforcement learning has produced strong results for discrete navigation tasks since the introduction of the Deep Q-Network [19], which demonstrated that a neural network trained with experience replay and a target network could learn stable value functions from high-dimensional inputs. The Double DQN [20] addressed the overestimation bias of the original formulation by decoupling action selection from value estimation, yielding more reliable policies in environments with discrete action spaces. Dueling network architectures [58] further improved value estimation by separating the state-value and advantage streams. These advances have been applied to navigation in grid-based and graph-based environments with considerable success [59]. However, standard deep RL navigation systems treat goal specification and reward computation as engineering choices made outside the learning loop, typically hardcoded in the environment simulator. This external design of the learning signal makes such systems opaque and difficult to verify: modifying the goal or the reward requires changes to the simulator, not to any inspectable symbolic representation. RAVEN addresses this limitation by grounding both goal encoding and reward computation in the BDI agent’s symbolic belief base, so that every aspect of the learning signal has a declarative justification that can be read and modified without touching the neural component.

2.3. Goal-conditioned reinforcement learning

Goal-conditioned reinforcement learning provides a principled framework for learning policies that generalize across multiple navigation targets by conditioning the policy on both the agent’s current state and a representation of the desired goal [34]. Rather than training a separate policy for each target, a goal-conditioned policy treats the goal as part of the input, enabling a single network to answer: given where I am and where I need to reach, which action reduces the distance to the goal? Andrychowicz et al. [35] demonstrated through Hindsight Experience Replay that sparse-reward goal-conditioned learning can be made substantially more sample-efficient by reinterpreting failed trajectories as successful demonstrations toward alternative goals. These techniques have been applied to robotic manipulation, maze navigation, and multi-task policy learning with considerable success. However, in existing goal-conditioned systems, the goal representation is constructed externally and passed to the neural component as a fixed encoding, with no connection to any symbolic description of what the goal means or how it will be verified. In RAVEN, the goal encoding is produced by the BDI symbolic layer from its own belief base, so the neural component receives a goal representation that is directly tied to the agent’s declarative intentions and can be verified against symbolic facts at any point during execution.

2.4. Action masking in reinforcement learning

Action masking addresses the practical challenge of maintaining feasibility constraints during reinforcement learning by restricting the set of actions available to the policy at each decision step. Penalty-based approaches assign large negative rewards to infeasible actions but cannot guarantee that the policy will never select them, particularly early in training when value estimates are unreliable. Masking-based approaches are more reliable: Huang and Ontañón [36] showed theoretically and empirically that assigning a value of negative infinity to invalid actions before the policy argmax preserves the convergence properties of Policy Gradient (PG) methods while eliminating infeasible selections entirely. Tang et al. [37] demonstrated the same mechanism in the context of Proximal Policy Optimization, PPO, showing that masking scales substantially better than penalty-based methods as the proportion of invalid actions grows. However, in both cases and in the broader literature, the mask is derived from external sources such as hard-coded game rules or simulator state, and there is no connection between the constraint specification and any symbolic, human-readable representation of the agent’s knowledge.

Table 1 summarizes the four dimensions along which RAVEN differs from some of the closest prior work: the reasoning cycle is left unchanged, learning is placed outside the cycle, and both the action

Table 1

Comparison with the closest prior work along four architectural dimensions. RAVEN sources both action constraints and rewards from the BDI belief base without modifying the Jason reasoning cycle.

System	BDI	Reasoning cycle	Learning location	Action constraints	Reward
Jason [2]	yes	unchanged	—	—	—
VEsNA [23]	yes	unchanged	—	RCC-8 (reas. only)	—
Edu-Jason [57]	yes	modified	in cycle (action)	none	environment
SafeRLJ [43]	yes	unchanged	in cycle (plans)	teacher hints	environment
IAM-PG [36]	no	—	standalone PG	game rules	game-defined
AM-PPO [37]	no	—	standalone PPO	simulator state	simulator
RAVEN	yes	unchanged	outside (DQN)	RCC-8 belief base	symbolic RM[†]

[†] RM: reward machine is a declarative automaton in the Jason belief base that computes the per-step penalty, the goal bonus, and timeout signal (Section 4.3).

mask and the reward are sourced from the BDI belief base.

RAVEN derives the action mask directly from the RCC-8 qualitative spatial relations maintained in the BDI agent’s belief base, making the constraint specification human-readable, formally grounded in a well-established spatial calculus, and modifiable without altering the neural component.

3. RAVEN: Architecture

The guiding principle behind RAVEN is that a BDI belief base already carries everything needed to structure a learning problem: the current state of the agent, the goal it pursues, the transitions that are spatially admissible at the current step, and the logical condition under which the goal counts as reached. The Jason reasoning cycle is left untouched, exactly as in the original VEsNA system; the learning problem is assembled from outside the cycle by reading the same beliefs the agent consults for reasoning, and the sub-symbolic component is reduced to a function approximator that optimizes a Q-function over numeric tensors with no awareness of what the tensors denote. Figure 2 illustrates the resulting pipeline, that instantiates the vision sketched in Figure 1.

3.1. Goal-conditioned Markov Decision Process over the navigation graph

We cast navigation as a goal-conditioned Markov Decision Process (GCMDP) on a discrete graph of regions. The environment is an undirected graph $\mathcal{G} = (V, E)$ whose vertices are named semantic regions (offices, corridors, common spaces) and whose edges connect pairs of regions declared adjacent through RCC-8 qualitative spatial relations [31] in the agent’s belief base. The navigation task is the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \mathcal{G}_{\text{RL}} \rangle$ with a joint state-goal space $\mathcal{S} = V \times V$, so that every state is a pair (c, g) of current and goal region; the state-dependent admissible action set $\mathcal{A}(s) \subseteq V$ is given by the RCC-8 neighbors of c ; the deterministic transition $\mathcal{T}((c, g), a) = (a, g)$ moves the agent and preserves the goal; the reward function $\mathcal{R}$ is realized by the reward machine of Section 3.2; the discount factor $\gamma \in [0, 1)$; and the goal space $\mathcal{G}_{\text{RL}} = V$. Each episode fixes a goal $g \in V$ different from the initial region c_0 , and the policy’s task is to select a sequence of admissible transitions that reaches g in as few steps as possible.

3.2. Symbolic layer

The symbolic layer is implemented in Jason and is responsible for three functions: state encoding, action masking, and reward computation. All three operate entirely on beliefs drawn from the agent’s own belief base; the sub-symbolic component is never invoked.

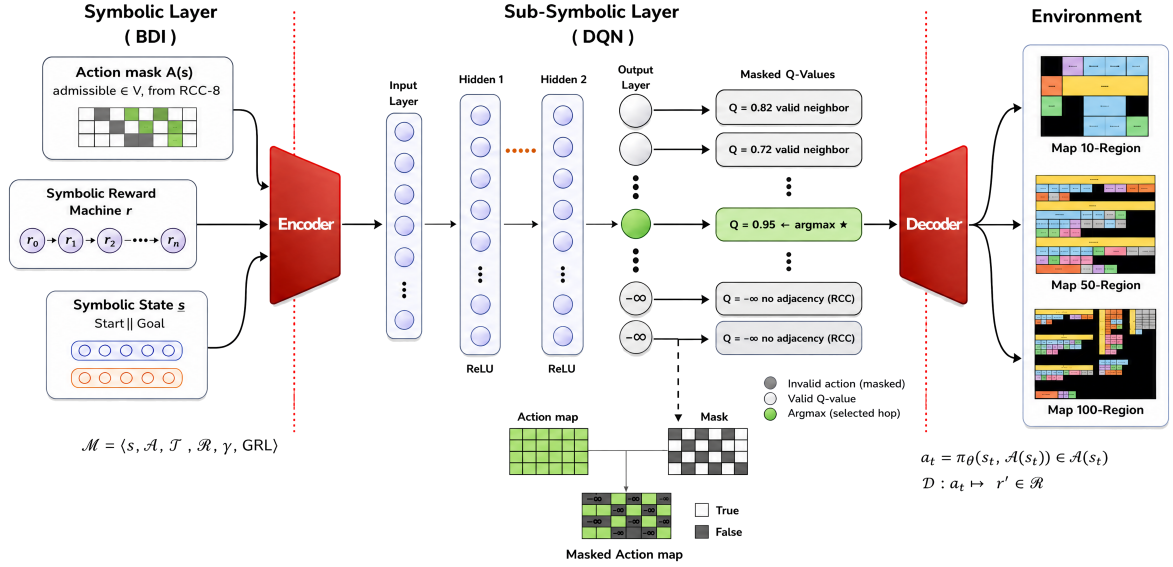


Figure 2: Overview of the RAVEN neuro-symbolic architecture. The symbolic layer (Jason) encodes the current region and the goal as a concatenated one-hot vector, computes the admissible action set $\mathcal{A}(s_t)$ from the RCC-8 facts in the belief base, and evaluates the reward machine. The sub-symbolic layer (DQN policy server) receives only these numeric artifacts, masks inadmissible transitions to $-\infty$ (False) before the ε -greedy argmax, and returns a single integer action (region) identifier. No goal symbol, plan name, or reward-computation rule crosses the interface.

Goal-conditioned state encoding. Rather than training a dedicated policy for each navigation target, the symbolic layer encodes the current region c_t and the goal region g jointly as a concatenated one-hot vector,

$$s_t = [\text{onehot}(c_t, N) \parallel \text{onehot}(g, N)] \in \{0, 1\}^{2N}, \quad (1)$$

where $N = |V|$ and $\parallel$ denotes concatenation. The first half of s_t tracks the agent’s position at the current decision step and is rewritten as the agent moves; the second half encodes the goal and remains fixed for the duration of a sub-goal, being overwritten only when the BDI agent assigns a new one. This goal-conditioned formulation [34, 35] makes the policy *region-agnostic*: a single network learns the mapping $(c, g) \mapsto a$ that selects, for any pair of regions, the neighbour of c that reduces the graph distance to g .

Two properties follow from producing the encoding inside the BDI layer. First, the goal component is grounded in the agent’s declarative intentions and can be verified against symbolic facts at any point during execution, in contrast with conventional goal-conditioned pipelines that pass an opaque external tensor to the policy. Second, the same encoding supports compositional deployment: a multi-hop mission is decomposed by the BDI agent into a sequence of sub-goals, each handled by the same trained policy without retraining (Figure 3).

Symbolic action masking. At every decision step the symbolic layer evaluates the RCC-8 relations in its belief base and returns the set of admissible successor regions $\mathcal{A}(s_t) \subseteq V$. The sub-symbolic layer applies this set as a mask on the Q-network output immediately before the policy argmax,

$$\tilde{Q}(s_t, a_i) = \begin{cases} Q_\theta(s_t, a_i) & \text{if } a_i \in \mathcal{A}(s_t), \\ -\infty & \text{otherwise,} \end{cases} \quad (2)$$

and selects $a_t = \arg \max_{a_i} \tilde{Q}(s_t, a_i)$. The mask guarantees that only spatially admissible transitions can be executed. Huang and Ontañón [36] and Tang et al. [37] have shown both theoretically and empirically that masking is substantially more reliable than penalty-based approaches as the fraction

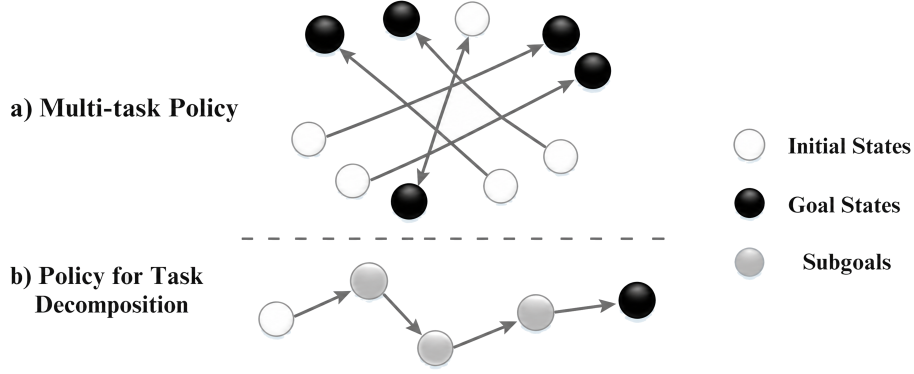


Figure 3: Goal-conditioned navigation in RAVEN. (a) A single region-agnostic policy is trained to navigate from any start region (open circles) to any goal region (filled circles). (b) At deployment, the BDI agent decomposes a multi-hop mission by sequencing navigation sub-goals (grey intermediate nodes), invoking the same trained policy for each segment without retraining.

of invalid actions grows, and that it leaves the convergence properties of the underlying optimizer intact. The novelty in RAVEN is not the mechanism but its source: the mask is derived from RCC-8 facts in the belief base rather than from hard-coded simulator rules, so the constraint specification is human-readable, formally grounded in a well-established spatial calculus, and modifiable without touching the sub-symbolic component.

Reward machine. The reward signal is generated by a compact, declarative reward machine. We use the term “reward machine” in the broad sense of a structured, declarative specification of the reward signal. The reward machine in our setting consists of three rules over agent beliefs, rather than a full finite-state automaton over propositional symbols as in Toro Icarte et al [60, 61]. We use this term because the rules are evaluated in the symbolic layer and produce the scalar reward through symbolic inference, which is the property we rely on for inspectability and modifiability. The automaton structure of full reward machines would become relevant if reward conditions depended on the agent’s history of visited regions – as is the case in some VEsNA tasks beyond shortest-path navigation – making this a natural extension for future work. Each reward component is encoded as a Prolog rule in the agent’s belief base (as Jason beliefs may also be defined via rules, and not just as facts), whose antecedent is a logical query over the agent’s beliefs and whose consequent is a scalar value forwarded to the sub-symbolic layer. Consequently, the scalar received by the policy is the result of symbolic inference rather than the output of a numeric function encapsulated within the environment simulator.

The navigation task is governed by three reward-specification rules. A per-step rule assigns a reward of $R_{\text{step}} = -1$ at every state transition, thereby incentivizing efficiency in trajectory length. A terminal rule assigns a reward of $R_{\text{goal}} = +100$ whenever the predicate $c_t = g$ is derivable within the reasoning cycle, i.e., whenever the beliefs corresponding to the current region and the goal region unify. In addition, a timeout rule assigns $R_{\text{timeout}} = -10$ when the per-episode step counter reaches its predefined budget. The sub-symbolic layer observes only the resulting scalar reward at each transition; the conditions under which this reward is generated, as well as the rules that produce it, remain confined to the symbolic component of the architecture. Because each reward component is implemented as a logical rule over the agent’s beliefs rather than as a numerical function over state variables, the resulting reward function is readily inspectable, amenable to formal verification, and can be modified without retraining the neural component. Introducing a new reward condition—for example, assigning an additional reward for entering a corridor adjacent to a designated region of interest—requires editing a single rule in the belief base, rather than modifying the simulator code or the network parameters. This is the intended operational meaning of the term “declarative reward machine” in this paper, and it is precisely this characteristic that enables the symbolic layer to retain normative authority over the objective that the policy is optimizing.

Table 2

Training configuration of the sub-symbolic learning layer, per environment scale. The top block reports hyperparameters that are held fixed across every run. The bottom block reports capacity parameters that are scaled with the size of the graph so that the policy has enough representation capacity and exploration horizon for the larger action spaces. Values marked N are the number of regions of the graph, equal to $|V|$.

Hyperparameter	Small ($N=11$)	Medium ($N=50$)	Large ($N=103$)
<i>Fixed across scales</i>			
Optimiser	Adam	Adam	Adam
Learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}
Discount factor γ	0.99	0.99	0.99
Activation	ReLU	ReLU	ReLU
Loss	Huber (SmoothL1)	Huber (SmoothL1)	Huber (SmoothL1)
Gradient clip (ℓ_2 norm)	10	10	10
Hidden layers (depth)	2	2	2
Exploration	ε -greedy over $\mathcal{A}(s_t)$	ε -greedy over $\mathcal{A}(s_t)$	ε -greedy over $\mathcal{A}(s_t)$
$\varepsilon_0 / \varepsilon_{\min}$	1.0 / 0.01	1.0 / 0.01	1.0 / 0.01
State / action dimension	$2N / N$	$2N / N$	$2N / N$
<i>Scaled with environment size</i>			
Hidden width	64	128	128
Replay buffer capacity	10,000	100,000	50,000
Mini-batch size	64	128	128
Target update (episodes)	10	200	500
ε decay per episode	0.9999	0.99997	0.999985
<i>Hardware platform</i>			
GPU	NVIDIA GeForce RTX 3060 (Ampere, 12 GB VRAM, CUDA 12.5)		
CPU	Intel Core i9-11900KF @ 3.50 GHz (8 cores, 16 threads, 5.3 GHz boost)		
System RAM	128 GB		
Operating system	Ubuntu 24.04.4 LTS (kernel 6.8.0)		

3.3. Sub-symbolic layer: a masked DQN policy server

The sub-symbolic layer is a Deep Q-Network (DQN) [19] with a target network, experience replay, and action masking, deployed as a REST policy server. The Q-network is a multilayer perceptron with two hidden layers and ReLU activations that maps the $2N$ -dimensional state vector to N action values; a target network of identical architecture computes the bootstrap targets and is refreshed by a hard parameter copy at a fixed interval. Experience tuples are stored in a circular replay buffer and sampled uniformly at random during training. The hidden layer width, mini-batch size, target-network update period, and ε -decay constant are scaled when transitioning from the small to the medium graph, and subsequently held fixed when transitioning from the medium to the large graph. Replay-buffer capacity is reduced for the large graph due to GPU memory limitations (12 GB VRAM). The complete configuration at each scale is reported in Table 2.

The only deviation from the standard DQN recipe concerns the bootstrap target. Given a transition $(s_t, a_t, r_{t+1}, s_{t+1}, d_t)$ retrieved from the buffer, together with the admissible action set $\mathcal{A}(s_{t+1})$ recorded at the time of the transition, the training target is

$$y_t = r_{t+1} + \gamma (1 - d_t) \max_{a' \in \mathcal{A}(s_{t+1})} Q_{\theta^-}(s_{t+1}, a'), \quad (3)$$

where $d_t \in \{0, 1\}$ is the termination flag. Restricting the maximization to the admissible set prevents bootstrap estimates from being poisoned by Q-values of spatially forbidden transitions, a form of consistency that the standard DQN formulation does not enforce. Training uses the Huber loss between $Q_{\theta}(s, a)$ and y , optimized with Adam and gradient-clipped to an ℓ_2 norm of 10. Exploration is ε -greedy over the admissible set, with a geometric decay schedule from 1.0 to a floor of 0.01.

The boundary between the symbolic and sub-symbolic layers in RAVEN is specified as an explicit design contract rather than arising as an incidental property of the implementation. The DQN is

implemented as an encapsulated Python service whose public interface accepts only numeric tensors and returns solely an integer action identifier. It is never directly exposed to region labels, goal symbols, plan identifiers, the RCC-8 relations that underlie the action mask, or the rules governing reward computation. From the perspective of the network, the input is treated as an anonymous binary vector, the action mask as a list of integer indices, the output as a single index, and the reward as a post-hoc scalar. The Encoder transforms the symbolic state into this numeric representation prior to learning, and the Decoder maps the returned index back into a symbolic action.

Interpretability is a structural property of the architecture rather than an ad-hoc augmentation. Because no symbolic content traverses the neural component, every symbolic justification the agent can produce about a decision is preserved by construction: why a state was encoded in the way it was, why a particular neighbour qualified as admissible, why the chosen index was decoded into one symbolic action rather than another, and why a given reward was issued. The symbolic layer retains semantic authority at both ends of the pipeline while the sub-symbolic layer performs optimization in between, and the explainability of the agent’s behaviour follows directly from this separation of concerns. The formal description of the interface as a composition of three deterministic mappings is given in Section 3.4.

3.4. Decoupling reasoning from learning: the encoder-decoder interface

The interface between the symbolic and sub-symbolic layers is the composition of three deterministic mappings. Let $c_t \in \mathcal{R}$ denote the current region and $g \in \mathcal{R}$ the goal region maintained in the Jason belief base, where $\mathcal{R}$ is the finite set of symbolic region identifiers. The Encoder

$$\mathcal{E} : (c_t, g) \mapsto s_t \in \{0, 1\}^{2N}$$

projects the symbolic state into the numeric observation defined in Eq. (1), and the spatial-reasoning component of the symbolic layer derives the admissible action set $\mathcal{A}(s_t) \subseteq \{0, \dots, N - 1\}$ from the RCC-8 belief atoms. At every decision step the symbolic layer issues a single JSON request to the REST server, carrying the tuple

$$\langle \text{id}_{\text{agent}}, s_t, \mathcal{A}(s_t), r_{t-1}, d_{t-1} \rangle,$$

where $r_{t-1} \in \mathbb{R}$ and $d_{t-1} \in \{0, 1\}$ are the scalar reward and termination flag emitted by the reward machine for the *previous* transition. The server replies with a single integer index

$$a_t = \pi_\theta(s_t, \mathcal{A}(s_t)) \in \mathcal{A}(s_t),$$

which the Decoder $\mathcal{D} : a_t \mapsto r' \in \mathcal{R}$ maps back to a symbolic action that the Jason agent executes as an atomic graph transition. Training and inference share the same endpoint and are distinguished only by a mode flag that disables exploration and weight updates at evaluation time.

The payload is intentionally minimal: it carries no region names, no goal literal, no plan identifier, and no textual content, and it does not expose the reward machine or the spatial calculus that produced $\mathcal{A}(s_t)$. All symbolic content remains within the symbolic layer; the sub-symbolic layer observes only the numeric projection $\mathcal{E}(c_t, g)$ together with the integer mask. The interface is therefore the composition

$$(c_t, g) \xrightarrow{\mathcal{E}} s_t \xrightarrow{\pi_\theta} a_t \xrightarrow{\mathcal{D}} r',$$

in which symbolic information enters and leaves through $\mathcal{E}$ and $\mathcal{D}$ but never traverses π_θ itself. This one-way restriction allows the symbolic layer to alter environments, goals, or reward rules by editing the belief base without retraining π_θ , and conversely permits π_θ to be replaced by an alternative learning algorithm (for instance, a Double DQN [20] or a Dueling DQN [58]) without disturbing the agent’s declarative programming.

Table 3

Navigation environments and per-scale training statistics. The first block reports structural properties of the office connectivity graph $G = (V, E)$; the second block reports training cost.

	$N=11$	$N=50$	$N=103$
<i>Environment</i>			
Regions $ V $	11	50	103
Edges $ E $	14	63	108
Start-goal pairs $N(N-1)$	110	2,450	10,506
<i>Training</i>			
Episodes N_{ep} (95% threshold)	≈ 300	6,621	16,885
Total episodes N_{total}	30,000	30,000	30,000
Wall-clock	33 min	1 h 16 min	3 h 26 min

4. Experiments and Results

We evaluate RAVEN on three navigation tasks of progressively increasing scale that share an identical underlying architecture. This section details the navigation environments (Section 4.1); the network configuration and training protocol (Section 4.2–Section 4.3); the deterministic evaluation procedure (Section 4.4); presents the empirical results, structured around four central research questions (Section 4.5); and concludes with a discussion of the method’s limitations (Section 4.6).

4.1. Navigation environments

The three graphs are derived from the office environment distributed with VEsNA. The **small** graph (11 regions, 14 edges) corresponds to the original VEsNA core layout (reception area, corridors, offices, and shared spaces). The **medium** graph (50 regions, 63 edges) augments this layout with additional laboratories, meeting rooms, and service areas. The **large** graph (103 regions, 108 edges) represents the complete extended building configuration.

All three layouts are encoded as RCC-8 spatial relations within the belief base and are dynamically instantiated during system execution. the sub-symbolic layer remains unchanged when the underlying graph is replaced. As N increases, the admissible action set $\mathcal{A}(s)$ occupies an increasingly smaller proportion of the overall action space, i.e., precisely the regime in which action masking provides the greatest benefit [36]. A concise overview of the three environments is presented in Table 3.

4.2. Network architecture and hyperparameters

For each environment, we train a single RAVEN agent from scratch using the scale-specific configuration given in Table 2. The overall architectural template is kept identical across the three runs; only the capacity-related hyperparameters (hidden layer width, replay-buffer size, mini-batch size, target-network update interval, and the ε -decay constant) are adjusted according to the graph size. In contrast, the learning rate, discount factor, activation function, loss, and optimiser remain unchanged. The Q-network’s input and output dimensions grow as $2N$ and N , respectively. We scale the capacity parameters because both the action space and the combined state–goal space expand with N , implying that the policy must have higher representational capacity and a longer effective exploration horizon on larger graphs in order to reach convergence within a similar number of episodes.

4.3. Training configuration

At the beginning of each episode, the symbolic layer draws a pair (c_0, g) uniformly at random from $V \times V$ with $c_0 \neq g$, so the policy is exposed to the complete joint distribution of navigation tasks, without any curriculum based on distance or difficulty. The step budget per episode is chosen as a

multiple of the expected shortest-path length for each graph: 100 steps for the small environment, 150 for the medium, and 300 for the large. An episode ends either when the agent arrives at g and the reward machine produces $R_{\text{goal}} = +100$, or when the step budget is depleted and the reward machine outputs $R_{\text{timeout}} = -10$. In both termination cases, the sub-symbolic layer receives the scalar reward along with the termination flag on the subsequent call to the policy server. Training is carried out for 30,000 episodes on each graph with no early stopping, allowing the complete learning curve to be observed.

4.4. Evaluation procedure

After training is complete, we assess the policy in a purely inference-based setting with $\varepsilon = 0$, fixed parameters, and an empty replay buffer; thus, the reported metrics capture only the behavior of the learned Q-function, excluding any residual exploration. For each environment, we perform an *exhaustive* deterministic rollout over all $N(N - 1)$ ordered start-goal pairs, using the same RCC-8 action mask and reward machine as in training.

BFS as the optimality oracle. On an unweighted graph, Breadth-First Search (BFS) computes the minimum number of edges separating any two vertices, i.e., the theoretical lower bound on path length that no algorithm can improve upon. We use BFS as the evaluation oracle for two reasons. First, every region transition costs one step, so BFS distance equals the true shortest path on the unweighted navigation graph. Second, the RCC-8 adjacency facts that BFS operates on are *the same facts* that the symbolic layer uses to construct the action mask during training, ensuring that any gap between the agent and the oracle is attributable to the learned policy, not to a mismatch in the underlying connectivity. We define the BFS optimality ratio

$$\bar{\rho} = \frac{\bar{L}_{\text{BFS}}}{\bar{L}_{\text{agent}}},$$

where $\bar{L}_{\text{BFS}}$ and $\bar{L}_{\text{agent}}$ are the mean path lengths of the BFS oracle and the RAVEN policy, respectively, taken over all $N(N - 1)$ start-goal pairs. A ratio of 1.000 indicates that the policy recovers the shortest path on every pair.

The evaluation is organized around four questions. **Q1 (path optimality):** does the learned policy recover shortest paths? **Q2 (generalisation):** does the region-agnostic encoding transfer to start-goal pairs unlikely to have been visited during training? **Q3 (scalability):** does the architecture continue to converge as N and $|A|$ grow, with no change to the sub-symbolic layer or the interface? **Q4 (role of the symbolic mask):** how much of the optimality result is attributable to the mask, as opposed to the learning algorithm itself?

4.5. Results

Q1: Path optimality. Across all three graph sizes, the learned policy reaches the goal for every ordered start-goal pair and exactly matches the BFS-optimal paths: we $\bar{\rho} = 1.000$ on the small, medium, and large graphs (Figure 5). We attribute this outcome to two key structural aspects of the masked Q-learning recursion: first, the mask eliminates inadmissible actions before the argmax is taken, ensuring the policy never deviates along detours; second, the bootstrap target in Eq. (3) is computed only over admissible successors, preventing Q-values of disallowed moves from contaminating neighbouring estimates. The Q4 ablation shows that this behaviour breaks down once the mask is removed. For clarity, on the small ($N=11$, 110 pairs) and medium ($N=50$, 2,450 pairs) graphs, 30,000 episodes are enough for each pair to be sampled multiple times, so the optimality result at these scales mainly indicates that the policy can memorize all pairs under the masked recursion. The claim of generalization is instead supported by Q2 on the large graph, where we expect roughly 5% of pairs never to be encountered during training.

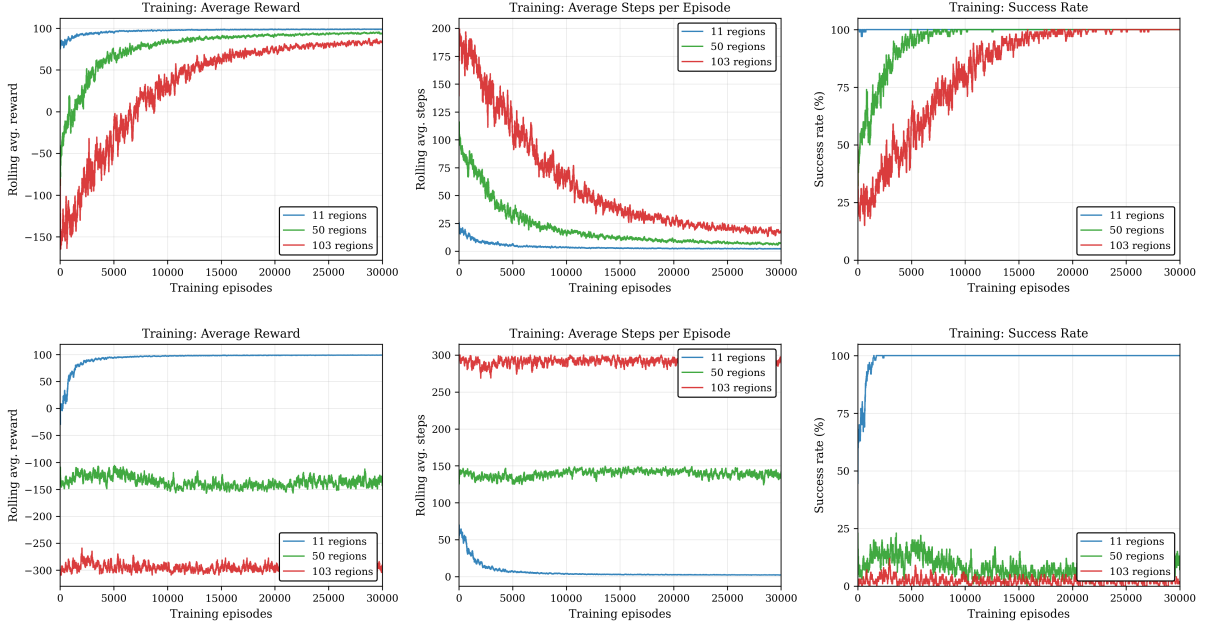


Figure 4: Training-time convergence across the three office-graph scales (11, 50, 103 regions). All curves are 100-episode rolling statistics recorded *during* training under ε -greedy exploration; deployment-time metrics ($\varepsilon=0$, frozen weights) are reported in Table 3. **Top row (with RCC-8 mask):** from left to right, rolling average reward, rolling average steps per episode, and rolling success rate. All three scales converge: reward rises toward $R_{\text{goal}} - \bar{L}_{\text{BFS}}$, steps fall toward the BFS-optimal path length, and success rate reaches 100%. The lower reward asymptote on larger graphs reflects longer mean paths and residual ε -greedy noise ($\varepsilon \approx 0.41$ for $N=50$ and 0.64 for $N=103$ at episode 30,000), not a degradation of the learned policy. **Bottom row (mask removed):** the same three metrics under unmasked training. $N=11$ still converges, as the graph is small enough that random exploration suffices, but $N=50$ and $N=103$ show no convergence across all three metrics over 30,000 episodes, confirming that the RCC-8 symbolic constraint is essential for sample efficiency at scale.

Q2: Within-graph generalisation across start-goal pairs. On a fixed graph, the goal-conditioned encoding converts the policy into a single mapping that must solve the navigation problem for every (c, g) pair, not just the ones encountered during training. Conducting an exhaustive evaluation on the large graph provides the clearest picture of this within-graph generalization capability. The joint state-goal space contains 10,506 pairs against 30,000 training episodes. Under uniform random sampling, ≈ 604 pairs (5.75%) are expected never to have been sampled. The trained policy reaches the goal optimally on every evaluation pair, including the expected-unseen subset, supporting the within-graph generalization claim. The near-optimal performance over the entire evaluation set, including those pairs separated by the graph diameter, supports the conclusion that a goal-conditioned one-hot encoding generalizes over the entire task distribution without requiring relabeling, curriculum design, or hindsight replay. Structurally, this holds because the goal component of the state is generated by the BDI agent from its own belief base: the encoding for a previously unseen pair at evaluation time is drawn from the same distribution as the encoding encountered during training.

Q3: Scalability across environment sizes. Figure 4 (top row) confirms convergence at all three scales with no change to the sub-symbolic architecture or interface; only the capacity parameters of Table 2 are scaled with $|\mathcal{A}|$. The empirical convergence episode grows from ≈ 300 to 6,621 to 16,885 as the joint state-goal space grows from 110 to 2,450 to 10,506 pairs, a sub-linear increase relative to $N(N-1)$. Wall-clock costs are 33 min, 1 h 16 min, and 3 h 26 min respectively (Table 3). The sub-linear growth is precisely what masking provides: as N increases, the fraction of inadmissible actions at each step grows accordingly, and the mask keeps the effective action set seen by the policy locally bounded regardless of the total graph size. Although Figure 3 reports residual $\varepsilon \approx 0.41$ for $N = 50$ and $\varepsilon \approx 0.64$ for $N = 103$ at episode 30,000, the rolling success rate is already stable above 95% from episodes 3,924

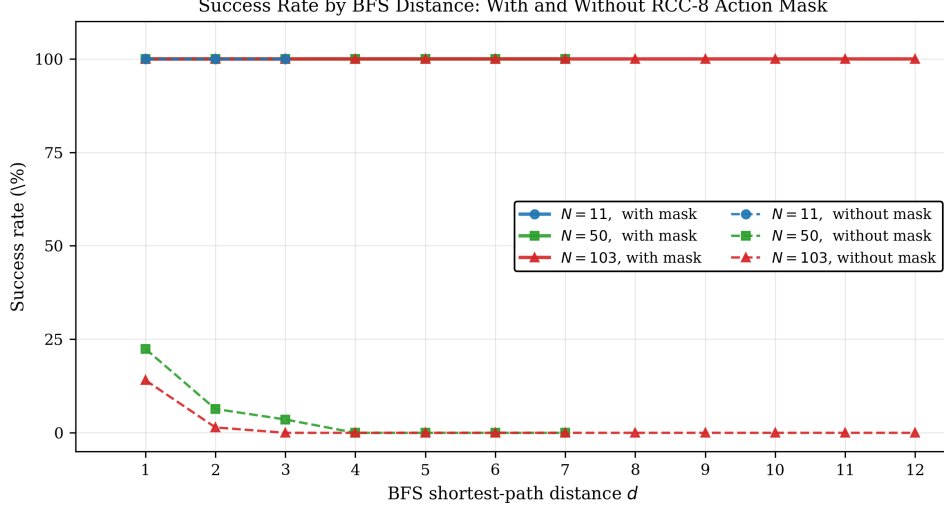


Figure 5: Success rate by BFS shortest-path distance d for all six trained policies: three graph scales ($N=11$, 50, 103, shown in blue, green, red respectively) each evaluated with the RCC-8 action mask (solid lines) and without it (dashed lines). The x-axis position k corresponds to BFS distance $d = k + 1$; the first tick ($k=0$) represents $d=1$. The masked policies maintain 100% success at every distance for every scale. The no-mask policies for $N=11$ also achieve 100% (the graph is small enough for unguided exploration to succeed), but the $N=50$ and $N=103$ no-mask policies collapse rapidly: at $d=1$ success is 22.4% and 14.1% respectively, and both reach 0% by $d=3$ (or $d=4$ for $N=50$), quantifying the structural contribution of the symbolic constraint across all evaluation scales.

and 11,527 respectively, well before the end of training; thus, the evaluation under $\varepsilon = 0$ in Table 3 reflects a converged policy rather than an undertrained one.

Q4: Role of the symbolic mask. Figure 5 shows that policies *trained* without the RCC-8 mask fail almost entirely at scale (3.02% for $N=50$, 0.42% for $N=103$). To separate the mask’s training-time benefit from its inference-time effect, we additionally evaluate the masked-trained 103-region policy under two inference regimes: *masked* (argmax over $\mathcal{A}(s_t)$) and *unmasked* (argmax over all $N=103$ actions, with the agent remaining in place when an inadmissible action is selected and the step counter still advancing). Removing the mask at inference collapses success from 100% to 14.0% (1,467/10,506) and the mean per-pair return from +95.45 to -253.12. The per-distance breakdown clarifies the mechanism. At $d=1$ the unmasked policy reaches the goal in 79.1% of cases, confirming that the unmasked Q -function does rank admissible moves above inadmissible ones on the first hop. Success then drops to 53.1% at $d=2$, 28.2% at $d=3$, 7.0% at $d=4$, and effectively 0% for $d \geq 9$. The decay is steeper than independent per-step compounding (0.791^d would predict 62.6%, 49.5%, 39.1%): per-step accuracy itself degrades with depth, since longer trajectories visit states whose Q -estimates are further from the terminal reward and therefore noisier.

Successful unmasked rollouts coincide exactly with BFS trajectories, since any chain of admissible picks is, by construction, optimal. The 86-percentage-point gap between masked and unmasked inference quantifies the structural contribution of the symbolic layer. With an externally supplied adjacency constraint and the masked bootstrap target of Eq. (3), Q -learning on a finite bounded-degree graph is expected to converge to BFS-optimal behaviour. RAVEN’s contribution is therefore not that learning is unusually effective on this task, but that the entire MDP specification, comprising state encoding, admissible action set, and reward signal, is supplied from the BDI belief base without modifying the Jason reasoning cycle.

Q5: Comparison with hand-crafted VEsNA plans. In [62], we implemented a concert scenario with 10 regions and we solved navigation through hand-crafted AgentSpeak(L) plans: for each pair of adjacent regions, a plan body specifies how the agent moves between them, and longer trajectories

are obtained by plan chaining. Compared with this baseline, RAVEN replaces explicit per-segment plans with a single learned policy and shifts the engineering effort from authoring plans to specifying the symbolic constraints that structure learning (the RCC-8 adjacency relations, the reward machine, the goal-conditioned state encoding). The trade-off is concrete. Hand-crafted plans are deterministic, immediately deployable without any training cost, and provide line-by-line interpretability of every transition; the limitation is brittleness, in that any extension to the office requires editing the plan library. RAVEN requires a one-time training cost per graph (Table 3), but once trained, no edits to the plan library are needed when goals change, and the symbolic constraints that shape behavior are factored out into declarative rules that can be modified without retraining. Within the scope of this paper, the empirical advantage of RAVEN over hand-crafted plans is not in path optimality but in the locus of engineering effort and in the path it opens toward stochastic and dynamic environments where deterministic plans cannot anticipate every situation.

4.6. Limitations

The main limitations of our work include:

(i) **Generalization and Graph-Specificity:** The learned policy is specific to the training environment. Because state and action dimensions scale with N and one-hot identifiers are graph-specific, a checkpoint cannot be transferred to a new environment without retraining. We demonstrate within-graph generalization—where a single policy handles all start-goal pairs within a fixed topology—rather than topological transfer, which would require permutation-invariant architectures like Graph Neural Networks.

(ii) **Environment Complexity:** Our experiments were conducted on abstract navigation graphs, validating shortest-path reasoning over discrete transitions. The current evaluation does not account for the robustness of the architecture under realistic execution conditions. In the full VEsNA/Godot pipeline, the symbolic layer (BDI/Jason) will function as an abstraction layer; future work will evaluate the interface’s behavior when handling perception errors, physics-based noise, and low-level control, where the BDI agent must re-trigger the sub-symbolic policy upon failures.

5. Conclusion

We presented RAVEN, a neuro-symbolic extension of the VEsNA toolkit that embeds reinforcement learning into BDI agents while maintaining symbolic interpretability. RAVEN is explicitly structured to separate concerns: the Jason-based symbolic layer handles goal specification, spatial constraints (using RCC-8), and reward definition, whereas a DQN policy server is responsible for sub-symbolic optimisation. Since no symbolic information passes through the neural interface, the agent’s symbolic explanations for its decisions remain complete and readable to humans. Our experiments indicate that this architecture scales efficiently across all goals in the graph.

Section 4.5 Q5 offers a qualitative comparison of RAVEN with the original, hand-engineered VEsNA plans, illustrating how RAVEN moves engineering effort away from crafting individual scenario-specific plans and toward writing declarative constraints that guide learning. Several avenues for future work remain. First, establishing a formal quantitative benchmark that contrasts both paradigms in dynamic, stochastic environments. Second, validating the approach within the full VEsNA/Godot pipeline, where the symbolic layer would also represent perceptual beliefs originating from the 3D engine, augment RCC-8 facts with metric tolerance zones to account for localisation noise, and convert region transitions into waypoints for the low-level controller, while the sub-symbolic component would still receive only numeric tensors and output only integer action identifiers. Additional penalties, such as for collisions or energy use, can be encoded as further Jason rules without any retraining, thereby preserving the separation of concerns that supports interpretability.

In addition, RAVEN serves as the trainable component of the broader framework introduced in our paper on the secret life of traces [33]. That framework supports the learning, explanation, and verification of each agent decision by examining the preceding sequences of states within the belief

and plan bases. The current paper focuses on demonstrating RAVEN in the navigation setting, while future work will apply it to other decision-making problems within the same framework and finalise its integration with the explanation and verification modules. A full implementation and evaluation of RAVEN are available at <https://github.com/ZahraD-a/Vesna-RL>.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: improve grammar, refine phrasing, and check the consistency of notation. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] A. S. Rao, M. P. Georgeff, BDI agents: From theory to practice, in: V. R. Lesser, L. Gasser (Eds.), *Proceedings of the First International Conference on Multiagent Systems*, June 12-14, 1995, San Francisco, California, USA, The MIT Press, 1995, pp. 312–319. URL: <https://api.semanticscholar.org/CorpusID:269838374>.
- [2] R. Bordini, J. Hübner, M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak Using Jason*, volume 8, 2007. doi:10.1002/9780470061848.
- [3] A. S. Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, in: 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, January 22-25, 1996, volume 1038 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 42–55. URL: <https://doi.org/10.1007/BFb0031845>. doi:10.1007/BFb0031845.
- [4] M. Winikoff, Z. Daoui, A. Gatti, V. Mascardi, A. Ferrando, AgentSpeaX: Explain, actually, 2026. Accepted for inclusion in the EMAS 2026 Post Proceedings.
- [5] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, *Artif. Intell.* 267 (2019) 1–38. URL: <https://doi.org/10.1016/j.artint.2018.07.007>. doi:10.1016/J.ARTINT.2018.07.007.
- [6] M. Harbers, K. van den Bosch, J. C. Meyer, Design and evaluation of explainable BDI agents, in: J. X. Huang, A. A. Ghorbani, M. Hacid, T. Yamaguchi (Eds.), *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2010*, Toronto, Canada, August 31 - September 3, 2010, IEEE Computer Society Press, 2010, pp. 125–132. URL: <https://doi.org/10.1109/WI-IAT.2010.115>. doi:10.1109/WI-IAT.2010.115.
- [7] J. Broekens, M. Harbers, K. V. Hindriks, K. van den Bosch, C. M. Jonker, J. C. Meyer, Do you get it? User-evaluated explainable BDI agents, in: J. Dix, C. Witteveen (Eds.), *Multiagent System Technologies, 8th German Conference, MATES 2010*, Leipzig, Germany, September 27-29, 2010. *Proceedings*, volume 6251 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 28–39. URL: https://doi.org/10.1007/978-3-642-16178-0_5. doi:10.1007/978-3-642-16178-0_5.
- [8] M. Winikoff, G. Sidorenko, V. Dignum, F. Dignum, Why bad coffee? Explaining BDI agent behaviour with valuings, *Artif. Intell.* 300 (2021) 103554. URL: <https://doi.org/10.1016/j.artint.2021.103554>. doi:10.1016/J.ARTINT.2021.103554.
- [9] L. A. Dennis, N. Oren, Explaining BDI agent behaviour through dialogue, *Auton. Agents Multi Agent Syst.* 36 (2022) 29. URL: <https://doi.org/10.1007/s10458-022-09556-8>. doi:10.1007/S10458-022-09556-8.
- [10] A. R. Panisson, D. C. Engelmann, R. H. Bordini, Engineering Explainable Agents: An argumentation-based approach, in: N. Alechina, M. Baldoni, B. Logan (Eds.), *Engineering Multi-Agent Systems - 9th International Workshop, EMAS 2021*, Virtual Event, May 3-4, 2021, Revised Selected Papers, volume 13190 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 273–291. URL: https://doi.org/10.1007/978-3-030-97457-2_16. doi:10.1007/978-3-030-97457-2_16.
- [11] Z. Buçinca, S. Swaroop, A. E. Paluch, F. Doshi-Velez, K. Z. Gajos, Contrastive explanations that anticipate human misconceptions can improve human decision-making skills, in: N. Yamashita,

- V. Evers, K. Yatani, S. X. Ding, B. Lee, M. Chetty, P. O. T. Dugas (Eds.), *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI 2025, Yokohama Japan, 26 April 2025- 1 May 2025*, ACM, 2025, pp. 1024:1–1024:25. URL: <https://doi.org/10.1145/3706598.3713229>. doi:10.1145/3706598.3713229.
- [12] M. Winikoff, Contrastive explanations of BDI agents, *CoRR* abs/2602.13323 (2026). URL: <https://doi.org/10.48550/arXiv.2602.13323>. doi:10.48550/ARXIV.2602.13323. arXiv:2602.13323.
- [13] H. M. R. Jasinski, M. M. Espinoza, C. A. Tacla, Towards generating p-contrastive explanations for goal selection in Extended-BDI agents, in: M. C. Naldi, R. A. C. Bianchi (Eds.), *Intelligent Systems - 12th Brazilian Conference, BRACIS 2023, Belo Horizonte, Brazil, September 25-29, 2023, Proceedings, Part I, Lecture Notes in Computer Science*, Springer, 2023, pp. 351–366. URL: https://doi.org/10.1007/978-3-031-45368-7_23. doi:10.1007/978-3-031-45368-7_23.
- [14] S. Rodriguez, J. Thangarajah, A. Davey, Design patterns for explainable agents (xag), in: M. Dastani, J. S. Sichman, N. Alechina, V. Dignum (Eds.), *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024, International Foundation for Autonomous Agents and Multiagent Systems / ACM*, 2024, pp. 1621–1629. URL: <https://dl.acm.org/doi/10.5555/3635637.3663023>. doi:10.5555/3635637.3663023.
- [15] M. Mauri, M. Minor, Towards Explainable BDI Agents for End Users, in: S. Rodriguez, L. Feng, J. P. Müller (Eds.), *Engineering Multi-Agent Systems - 13th International Workshop, EMAS 2025, Detroit, MI, USA, May 19-20, 2025, Revised Selected Papers, Lecture Notes in Computer Science*, Springer, 2025, pp. 221–237. URL: https://doi.org/10.1007/978-3-032-18011-7_13. doi:10.1007/978-3-032-18011-7_13.
- [16] E. Yan, S. Burattini, J. F. Hübner, A. Ricci, A multi-level explainability framework for engineering and understanding BDI agents, *Auton. Agents Multi Agent Syst.* 39 (2025) 9. URL: <https://doi.org/10.1007/s10458-025-09689-6>. doi:10.1007/s10458-025-09689-6.
- [17] G. Ciatto, G. Aguzzi, R. Battistini, M. Baiardi, S. Burattini, A. Ricci, Exploiting GenAI for Plan Generation in BDI Agents, in: I. Lynce, N. Murano, M. Vallati, S. Villata, F. Chesani, M. Milano, A. Omicini, M. Dastani (Eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025), Frontiers in Artificial Intelligence and Applications*, IOS Press, 2025, pp. 3495–3502. URL: <https://doi.org/10.3233/FAIA251223>. doi:10.3233/FAIA251223.
- [18] A. Y. Ichida, F. Meneguzzi, R. C. Cardoso, BDI agents in natural language environments, in: M. Dastani, J. S. Sichman, N. Alechina, V. Dignum (Eds.), *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024, International Foundation for Autonomous Agents and Multiagent Systems / ACM*, 2024, pp. 880–888. URL: <https://dl.acm.org/doi/10.5555/3635637.3662942>. doi:10.5555/3635637.3662942.
- [19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, *Nat.* 518 (2015) 529–533. URL: <https://doi.org/10.1038/nature14236>. doi:10.1038/NATURE14236.
- [20] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: D. Schuurmans, M. P. Wellman (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, AAAI Press*, 2016, pp. 2094–2100. URL: <https://doi.org/10.1609/aaai.v30i1.10295>. doi:10.1609/AAAI.V30I1.10295.
- [21] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, *Nat.* 529 (2016) 484–489. URL: <https://doi.org/10.1038/nature16961>. doi:10.1038/NATURE16961.
- [22] S. Amarjyoti, Deep reinforcement learning for robotic manipulation - the state of the art, *CoRR* abs/1701.08878 (2017).

- [23] A. Gatti, V. Mascardi, Vesna, a framework for virtual environments via natural language agents and its application to factory automation, *Robotics* 12 (2023) 46. URL: <https://doi.org/10.3390/robotics12020046>. doi:10.3390/ROBOTICS12020046.
- [24] A. Ferrando, A. Gatti, V. Mascardi, Integrating Virtual Reality, Chatbots, and BDI Agents: VEsNA Goes Fast!, Springer Nature Switzerland, Cham, 2026, pp. 289–322. URL: https://doi.org/10.1007/978-3-032-01082-7_11. doi:10.1007/978-3-032-01082-7_11.
- [25] A. Gatti, A. Ferrando, V. Mascardi, VEsNA-Toolkit web site, 2026. URL: <https://github.com/VEsNA-Toolkit>, accessed on June 10, 2026.
- [26] A. Gatti, V. Mascardi, A. Ferrando, ChatBDI: Think BDI, talk LLM, in: S. Das, A. Nowé, Y. Vorobeychik (Eds.), *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025, International Foundation for Autonomous Agents and Multiagent Systems / ACM*, 2025, pp. 2541–2543. URL: <https://dl.acm.org/doi/10.5555/3709347.3743930>. doi:10.5555/3709347.3743930.
- [27] A. Gatti, V. Mascardi, A. Ferrando, Let me talk to you! natural language interaction between humans and BDI agents via ChatBDI, in: I. Lynce, N. Murano, M. Vallati, S. Villata, F. Chesani, M. Milano, A. Omicini, M. Dastani (Eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025), Frontiers in Artificial Intelligence and Applications*, IOS Press, 2025, pp. 3646–3654. URL: <https://doi.org/10.3233/FAIA251242>. doi:10.3233/FAIA251242.
- [28] Godot foundation, Godot web site, 2025. URL: <https://godotengine.org/>, accessed on June 10, 2026.
- [29] Unity technologies, Unity web site, 2026. URL: <https://unity.com/>, accessed on June 10, 2026.
- [30] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with JaCaMo, *Sci. Comput. Program.* 78 (2013) 747–761. URL: <https://doi.org/10.1016/j.scico.2011.10.004>. doi:10.1016/J.SCICO.2011.10.004.
- [31] D. A. Randell, Z. Cui, A. G. Cohn, A spatial logic based on regions and connection, in: B. Nebel, C. Rich, W. R. Swartout (Eds.), *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. Cambridge, MA, USA, October 25-29, 1992, Morgan Kaufmann, 1992, pp. 165–176.
- [32] A. Ferrando, A. Gatti, V. Mascardi, Geometric and spatial reasoning in BDI agents: A survey, in: E. D. Angelis, M. Proietti (Eds.), *Proceedings of the 39th Italian Conference on Computational Logic, Rome, Italy, June 26-28, 2024, CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: <https://ceur-ws.org/Vol-3733/paper1.pdf>.
- [33] D. Ancona, Z. Daoui, A. Ferrando, A. Gatti, V. Mascardi, M. Winikoff, The secret life of traces: A MAS engineering perspective, 2026. Accepted for inclusion in the EMAS 2026 Post Proceedings.
- [34] L. P. Kaelbling, Learning to achieve goals, in: R. Bajcsy (Ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Chambéry, France, August 28 - September 3, 1993, Morgan Kaufmann, 1993, pp. 1094–1099.
- [35] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, Hindsight experience replay, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017*, pp. 5048–5058. URL: <https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html>.
- [36] S. Huang, S. Ontañón, A closer look at invalid action masking in policy gradient algorithms, in: R. Barták, F. Keshtkar, M. Franklin (Eds.), *Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2022, Hutchinson Island, Jensen Beach, Florida, USA, May 15-18, 2022, Florida Online Journals*, 2022. URL: <https://doi.org/10.32473/flairs.v35i.130584>. doi:10.32473/FLAIRS.V35I.130584.
- [37] C. Tang, C. Liu, W. Chen, S. D. You, Implementing action mask in proximal policy optimization (PPO) algorithm, *ICT Express* 6 (2020) 200–203. URL: <https://doi.org/10.1016/j.ict.2020.05.003>. doi:10.1016/J.ICTE.2020.05.003.
- [38] A. Gatti, F. Casale, V. Mascardi, A. Stucchi, A. Ferrando, Vesna-Pro: Exploiting BDI agents

- with propensities for emergent narrative, in: Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '26, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2026, p. 967–976. URL: <https://doi.org/10.65109/PDPF4024>. doi:10.65109/PDPF4024.
- [39] Integrating JaCa-Unity in VEsNA for behavioural realistic VR simulations, 2026. To appear in the Proceedings of the 22nd European Conference on Multi-Agent Systems, EUMAS 2025, Bucharest, Romania, September 3–5, 2025, Part I.
 - [40] A. Gatti, A. Ferrando, V. Mascardi, Situated agents in action: Extending VEsNa with spatial and grabbable artifacts, in: P. Mathieu, F. D. la Prieta (Eds.), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Computational Social Science: The PAAMS Collection - 23rd International Conference, PAAMS 2025, Lille, France, June 25–27, 2025, Proceedings, Lecture Notes in Computer Science*, Springer, 2025, pp. 348–353. URL: https://doi.org/10.1007/978-3-032-07638-0_29. doi:10.1007/978-3-032-07638-0_29.
 - [41] J. Nazzaro, A. Gatti, A. Ferrando, VEsNA goes very fast: Event-based BDI control in Formula One racing, 2026. Accepted for inclusion in the EMAS 2026 Post Proceedings.
 - [42] Ö. I. Erduran, Machine learning for cognitive BDI agents: A compact survey, in: A. P. Rocha, L. Steels, H. J. van den Herik (Eds.), *Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22–24, 2023, SCITEPRESS*, 2023, pp. 257–268. URL: <https://doi.org/10.5220/0011678100003393>. doi:10.5220/0011678100003393.
 - [43] M. Parmiggiani, A. Ferrando, V. Mascardi, Together is better! Integrating BDI and RL agents for safe learning and effective collaboration, in: A. P. Rocha, L. Steels, H. J. van den Herik (Eds.), *Proceedings of the 17th International Conference on Agents and Artificial Intelligence, ICAART 2025 - Volume 3, Porto, Portugal, February 23–25, 2025, SCITEPRESS*, 2025, pp. 48–59. URL: <https://doi.org/10.5220/0013077000003890>. doi:10.5220/0013077000003890.
 - [44] A. Badica, C. Badica, M. Ivanovic, D. Mitrovic, An approach of temporal difference learning using agent-oriented programming, in: *20th International Conference on Control Systems and Computer Science, CSCS 2015, Bucharest, Romania, May 27–29, 2015, IEEE*, 2015, pp. 735–742. URL: <https://doi.org/10.1109/CSCS.2015.71>. doi:10.1109/CSCS.2015.71.
 - [45] C. Badica, A. Becheru, S. Felton, Integration of Jason reinforcement learning agents into an interactive application, in: T. Jebelean, V. Negru, D. Petcu, D. Zaharie, T. Ida, S. M. Watt (Eds.), *19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Timisoara, Romania, September 21–24, 2017, IEEE Computer Society*, 2017, pp. 361–368. URL: <https://doi.org/10.1109/SYNASC.2017.00065>. doi:10.1109/SYNASC.2017.00065.
 - [46] M. Persiani, T. Hellström, The mirror agent model: A bayesian architecture for interpretable agent behavior, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), *Explainable and Transparent AI and Multi-Agent Systems - 4th International Workshop, EXTRAAMAS 2022, Virtual Event, May 9–10, 2022, Revised Selected Papers, volume 13283 of Lecture Notes in Computer Science*, Springer, 2022, pp. 111–123. URL: https://doi.org/10.1007/978-3-031-15565-9_7. doi:10.1007/978-3-031-15565-9_7.
 - [47] A. Y. Ichida, F. Meneguzzi, Modeling a conversational agent using BDI framework, in: J. Hong, M. Lanperne, J. W. Park, T. Cerný, H. Shahriar (Eds.), *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC 2023, Tallinn, Estonia, March 27–31, 2023, ACM*, 2023, pp. 856–863. URL: <https://doi.org/10.1145/3555776.3577657>. doi:10.1145/3555776.3577657.
 - [48] E. Norling, Folk psychology for human modelling: Extending the BDI paradigm, in: *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, 19–23 August 2004, New York, NY, USA, IEEE Computer Society, 2004, pp. 202–209. URL: <https://doi.ieeecomputersociety.org/10.1109/AAMAS.2004.10066>. doi:10.1109/AAMAS.2004.10066.
 - [49] R. Bellman, On the theory of dynamic programming, *Proceedings of the National Academy of Sciences* 38 (1952) 716–719. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.38.8.716>. doi:10.1073/pnas.38.8.716. arXiv:<https://www.pnas.org/doi/pdf/10.1073/pnas.38.8.716>.
 - [50] P. Lokuge, D. Alahakoon, Reinforcement learning in neuro BDI agents for achieving agent's

- intentions in vessel berthing applications, in: 19th International Conference on Advanced Information Networking and Applications (AINA 2005), 28-30 March 2005, Taipei, Taiwan, IEEE Computer Society, 2005, pp. 681–686. URL: <https://doi.org/10.1109/AINA.2005.293>. doi:10.1109/AINA.2005.293.
- [51] P. Lokuge, D. Alahakoon, Handling multiple events in hybrid BDI agents with reinforcement learning: A container application, in: C. Chen, J. Filipe, I. Seruca, J. Cordeiro (Eds.), ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005, 2005, pp. 83–90.
- [52] S. Airiau, L. Padgham, S. Sardiña, S. Sen, Enhancing the adaptation of BDI agents using learning techniques, *Int. J. Agent Technol. Syst.* 1 (2009) 1–18. URL: <https://doi.org/10.4018/jats.2009040101>. doi:10.4018/JATS.2009040101.
- [53] G. Qi, W. Bo-ying, Study and application of reinforcement learning in cooperative strategy of the robot soccer based on BDI model, *International Journal of Advanced Robotic Systems* 6 (2009). doi:10.5772/6795.
- [54] S. Lee, Y. J. Son, Dynamic learning in human decision behavior for evacuation scenarios under BDI framework, 2009. URL: <https://api.semanticscholar.org/CorpusID:111254163>.
- [55] A. Tan, Y. Ong, A. Tapanuj, A hybrid agent architecture integrating desire, intention and reinforcement learning, *Expert Syst. Appl.* 38 (2011) 8477–8487. URL: <https://doi.org/10.1016/j.eswa.2011.01.045>. doi:10.1016/J.ESWA.2011.01.045.
- [56] Q. Wan, W. Liu, L. Xu, J. Guo, Extending the BDI model with Q-learning in uncertain environment, in: Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence, ACAI 2018, Sanya, China, December 21-23, 2018, ACM, 2018, pp. 33:1–33:6. URL: <https://doi.org/10.1145/3302425.3302432>. doi:10.1145/3302425.3302432.
- [57] M. Bosello, A. Ricci, From programming agents to educating agents - A Jason-based framework for integrating learning in the development of cognitive agents, in: L. A. Dennis, R. H. Bordini, Y. Lespérance (Eds.), Engineering Multi-Agent Systems - 7th International Workshop, EMAS 2019, Montreal, QC, Canada, May 13-14, 2019, Revised Selected Papers, volume 12058 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 175–194. URL: https://doi.org/10.1007/978-3-030-51417-4_9. doi:10.1007/978-3-030-51417-4_9.
- [58] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling network architectures for deep reinforcement learning, in: M. F. Balcan, K. Q. Weinberger (Eds.), Proceedings of The 33rd International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, 2016, pp. 1995–2003. URL: <https://proceedings.mlr.press/v48/wangf16.html>.
- [59] R. S. Sutton, A. G. Barto, Reinforcement learning - an introduction, 2nd Edition, MIT Press, 2018. URL: <http://www.incompleteideas.net/book/the-book-2nd.html>.
- [60] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Using reward machines for high-level task specification and decomposition in reinforcement learning, in: J. G. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018, Proceedings of Machine Learning Research, PMLR, 2018, pp. 2112–2121. URL: <http://proceedings.mlr.press/v80/icarte18a.html>.
- [61] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, *J. Artif. Intell. Res.* 73 (2022) 173–208. URL: <https://doi.org/10.1613/jair.1.12440>. doi:10.1613/JAIR.1.12440.
- [62] A. Gatti, F. Casale, V. Mascardi, A. Stucchi, A. Ferrando, Vesna-pro: Exploiting bdi agents with propensities for emergent narrative, in: Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '26, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2026, p. 967–976. URL: <https://doi.org/10.65109/PDPF4024>. doi:10.65109/PDPF4024.