Experimenting an approach to Neuro-Symbolic RL

Andrea Rafanelli[1,2,*], Stefania Costantini[2] and Giovanni De Gasperis[2]

[1]*University of Pisa, Largo B. Pontecorvo, Pisa, 57127, Italy*
[2]*University of L'Aquila, Via Vetoio, L'Aquila, 67100, Italy*

Abstract
In past work, we defined an approach to neuro-symbolic Reinforcement Learning (RL), where logical rules, whose applicability can be either upon conditions and/or checked periodically, influence a reinforcement learning agent in exploring an environment. In this paper, we focus on the experiments that we made to show the effectiveness of the blended approach.

Keywords
Reinforcement Learning, Logical Agents, Neuro-Symbolic AI, Exploration

1. Introduction

A particularly significant application of Reinforcement Learning (RL) is to empower the ability of robots to autonomously explore unknown environments. Such autonomous exploration capabilities serve as a keystone in various crucial applications where robots play pivotal roles, including identifying areas of interest, retrieving specific targets, and much more.

Robotic explorations are especially pertinent for operations in outdoor, real-world environments, where human explorers would be in danger, or incapable of accessing the territory in question because of natural obstacles or obstruction due, e.g., to catastrophic events of various kinds. Indeed, these agents can actively participate in search and rescue missions, environmental monitoring, and other activities necessitating navigation through unfamiliar settings. Deep Reinforcement Learning [2, 3, 4] has proven to be one of the most effective methodologies for training this kind of agents; however, despite some promising results, several significant challenges still need to be addressed. These challenges include—but are not limited to— the poor adaptability and generalization of these systems. This shortcoming hinders their effectiveness in intricate real-world settings that would, in principle, demand capabilities like logical reasoning, causal inference, and commonsense.

According to [5], the majority of RL research has traditionally focused on leveraging representational models to encapsulate the fundamental components of the RL framework–rewards, states, and actions-—while largely disregarding the potential benefits of incorporating rich, high-level declarative knowledge. Leaving out this knowledge restricts RL's potential to take

IPS-RCRA-SPIRIT 2023: Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 7-9th, 2023, Rome, Italy [1]
*Corresponding author.
andrea.rafanelli@phd.unipi.it (A. Rafanelli); stefania.costantini@univaq.it (S. Costantini); giovanni.degasperis@univaq.it (G. D. Gasperis)

advantage of the information and insights that could enhance its problem-solving abilities in multifaceted, dynamic settings. In [6], we tried to bridge the gaps between reinforcement learning's current capabilities and its potential to handle complex real-world problems. This by integrating declarative knowledge to demonstrate that it can serve as a critical driver in overcoming the RL systems' challenges. As an example, in this work, we implemented and tested a deep reinforcement learning agent meant to search for specific targets in simulated environments, which, in critical stages of its operation, is aided by (small blocks of) symbolic rules. More specifically, we developed a deep Q-learning neural network agent able to explore procedurally generated virtual environments and locate target objects, improving its search performance through a reinforcement learning training process. The agent starts naive but gradually enhances its search skills by receiving virtual rewards and punishments based on its actions.

To augment the agent's neural network functioning, we also implemented a symbolic reasoning component inspired by the *subsumption architecture* [7], where reactive rules of various priorities guide an agent's behavior. This module overrides the neural controls when needed to improve the flexibility and context-awareness of the agent's search behavior. In essence, the agent fuses reinforcement learning with symbolic reasoning to exhibit increasingly intelligent target search abilities as it gains experience in the training environments. In this paper, after summarizing our past work, we illustrate and discuss the experiments that have been done, to show in practice the effectiveness of the approach.

The paper is structured as follows. In Section 2, we shortly recall Reinforcement learning (and in particular Deep Reinforcement Learning), Neuro-Symbolic AI, and the DALI language, i.e., the logic agent-oriented programming language that we adopted in our solution. In Section 3, we discuss some related work. In Section 4, we introduce our explorer agent as defined in [6]. In Section 5, we present and discuss the experimental evaluation of our solution, and, finally, in Section 6, we conclude.

2. Preliminaries

The following section provides overviews of RL and neuro-symbolic integration techniques.

2.1. Reinforcement Learning

Reinforcement learning [8] is a technique where agents learn through "trial and error" experience within an environment. Reinforcement learning agents explore and experiment in their environment aimed at maximizing cumulative future rewards, typically represented numerically. For instance, an agent may try to reach a goal state that provides a positive reward.

The main elements in RL methods are: (i) the **policy**, π, is a mapping from environment states to agent actions $\pi(s) : S \rightarrow A$, that specifies what action the agent should take in any given situation, where the state s refers to the current situation the agent finds itself, based on previous actions and observations; (ii) the **reward**, r which is a function of the state and action describing the numerical payoff received for taking a particular action a from state s; the agent's goal is to maximize the overall reward it receives over the long run; (iii) the **value-function**, $V_\pi(s)$, representing the expected cumulative reward from state s onwards, that should be obtained when following policy π.

2.1.1. Deep Q-Learning:

Q-learning is a so-called value-based RL algorithm aimed at estimating the value of state-action pairs to determine future actions. This approach relies on *Q-Table*, which is a grid with two dimensions: one representing different states and the other representing possible actions to take. In this grid, each cell holds a value that represents the maximum expected future reward for a particular combination of state and action. This value is known as the *Q-value*. The Q-table is updated iteratively through a training process.

Deep Q-learning (DQL) [9] is a variation of Q-Learning that replaces the traditional Q-table with a neural network that takes the current state as input and approximates the Q-values for each possible action based on that state. The learning process remains the same, with iterative updates, but instead of adjusting the Q-Table, the neural network's weights are updated to improve its output.

2.2. Neuro-Symbolic AI

The field of Neuro-Symbolic AI offers a promising path forward to master both intricate perceptual patterns and abstract conceptual reasoning. Thus, Neuro-Symbolic (NeSy) systems integrate structured symbolic representations with neural learning, aiming to combine their complementary strengths.

NeSy systems encompass a rich diversity of architectures combining neural networks and symbolic systems in different ways. One influential taxonomy proposed by Kautz [10] (see also the references therein) outlines six different possible architectures: *(i) Symbolic Neuro Symbolic*, involves translating symbolic input into feature vectors for neural networks, which subsequently yield results in symbolic form. *(ii) Symbolic[Neuro]*, involves a neuro pattern recognition subroutine that operates within a symbolic problem-solving framework that emphasizes symbolic reasoning. *(iii) Neuro:Symbolic*, involves a sequential flow of information, starting from neural networks and cascading into a symbolic reasoner. *(iv) Neuro : Symbolic →Neuro*, involves compile symbolic rules within neural network training process. *(v) Neuro_{Symbolic}*, involves compiling symbolic rules within neural network structure. *(vi) Neuro[Symbolic]* involves the incorporation of symbolic reasoning into a neural engine.

Another notable taxonomy is the one of [11], which categorizes NeSy architectures as: *(i) Learning for reasoning*, where neural networks preprocess data for symbolic modules. *(ii) Reasoning for learning*, where symbolic systems provide knowledge for neural learning. *(iii) Learning-reasoning*, where there is a tight interaction between neural and symbolic reasoning.

In this work, we mainly refer to the taxonomy proposed by [11].

2.3. DALI

To empower autonomous agents with human-like flexibility and adaptiveness, the logical agent-oriented language DALI [12, 13, 14, 15] introduces several innovative event-driven features. DALI agents react to external events in the environment as well as internal triggers stemming from their reasoning. These events enable interleaved activities, planning, and reactive responses.

Specifically, DALI recognizes four event types: *(i)* external, *(ii)* internal, *(iii)* present, and *(iv)* past. External events model environmental occurrences like sensory data. Internal events embody self-generated triggers from the agent's own knowledge and inferences. Present events signify situations the agent is currently aware of but hasn't reacted to yet. Past events represent previous events, allowing reflection.

Crucially, DALI agents can respond to events through declarative reactive rules. The syntax concisely defines triggering conditions and corresponding actions. Internal events prove particularly significant since they allow proactive behaviors based on motivation, curiosity, and goals beyond just external stimuli reactions. DALI agents exploit past events to inform future decisions.

In our work, we leverage DALI reactive rules to implement top-down constraints and behaviors in our agent's symbolic module. These rules react to environmental events detected through the neural network's sensory processing. For example, a rule triggers obstacle avoidance behavior when the neural perception component signals a nearby obstacle.

3. Related Works

We review below related works in neuro-symbolic RL.

From the very beginning, RL has been intertwined with planning. This connection is especially clear in model-based RL, where agents build models of their environment to enable planning. A pioneering system called *Dyna* [16], developed back in the 1990s, nicely demonstrated this idea by combining real-time action selection with behind-the-scenes planning over both real experiences and simulated scenarios.

An innovative method called *Darling* [17] pushed this concept even further, using planning to guide an agent's behavior towards reasonable choices while letting reinforcement learning handle adaptation to the environment.

Leveraging symbolic knowledge for reward shaping is another active area. In [18], the authors provided both a framework for defining goal-based tasks and a method to automatically generate matching reward functions.

Similarly, [19] proposed a way to leverage domain knowledge as a catalyst for boosting the speed and optimality of various RL techniques. Their experiments revealed that this symbolic knowledge-infused reward shaping outperformed other methods, including manual and model-based rewards, when applied in its basic form.

Another area in which the integration of symbolic systems can increase the efficiency of RL systems is the design of policies.

Recent works apply program synthesis to deduce policies from examples and constraints [20, 21]. In [22], the authors introduce an algorithm designed to learn state machine policies that capture repeating behaviors.

Our work is outside these established fields, as a symbolic component influences the policy of the RL agent. In particular, the symbolic reasoning module(s) can provide top-down guidance and structure to accelerate learning. According to the taxonomy proposed by [11], our model falls under the ”reasoning for learning” category of neuro-symbolic reinforcement learning.

4. Explorer Agent

In this section, we detail the different modules that collectively make up the structure of the *Explorer agent* that we devised in [6]. This agent navigated a 2D virtual environment in order to retrieve a target. The agent has a simulated ”body” with 40 optical sensors covering a 120-degree frontal view. These sensors, or ”rays,” return the distance to objects like walls and obstacles. With this input, the agent chooses from three action primitives: `rotate left`, `move forward`, or `rotate right`.

In early experiments, we noticed that random action selection leads to poor exploration. The agent often remains in the same place, turning on itself rather than moving forward. We hypothesized that the 2:1 ratio of turn actions to forward movement is to blame. In fact, by increasing the odds of moving forward to 93%, the agent spends more time traveling and less time spinning in circles. This simple tweak produces more realistic and varied sensor data to train the neural network.

Another design choice is to handle essential obstacle avoidance using a specific symbolic module. This ”avoidance” behavior triggers rotational actions when sensors report too close objects. By allowing the avoidance module to replace the output of the neural network, we prevent the agent from expending unnecessary effort to learn obstacle-avoidance behaviors. In this way, the learning system can devote its resources to higher-level goals, such as full exploration of the environment and sophisticated navigation. Combining random wandering with reactive avoidance, the agent navigates the world from the start, gradually shifting from aimless wandering to purposeful exploration. It turns out that by separating navigation and/from collision detection, we simplify the learning problem.

Neural network: The agent is trained with a fully connected feed-forward neural network (see Figure 1). The input layer intakes a 40-element vector of pre-processed sensor data.

The initial hidden layers employ densely connected nodes to identify low-level patterns. A utility layer then ”flattens” the multidimensional input into a 1D array. The hidden layers utilize Rectified Linear Unit (ReLU) [23] activations to introduce non-linearities.

The widening progression of layers transforms the compact input into an enriched feature space. This expanded representation collapses into a compact output layer signaling the desired action. Softmax activation allows comparison of the three discrete movement options (`turn right`, `turn left`, `move forward`). The final output constitutes the network’s navigation policy recommendation. The loss function employed in this context is the Mean Square Error (MSE) [24], which aligns with the formula derived from the Q-Function upon which the network conducts its optimization. The deep learning algorithm in use is propelled by the Adam stochastic gradient [25] optimization method.

Symbolic supervisor: The logic module allows the neural network to focus its resources on navigation. The logic provides capabilities for obstacle avoidance and door detection. Obstacle

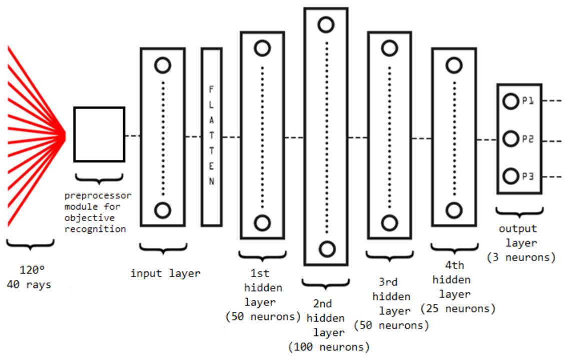
120°
40 rays
preprocessor module for objective recognition
input layer
FLATTEN
1st hidden layer (50 neurons)
2nd hidden layer (100 neurons)
3rd hidden layer (50 neurons)
4th hidden layer (25 neurons)
P1
P2
P3
output layer (3 neurons)

Figure 1: Deep learning model for the Explorer agent's controller. The input layer receives distance readings from proximity sensors, and the output layer determines the agent's movement direction or rotation.

perception relies on the same sensory array as the neural network. For doors, we simulate Bluetooth emitters with a 1-2 meter range, detectable when near. The logic rules have the highest priority, thus overriding the network's outputs if necessary. Reactive behaviors are defined by exploiting DALI reactive rules; these are *condition-action-rules* designed such that, where the event in the *head* (lefthand part) of the rule is detected, the *body* (righthand part of the rule, after special token :>) is executed; the body may include reasoning tasks and actions that are devised in consequence. The two simple behaviors that we defined are shown below.

```
obtstacleE(D) :> direction(D),
                 right(D), rotateA(left_angle);
                 left(D),  rotateA(right_angle);

seen_door(D) :> direction(D),
                rotateA(D), forwardA(1).
```

The first rule governs obstacle avoidance behavior. When an obstacle is detected in direction D (*obstacleE(D)*), the agent sets its current facing to D (*direction(D)*), determines if the obstacle is positioned to the right or left of D (*right(D)* or *left(D)*), and executes a rotational action to avoid the obstacle by turning left or right by a fixed angle (*rotateA(left_angle)* or *rotateA(right_angle)*). This allows the agent to redirect its movement away from obstacles and, in particular, to rotate away from walls and furniture. This prevents the agent from countless collisions.

The second rule aims to push the agent towards the nearest door using the agent's current position. When a door is detected in direction D (*seen_door(D)*), the agent sets its facing towards the door (*direction(D)*), rotates to directly face the door (*rotateA(D)*), and executes a forward movement action to approach the door (*forwardA(1)*).

Indeed, the door detection logic offers–by encoding the very concept of doorways–the abstract strategic understanding required to systematically leave rooms where the agent has spent some time already. This prevents the agent from wandering uselessly before grasping a fundamental spatial logic. Rather, the agent can thus parse its environment into distinct zones and travel between them.

Training: The agent's core reinforcement learning algorithm allows it to improve through trial-and-error experience in the procedurally generated environments. To further enhance learning, we utilize the proven *Remember & Replay* technique [26]. The agent stores experiences in memory as transition tuples–state, action, reward, next state, terminal–. This captures the full flow from one observation to the next. The terminal flag indicates the episode ends due to collision, target find (score), or timeout. The pseudo-code of the training algorithm is reported in Figure1.

Algorithm 1 Pseudo-code for the Explorer agent training

```
1: for $i$ in range(1, episodes) do
2:     $terminal \leftarrow$ False
3:     $frame\_count, random\_actions, room\_changes \leftarrow 0, 0, 0$
4:     $reward\_accumulator, score\_accumulator \leftarrow 0, 0$
5:     reset_objective()
6:     reset_agent()
7:     set_initial_state()
8:     while not $terminal$ do
9:         $frame\_count \leftarrow frame\_count + 1$
10:        $reward \leftarrow 0$
11:        if $logical\_driver$ then
12:            $action, was\_it\_random \leftarrow$ slam_agent.act_logic($state$)
13:        else
14:            $action, was\_it\_random \leftarrow$ slam_agent.act($state$)
15:        end if
16:        $reward, score \leftarrow$ make_reward()
17:        $terminal \leftarrow$ check_episode_end()
18:        $state \leftarrow$ environment.update_sensory_input()
19:        slam_agent.remember($state, action, reward, terminal$)
20:        $room\_changed \leftarrow$ visual_scene_update($action$)
21:        if $was\_it\_random$ then
22:            $random\_actions \leftarrow random\_actions + 1$
23:        end if
24:        if $room\_changed$ then
25:            $room\_changes \leftarrow room\_changes + 1$
26:        end if
27:        $reward\_accumulator \leftarrow reward\_accumulator + reward$
28:        $score\_accumulator \leftarrow score\_accumulator + score$
29:    end while
30:    slam_agent.replay()
31: end for
```

Here, the score refers to the object that the RL agent is trying to find within the generated virtual environment. This target object provides a training signal for the agent–by attempting to locate this object during episodes, the agent can learn an optimal policy for taking actions within the environment. Every time the agent starts a new try, called an ”episode”, it can move around and take different actions to try and locate the target objects. If it does find the object, the agent gets a reward.

Accordingly, for the reward scheme, we defined different situations:

- **Reward** = **-1**, small penalty after every 10 frames elapsed. This prevents dawdling and encourages efficiency.
- **Reward** = **1**, for changing rooms. Reinforces systematic area coverage.
- **Reward** = **3**, for looking towards target when in same room. Aligns with goal-finding.
- **Reward** = **5**, for moving towards target. Motivates motion planning.
- **Reward** = **10**, for reaching target. This strongly reinforces successful mission completion.

The choice to have different rewards at multiple levels has been motivated by wanting to incentivize (or disincentivize) the agent on multiple aspects. Room change rewards drive coverage and exploration of the area at a high level. The view angle reward tunes the agent's attention. The target approach reward models basic navigation. Finally, the payoff for the end goal consolidates overall success.

5. Experimental Evaluation

Our experiments compare the pure RL agent against the hybrid neuro-symbolic version over 600 training episodes. The agents navigate artificially generated virtual environments with new target locations for each episode. Performance metrics track (per episode) room coverage, reward, randomness, and an overall target score.

For the sake of reproducibility, the source code of our experiments is public and available[1].

Environment: To provide a rich training environment, we procedurally generated unique 2D house layouts (c.f. [6] for a detailed explanation) using constraint logic programming (CLP) [27, 28].

CLP allowed us to declaratively define relationships and constraints to algorithmically create plausible spaces. We used CLP(R) [29], which extends logic programming with real number arithmetic. This enabled asserting geometric constraints between room sizes, positions, boundaries, and doorways and equipping rooms with suitable furniture. The CLP system searches for layouts satisfying all constraints.

Key rules include:

- rooms connect to a central antechamber;

[1]https://github.com/AAAI-DISIM-UnivAQ/Neural-Logic-Reinforcement-Learning

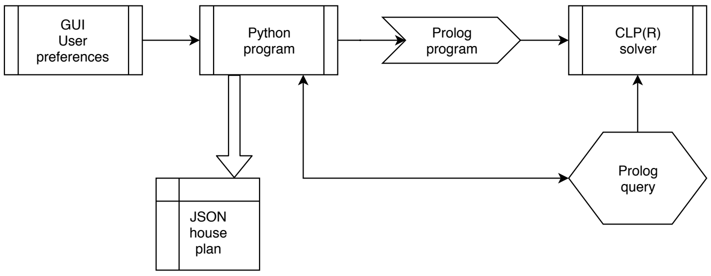

Figure 2: Data flow to obtain the constrained generated virtual house plan

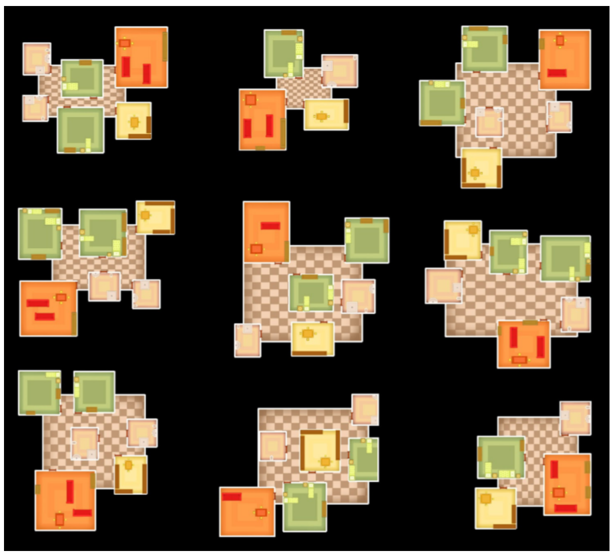

Figure 3: A set of generated home-like spaces derived from the constraints.

- no room overlapping;
- room areas don't exceed the antechamber area;
- each room has one door;
- realistic furniture positioning is provided.

The user sets high-level parameters like the number and types of rooms, which can be dining rooms, bedrooms, etc. The CLP generator then formulates the specific constraints and relationships needed to create structures matching the requested configuration. After the CLP system successfully identifies a layout meeting all constraints, that layout gets passed along to our Python module responsible for realizing the 2D environment file (see Figure2).

In Figure3, we show some results of the generation.

Experiment Design: We extensively trained the two agent types across 600 episodes, enabling robust comparison. Agents navigated a generative environment with 4 unique rooms

–hall, kitchen, bedroom, bathroom– populated with varying obstacles.

We utilized a memory replay buffer of 100 experiences to augment online learning. By replaying past events, agents could reiterately learn from rare or critical cases. This improved sample efficiency and stability.

The training concluded after a maximum number of steps proportional to the environment size–in our setting 1500–, thus preventing overfitting.

The overall experiment design enabled the assessment of the hybrid neuro-symbolic architecture versus standard reinforcement learning under equivalent settings.

Metrics: To thoroughly compare the pure reinforcement learning agent against the hybrid neuro-symbolic version, we employed complementary evaluation metrics. These provide multi-faceted quantifications of performance per episode.

The first metric simply tracks the number of room changes achieved. The second metric calculates the percentage of random actions taken. Lower randomness indicates more purposeful and efficient navigation. The third metric looks at the cumulative reward obtained by the agent. It provides a summary of success on the incremental goal curriculum. More reward implies proper object viewing, approach, and target finding. Finally, a composite score accounts for the amount of target objects reached by the agent.

Analyzing these diverse metrics reveals the strengths and weaknesses of each agent design. The room changes metric highlights exploration capabilities. Action randomness evaluates efficiency. Reward accumulation reflects goal achievement. And the overall score measures the capacity of the agent to reach target objects.

5.1. Discussion

Our experiments offer relevant insights into the benefits of incorporating logic modules into learning agents. The core objective was to assess how logic augmentation impacts performance across training episodes. We tracked two agent types–with and without logic–over hundreds of simulated trials.

Analysis of random action rates (Figure 4) provides a window into learned behaviors. Initially, both agents act randomly as their policies are untrained. Then, the randomness declines steadily as experience grows, indicating learned determinism. However, the non-logic agent plateaued after 300 episodes, suggesting limits in its learning. In contrast, the logic-enhanced agent continued improving, leveraging structured knowledge to advance beyond the neural network's capabilities. In a way, the rules provided to the agent, like room changes and door detection, enabled higher-level learning.

The examination of reward accumulation (Figure 5) paints a nuanced picture of goal achievement. Reward incentives were provided for behaviors like room changes, target viewing, and target reaching. We found that both agent types accrued rewards through training. However, the logic agent displayed more frequent reward spikes than its non-logic counterpart.

Room change rates (Figure 6) quantify the exploratory drive essential for environmental mastery. Here major differences emerged, with the logic agent changing rooms at a higher rate across the episodes than the non-logic one. In a way, the logic's incentives for systematic area

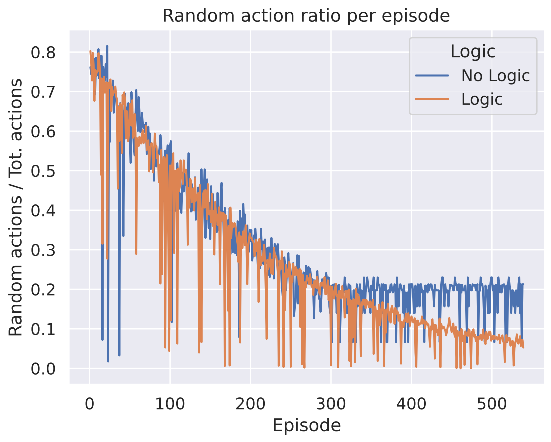

Figure 4: Ratio of random actions taken by the agents in each training episode.

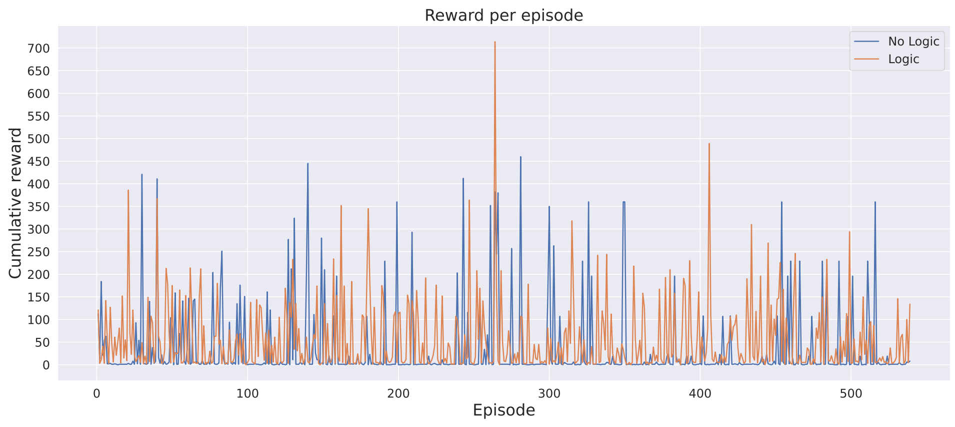

Figure 5: Cumulative reward accrued by the agents within each episode during training.

coverage drove extensive exploration. The non-logic agent often lingered in the same room without this top-down guidance.

Finally, overall score distributions (Figure 7) highlight comprehensive performance. The non-logic agent scored zero on 80% of episodes, failing to find targets. Meanwhile, the logic agent scored zero just 60% of the time while accruing top scores more frequently. The logic allowed efficient parsing of the environment into navigable rooms and objectives.

Overall, these extensive results provide clear evidence that combining neural networks with structured knowledge can profoundly enhance learning agents. Logic modules compensate for limitations in solo neural learning by injecting new competencies.

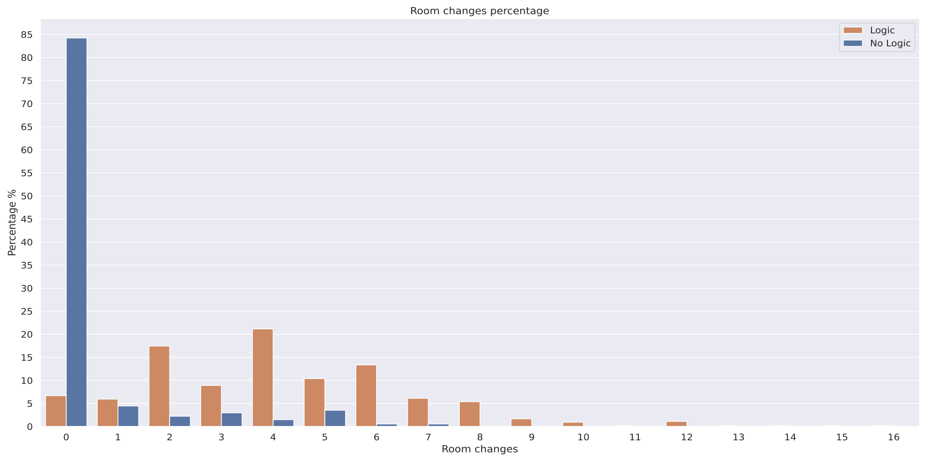

Figure 6: Percentage of room changes by the agents over the experiments.

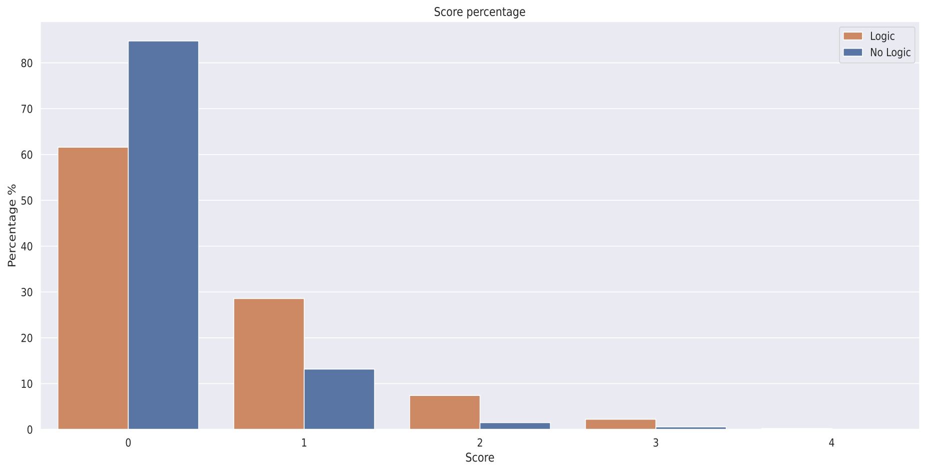

Figure 7: Percentage of score gained by the agents over the experiments.

6. Conclusion

In this work, we review the definition and implementation of a virtual agent that we developed in recent past work, and we discuss the experimental evaluation that we made. Our objective, which turns out to have been successfully reached, was to demonstrate how the integration of symbolic knowledge can enhance reinforcement learning processes adopted for real-world tasks.

Our agent leveraged deep Q-learning to search procedurally generated realistic environments, to locate target objects. A key contribution has been the addition of a symbolic reasoning module, inspired by the subsumption architecture, to augment the capabilities of the RL agent. The symbolic component is very small. Nonetheless, it has been useful as a proof of concept to outline the possibility of improving RL agents using such a methodology consisting of logically inducing the agent to avoid shortcomings that might slow down or even make its operation inconclusive. In fact, in real-world settings, there are in general, time deadlines to reach an objective, which becomes less useful or superfluous if reached too late.

The experiments provided strong validation that combining flexible neural learning with reactivity dictated by structured knowledge can help overcome major challenges in applying RL to complex real-world problems. Notice that the reasoning tasks in our reactive rules are indeed basic: however, any kind of reasoning can be incorporated therein. The right balance between exploration and reasoning has itself to be experimentally devised. In our case, we applied the reactive rules whenever applicable. An alternative might be to invoke them at a certain frequency or after the RL agent has been stuck for a certain limit time. The logic-infused agent exhibited more efficient and effective search behaviors than a pure learning agent. Indeed, the logic module allowed the neural network to concentrate resources on higher-level planning instead of focusing on low-level skills.

Overall, this work helps bridge the gap between RL's current capabilities and its future potential for autonomous exploration, discovery, and decision-making in complex spaces. We argue that the logic components can compensate for issues like poor generalization, sample inefficiency, and lack of transparency that impede purely data-driven neural agents. There remain ample opportunities to further enhance the integration of human knowledge into RL systems. Promising directions include probabilistic logic programming to handle uncertainty, causal reasoning to understand interactions, and meta-learning techniques to optimize knowledge infusion. With further research, neuro-symbolic RL can perhaps better approach human-like intelligence.

References

[1] R. De Benedictis, N. Gatti, M. Maratea, A. Murano, E. Scala, L. Serafini, I. Serina, E. Tosello, A. Umbrico, M. Vallati, Preface to the Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (IPS-RCRA-SPIRIT 2023), in: Proceedings of the Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (IPS-RCRA-SPIRIT 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI* IA 2023), 2023.

[2] Y. Kato, K. Morioka, Autonomous robot navigation system without grid maps based on double deep q-network and RTK-GNSS localization in outdoor environments, in: IEEE/SICE International Symposium on System Integration, SII 2019, Paris, France, January 14-16,

2019, IEEE, 2019, pp. 346–351. doi:10.1109/SII.2019.8700426.
URL https://doi.org/10.1109/SII.2019.8700426

[3] A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, P. Valigi, Towards generalization in target-driven visual navigation by using deep reinforcement learning, IEEE Trans. Robotics 36 (5) (2020) 1546–1561. doi:10.1109/TRO.2020.2994002.
URL https://doi.org/10.1109/TRO.2020.2994002

[4] F. Zeng, C. Wang, W. Wang, Visual navigation with asynchronous proximal policy optimization in artificial agents, J. Robotics 2020 (2020) 8702962:1–8702962:7. doi:10.1155/2020/8702962.
URL https://doi.org/10.1155/2020/8702962

[5] C. Yu, X. Zheng, H. H. Zhuo, H. Wan, W. Luo, Reinforcement learning with knowledge representation and reasoning: A brief survey, CoRR abs/2304.12090. arXiv:2304.12090, doi:10.48550/arXiv.2304.12090.
URL https://doi.org/10.48550/arXiv.2304.12090

[6] G. De Gasperis, S. Costantini, A. Rafanelli, P. Migliarini, I. Letteri, A. Dyoub, Extension of constraint-procedural logic-generated environments for deep Q-learning agent training and benchmarking, Journal of Logic and Computation (2023) exad032arXiv:https://academic.oup.com/logcom/advance-article-pdf/doi/10.1093/logcom/exad032/50530031/exad032.pdf, doi:10.1093/logcom/exad032.
URL https://doi.org/10.1093/logcom/exad032

[7] R. A. Brooks, Intelligence without reason, in: J. Mylopoulos, R. Reiter (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991, Morgan Kaufmann, 1991, pp. 569–595.
URL http://ijcai.org/Proceedings/91-1/Papers/089.pdf

[8] R. S. Sutton, A. G. Barto, Reinforcement learning - an introduction, Adaptive computation and machine learning, MIT Press, 1998.
URL https://www.worldcat.org/oclc/37293240

[9] 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 (7540) (2015) 529–533. doi:10.1038/nature14236.
URL https://doi.org/10.1038/nature14236

[10] H. A. Kautz, The third AI summer: AAAI robert s. engelmore memorial lecture, AI Mag. 43 (1) (2022) 93–104. doi:10.1609/aimag.v43i1.19122.
URL https://doi.org/10.1609/aimag.v43i1.19122

[11] D. Yu, B. Yang, D. Liu, H. Wang, S. Pan, A survey on neural-symbolic learning systems, Neural Networks 166 (2023) 105–126. doi:https://doi.org/10.1016/j.neunet.2023.06.028.
URL https://www.sciencedirect.com/science/article/pii/S0893608023003398

[12] S. Costantini, A. Tocchio, A logic programming language for multi-agent systems, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings, Vol. 2424 of Lecture Notes in Computer Science, Springer, 2002, pp. 1–13. doi:10.1007/3-540-45757-7_1.
URL https://doi.org/10.1007/3-540-45757-7_1

[13] S. Costantini, A. Tocchio, The DALI logic programming agent-oriented language, in: J. J. Alferes, J. A. Leite (Eds.), Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings, Vol. 3229 of Lecture Notes in Computer Science, Springer, 2004, pp. 685–688. doi:10.1007/978-3-540-30227-8_57.
URL https://doi.org/10.1007/978-3-540-30227-8_57

[14] S. Costantini, G. De Gasperis, V. Pitoni, A. Salutari, DALI: A multi agent system framework for the web, cognitive robotic and complex event processing, in: D. D. Monica, A. Murano, S. Rubin, L. Sauro (Eds.), Joint Proceedings of the 18th Italian Conference on Theoretical Computer Science and the 32nd Italian Conference on Computational Logic co-located with the 2017 IEEE International Workshop on Measurements and Networking (2017 IEEE M&N), Naples, Italy, September 26-28, 2017, Vol. 1949 of CEUR Workshop Proceedings, CEUR-WS.org, 2017, pp. 286–300.
URL https://ceur-ws.org/Vol-1949/CILCpaper05.pdf

[15] G. De Gasperis, S. Costantini, G. Nazzicone, Dali multi agent systems framework, doi 10.5281/zenodo.11042, DALI GitHub Software Repository, DALI: http://github.com/AAAI-DISIM-UnivAQ/DALI (July 2014).

[16] R. S. Sutton, Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, in: B. W. Porter, R. J. Mooney (Eds.), Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990, Morgan Kaufmann, 1990, pp. 216–224. doi:10.1016/b978-1-55860-141-3.50030-4.
URL https://doi.org/10.1016/b978-1-55860-141-3.50030-4

[17] M. Leonetti, L. Iocchi, P. Stone, A synthesis of automated planning and reinforcement learning for efficient, robust decision-making, Artif. Intell. 241 (2016) 103–130. doi:10.1016/j.artint.2016.07.004.
URL https://doi.org/10.1016/j.artint.2016.07.004

[18] L. Illanes, X. Yan, R. T. Icarte, S. A. McIlraith, Symbolic plans as high-level instructions for reinforcement learning, in: J. C. Beck, O. Buffet, J. Hoffmann, E. Karpas, S. Sohrabi (Eds.), Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020, AAAI Press, 2020, pp. 540–550.
URL https://ojs.aaai.org/index.php/ICAPS/article/view/6750

[19] S. Devlin, D. Kudenko, Plan-based reward shaping for multi-agent reinforcement learning, Knowl. Eng. Rev. 31 (1) (2016) 44–58. doi:10.1017/S0269888915000181.
URL https://doi.org/10.1017/S0269888915000181

[20] H. Young, O. Bastani, M. Naik, Learning neurosymbolic generative models via program synthesis, in: Deep Reinforcement Learning Meets Structured Prediction, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=S1gUCFx4dN

[21] L. Valkov, D. Chaudhari, A. Srivastava, C. Sutton, S. Chaudhuri, HOUDINI: lifelong learning as program synthesis, in: S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 8701–8712.
URL https://proceedings.neurips.cc/paper/2018/hash/edc27f139c3b4e4bb29d1cdbc45663f9-Abstract.

html

[22] J. P. Inala, O. Bastani, Z. Tavares, A. Solar-Lezama, Synthesizing programmatic policies that inductively generalize, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.
URL https://openreview.net/forum?id=S1l8oANFDH

[23] A. F. Agarap, Deep learning using rectified linear units (relu), CoRR abs/1803.08375. arXiv:1803.08375.
URL http://arxiv.org/abs/1803.08375

[24] Mean squared error, The Concise Encyclopedia of Statistics (2008) 337–339doi:10.1007/978-0-387-32833-1_251.
URL https://doi.org/10.1007/978-0-387-32833-1_251

[25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1412.6980

[26] L.-J. Lin, Reinforcement learning for robots using Neural Networks, Carnegie-Mellon University. Department of Computer Science, 1993.

[27] C. Lassez, K. McAloon, R. H. C. Yap, Constraint logic programming and option trading, IEEE Expert 2 (3) (1987) 42–50. doi:10.1109/MEX.1987.4307090.
URL https://doi.org/10.1109/MEX.1987.4307090

[28] J. Cohen, Constraint logic programming languages, Commun. ACM 33 (7) (1990) 52–68. doi:10.1145/79204.79209.
URL https://doi.org/10.1145/79204.79209

[29] J. Jaffar, S. Michaylov, P. J. Stuckey, R. H. C. Yap, The CLP(R) language and system, ACM Trans. Program. Lang. Syst. 14 (3) (1992) 339–395.