Proceedings of Machine Learning Research 192:89–107, 2022 International Workshop on Self-Supervised Learning

Reasoning-Learning Systems Based on Non-Axiomatic Reasoning System Theory

Patrick Hammer PATRICK.HAMMER@PSYCHOOLOGY.SU.SE
Department of Psychology
Stockholm University
113 47 Stockholm, Sweden

Editor: Kristinn R. Thórisson

Abstract

In this paper a strong motivation for real-time reasoning-learning systems based on Non-Axiomatic Reasoning System (NARS) Theory as an approach to build intelligent systems with agency is given. This contains the requirement to work under the Assumption of Insufficient Knowledge and Resources which demands open-ended adaptation while obeying to strict computational resource restrictions to allow for real-time response. We show how this aligns with the phenomenon of intelligence as found in nature, allowing for systems which can both react instantly, and plan ahead deliberately dependent on implicitly outcome-dependent time pressures. In this context a specific implementation design is considered, OpenNARS for Applications (ONA), and how its learning and reasoning abilities lead to data-efficient adaptation in novel circumstances in various domains, whereby we compare with a reinforcement learning method, Q-Learning, in Space Invaders, Pong and a grid robot environment. We will see that both techniques perform comparably well for reactive tasks in Markovian environments, while the uncertainty reasoner performs better when the Markov property is violated, with the additional property that it can plan ahead to exploit task compositionality, also taking explicit background knowledge into account.

Keywords: Reasoning Under Uncertainty, Reinforcement Learning, Non-Axiomatic Reasoning, Procedure Learning, Practical Reasoning

1. Introduction

In this paper, after re-visiting considerations derived from research on the phenomenon of natural intelligence, we will see a reasoner able to learn from data (Non-Axiomatic Reasoning System (Wang, 2013)) being compared with a common reinforcement learning technique (Q-Learning (Watkins, 1989)) in a set of reinforcement learning problems. Reinforcement learning is usually seen as orthogonal to practical reasoning, as two sub-areas of AI which serve different purposes. The former is expected to learn behaviors which maximize expected future reward, while the latter is expected to find ways to reach goals based on already available knowledge which can be used for planning purposes. The outcome of the planning process is a sequence of steps which is expected to lead to the desired goal state when starting from current circumstances. In reinforcement learning usually the goal is

fixed and represented as a utility function which provides the agent with different reward values in different circumstances. Also, the planning is implicit: when a certain action is taken it is chosen exactly because it is expected to lead to the highest future reward, but this can still mean that additional steps are required to get the reward, which demands dealing with temporal credit assignment, that is, to evaluate to what degree the taken actions were responsible for the reward outcome (Sutton, 1988). When this situation is common (that only specific states provide reward feedback) we usually refer to this as rewards being "sparse." In practical reasoning terms this is usually the default, as it just means that the only outcome of interest is the achievement of the goal state. Intermediate outcomes don't have value by themselves other than making achieving the goal state easier or more difficult, which to be learned needs involve a form of temporal credit assignment.

To perform an initial comparison with a machine learning technique capable of temporal credit assignment, in this paper we will see 'OpenNARS for Applications' (ONA, see (Hammer and Lofthouse, 2020)) which can reason under uncertainty, competing with a table-based Q-Learner (see (Watkins, 1989)) with eligibility traces in domains with sparse rewards. This will show the key point this paper is about to convey: that uncertainty reasoning can be used to learn behaviors typically learned by reinforcement learners, and reach comparable results in certain domains in which reinforcement learning techniques are typically applied. Additionally, we will see how the reasoning approach performs better when the Markov property of next state only being dependent on previous state and previous action is violated. This result is obtained without relying on state merging heuristics to make the Markov property for reward hold again (Gaon and Brafman, 2020). And compared to Georgeon et al. (2015), which eliminates hidden Markov state dependence in the reward function, non-axiomatic logic with explicit goal statement representations is utilized. Most importantly, this work establishes uncertainty reasoning (based on non-axiomatic logic in particular) as an additional machine learning technique to deal with reinforcement learning problems, and points to a solution which can both deal with reactive tasks and planning, the latter of which requires the kind of causal representations that this reasoner is able to learn.

2. Inspirations from nature

If natural intelligence is to be replicated in computer systems, instead of turning AI into a different field with different objectives, psychological literature has to be taken into account. In psychological experiments intelligence is usually tested by putting evolved systems into situations they did not evolve for (Fig. 1), or an artificial system into situations it was not designed for. These experiments demand the animal to adapt beyond its current knowledge or skill level, and analyzing how they are able to deal with these situations, and how quickly, drives a large part of the related research. Situations which have been thoroughly analyzed include, among others, corvids which drop nuts on asphalt roads from great heights (as analyzed in Cristol et al. (1997)) and are able to perform well in various planning experiments (as in Kabadayi and Osvath (2017)), and bumblebees which were trained to pull a string or roll a ball into a hole to obtain honey by watching a mockup bumblebee or an individual of their own kind carry out the task (Loukola et al., 2017). This goes beyond model-free reinforcement learning, as it is a case of learning and explicitly representing

cause-effect hypotheses obtained from observations. This is clearly desirable to be replicated in AI systems, such as argued for by Pearl (2010), which attempts to model and utilize causal relationships.

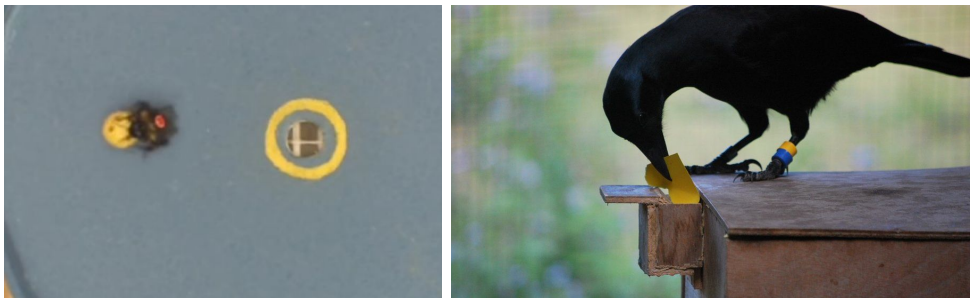

Figure 1: Bumblebee fetch learned from observation, planning in ravens.

Additionally, we can observe a list of considerations obtained from the study of natural intelligence which largely informed the design of our reasoning system and demand the causal structure itself to be learned from observation and interaction with the environment:

1. To be able to transfer knowledge to new goals, knowledge needs to be independent of current utility function / goals; causal $(precondition, operation) \Rightarrow consequent$ representations are a good way to achieve this, and can be chained in ways the agent hasn't experienced previously.

2. Data-efficient learning is key in nature and clearly survival relevant, as slow learning can lead to death rather than just to the jumping from one episode to the next as in a reinforcement learning setting.

3. There is no clear-cut training, test and operating phase in nature, instead learning continues forever, incrementally building on prior knowledge, sometimes referred to as cumulative learning (Thórisson et al., 2019).

4. Learning rate decays do not work for systems which are supposed to be able to continue to adapt with same speed. Ideally an AI system would be able to deal with non-stationary environments, to deal with concept drift (Hu et al., 2020) and to learn causal structure rather than just adjusting existing probability weights.

5. In complex environments, situations never exactly repeat, conditions capture relevant invariances and not complete state. Latent encodings and structural encodings are two ways to achieve this in AI (Hitzler et al., 2022), roughly corresponding to the symbolic and connectionist AI traditions.

6. With sufficient sample counts the system doesn't need to keep track of the size of the sample spaces in which case a probabilistic estimate suffices, however this isn't the case when learning decision-relevant hypotheses from a few examples (Wang, 2009b).

Many of these points show up in the following reasoner design in one way or the other and will help the reader to understand the involved design decisions better.

3. A reasoner which learns and makes decisions

Whilst practical reasoning systems have multiple existing instantiations (such as Ferrein et al. (2012) and Bordini and Hübner (2006)), most are not designed to allow knowledge to be uncertain but rely on it to be sufficient for the task at hand. Multiple logics have been proposed to support reasoning under uncertainty, such as Markov logic networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007), fuzzy logic (Zadeh, 1988), probabilistic logic networks (Goertzel et al., 2008), non-axiomatic logic (Wang, 2013). What all these have in common is extending truth value of propositions from boolean to a degree of belief. This allows them to capture knowledge which is not either true or false, but somewhere in-between. Of these logics, De Raedt et al. (2007), Richardson and Domingos (2006) and Goertzel et al. (2008) operate with probability values associated to the prepositions. To take into account the size of the sample spaces, Goertzel et al. (2008) and Wang (2013) use a second value which intuitively speaking corresponds to the stability of the probability in light of new evidence. This allows them to allocate a higher certainty to say a 50/50 over a 5/5 coin flip scenario, while still converging to the same truth value in the limit of infinite samples. This makes these two logics extremely well-suited for cases where "degree of belief" has to be estimated from samples and justifiable conclusions should be drawn (or decisions being made) even when samples supporting a relevant hypothesis are low in count. In this case the ratio of confirming cases over total cases is not yet representative and the amount of samples needs to be considered in addition.

For this paper, NAL (Wang, 2013) was chosen over PLN (Goertzel et al., 2008) since it incorporates goal reasoning and decision making, hence can be considered a Practical Reasoner able to learn from experience. For this paper we will show only the NAL definitions necessary to replicate the included experiments.

Before we go into the details of truth calculation, Fig. 2 helps to understand the structure of the overall architecture.[1] It consists of:

- Event providers, which consist of dedicated sensor processing for different modalities, each encoding information as statements to reason on.
- FIFO sequencer, a sliding window over the recent events responsible for building sequences of recent events. This component is also responsible for the building and strengthening of temporal implication links.
- A Cycling Events Queue, which is a Priority Queue datastructure. It acts as the system's central attention buffer. All input and derived statements enter there, but only a minority can be selected in a given time-frame. Most are removed due to the fixed capacity of this structure and the pressure of new arriving events of higher priority.
- Concept Memory: the long-term memory of the system. For the sake of this publication, this block stores temporal hypotheses and supports their strengthening and weakening based on their prediction success.

1. While the FIFO has been removed and an explicit temporal inference block added in recent versions of ONA (to condition on derived events), the comparisons in this paper are based on the design with FIFO

- Sensorimotor Inference block: it invokes the Decision and Subgoaling algorithm for goal events selected from the Cycling Events Queue, to be introduced later.
- Declarative Inference block responsible for feature association, prototype formation, relational reasoning and usage of human-provided knowledge. For this publication, this block was not utilized as it would make comparisons more difficult.

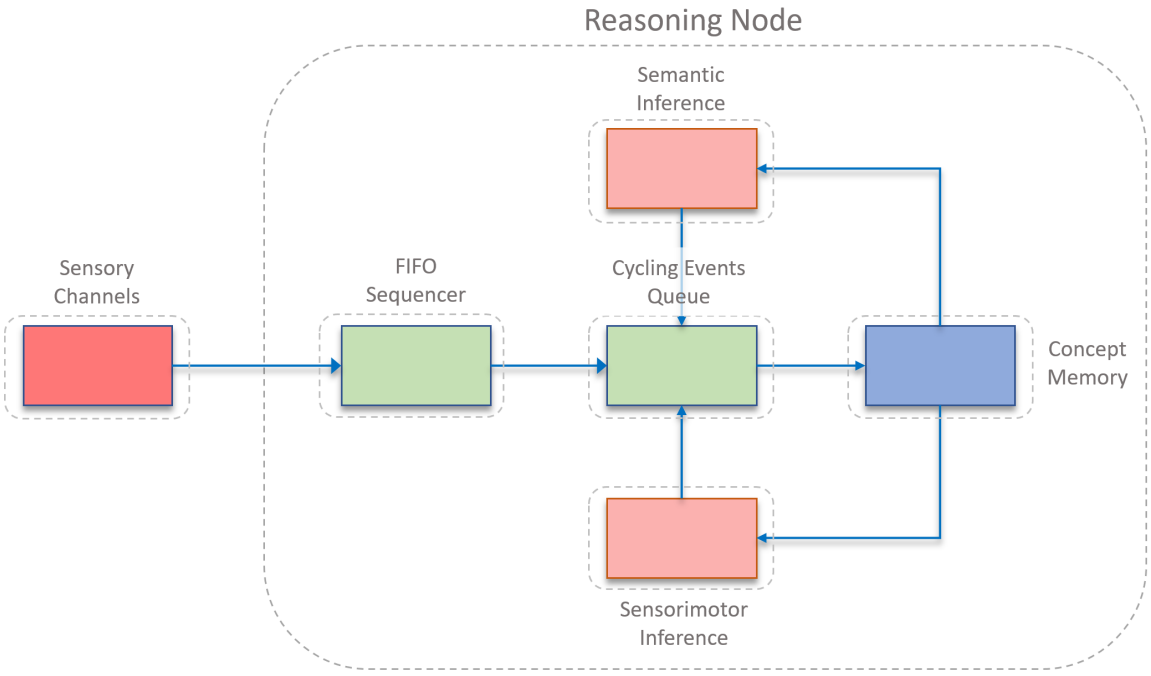

Figure 2: ONA Architecture.

Truth Value Truth Value in NAL is based on positive evidence w_+ and negative evidence w_- which speaks for or against a statement / belief / hypothesis, and the total evidence $w := w_+ + w_-$, each of which is zero or greater. Based on these evidence values, the NAL truth value is defined as the tuple (f, c) with frequency

$$f := \frac{w_+}{w} \in [0, 1]$$

and confidence

$$c := \frac{w}{w + 1} \in [0, 1).$$

Please note the similarity between frequency and probability, with the difference being that the limit $\lim_{w\to\infty} f$ is not taken, as it cannot be obtained from any finite amount of samples. Also, clearly for $w > 0$, the mapping $(w+, w-) \mapsto (f, c)$ is for practical purposes bijective, as statements with $w = 0$ don't need to be handled as they don't contribute any evidence.

Additionally, truth expectation is defined as

$$expectation(f, c) = (c * (f - \frac{1}{2}) + \frac{1}{2}).$$

This measure summarizes the two-valued truth value into a single value with the extremes being 0 for $c = 1, f = 0$, and 1 for $c = 1, f = 1$, which both are approachable but unreachable, since $\forall w \in \mathbb{R} : c < 1$ while $\lim_{w \mapsto \infty} c = 1$.

Implications For the sake of this paper we will restrict ourselves to temporal implications $(A \Rightarrow B)$ and procedural implications of the form $((A, op) \Rightarrow B)$. The former denotes that B will happen after A, and the latter that B will happen when op is executed right after A happened. To calculate the truth values of these implications, the evidences of w_+ and w_- are needed. If events would have binary truth, for $(A \Rightarrow B)$, w_+ would be the amount of cases in which A happened and B happened after it, and w_- would be the amount of cases where A happened but B did not happen thereafter. Slightly more complex but following the same idea, for $((A, op) \Rightarrow B)$, w_+ would be the amount of cases in which A happened, op was executed and B happened after it. And w_- would be the amount of cases where A happened and op was executed but B did not happen thereafter. Now, using w_+ and w_-, the truth value (f, c) of the implication statements would be fully determined. While this captures the main idea, to make the temporal reasoning more robust in regards to timing variations, the following treatment is used instead:

Event uncertainty Events are not "true" at only a specific moment in time, instead they have an occurrence time and truth value attached to them. Hereby, the confidence decreases with increasing time distance to the second premise (also called 'projection' in Wang (2013)). The way this is realized is that when two premises are used in inference, the confidence of the second premise is discounted by the factor $\beta^{|\Delta t|}$ with $\Delta t = time(B) - time(A)$, where β is the truth projection decay, a hyperparameter.

Now, the way implications are formed is via the Induction rule

$$\{A, B\} \vdash (A \Rightarrow B)$$

with Δt stored as metadata and the truth of the conclusion being (as described in more detail in Wang (2013)):

$$truth((A \Rightarrow B)) = f_{ind}((f_1, c_1), (f_2, c_2)) = (f_1, \frac{f_2 * c_2 * c_1}{f_2 * c_2 * c_1 + 1}).$$

Now, when the same implication is derived multiple times, their truth values are revised, by simply adding up the evidences of the premises: $w_+ = w_{+_1} + w_{+_2}$, $w_- = w_{-_1} + w_{-_2}$. This makes sure that the implication receives increasing amounts of evidence when the events which support it (the antecedent and consequent) do occur, exactly as we intended. But with the addition that evidence is discounted based on temporal distance, which is what makes the temporal credit assignment succeed. On this matter, projection plays the same role as eligibility traces do for reinforcement learners (Sutton, 1988).

As last detail, the Δt is also updated in revision, by taking a weighted average between the time deltas of the premises, weighted by the confidence of the premises. We will need this soon to decide the occurrence time of derived events.

Learning To form the temporal and procedural implications from input events (to calculate their evidence), a sliding-window approach is taken, where the sliding window (a first-in-first-out buffer) only holds the latest k events. This way evidence for implication $(A \Rightarrow B)$ is only attributed (based on the Induction rule we just described) when both the antecedent A and consequent B of the implication exist within the sliding window. Please

note that A can as well be a sequence here, like (X, Op), encoding that X happened and then operation event Op happened. In principle sequences don't need to contain operation events and can contain more than just two elements, this allows ONA to learn temporal patterns which span a larger time distance (up to the sliding window size). This helps especially in environments where the Markov property does not hold. But since we compare with Q-Learning which assumes the Markov property to hold (next state only being dependent on current state and current action), we will leave this out for now to make the comparison fair.

Collecting negative evidence for an implication is slightly more tricky ('anticipation' in NAL; see Wang (2013)), as it is supposed to be added when the consequent will not happen, but how long to wait for the consequent? Ideally this would not depend on the buffer size, and would be dependent on the averages of the experienced timings and related variances. However, timing assumptions can go wrong when certain distributional assumptions aren't met, which is why we went for a simpler solution for now which is at least not dependent on the size of the sliding window: to add a small amount of negative evidence immediately when the antecedent arrives, small enough that should be consequent arrive as predicted by the implication, the truth expectation of the implication will still increase (the positive evidence over-votes the negative), while else it would decrease due to the negative evidence which was added. Overall, the accumulation of positive and negative evidence leads to frequency values which encode the hypotheses (the implications) proficiency to predict successfully, whereby truth expectation can be seen as the expected frequency, which as we will now see is used in decision making (as it takes into account how many samples have been seen about a certain implication, eliminating initially lucky ones to be preferred over consistently competently predicting ones, and without having to assume a prior probability distribution).

Decision Making: Goal events $G!$ are represented as temporal implication $(G \Rightarrow D)$ where D is implicitly present and stands for "desired state", and their desire value is the truth value of this implication. When processed, goals either trigger decisions or lead to the derivations of subgoals. For this purpose, the existing procedural implications are checked. If the implication $((A, op) \Rightarrow B)$ has a sufficiently high truth value, and event A recently happened, it will generate a high desire value for the reasoner to execute op. The truth expectations of the implications with G as consequent are compared, and the operation from the candidate with the highest expectation desire value will be executed if above decision threshold (a hyperparameter). If not, all the preconditions (such as A) of the implications with G as consequent will be derived as subgoals, competing for attention and processing in a bounded priority queue ranked by the expectation of the desire value multiplied with the parent goal priority (this way only the most desired goals are pursued, whereby input goals have priority 1). Hereby, the desire value of the subgoal is evaluated using deduction between the implication and the goal Wang (2013). And to determine the operation's desire value one additional deduction step to take the precondition truth value into account is necessary. This corresponds to the inference rule

$$\{(X \Rightarrow G), (G \Rightarrow D)\} \vdash (X \Rightarrow D) = \{(X \Rightarrow G), G!\} \vdash X!$$

where the conclusion goal's occurrence time (the time at which X would have to have occurred if G had to happen right now) is G's occurrence time minus the Δt stored as

metadata of the implication. And the following inference rule in case X is of the form (Y, op):

$$\{((Y, Op) \Rightarrow D), Y\} \vdash (op \Rightarrow D) = \{(Y, Op)!, Y\} \vdash op!$$

which encodes that op is wanted to be executed if op is wanted to be executed after Y happened and Y happened.

The conclusion goal desire values are:

$$desire(X) = f_{ded}(desire(G), truth(((X, op) \Rightarrow G))$$

for the subgoal which corresponds to the antecedent of the implication, and

$$desire(op) = f_{ded}(desire((X, op)), truth(X))$$

for the operation subgoal to potentially execute if X happened, with f_{ded} being (as in Wang (2013)):

$$f_{ded}((f_1, c_1), (f_2, c_2)) = (f_1 * f_2, f_1 * f_2 * c_1 * c_2)$$

Using this model, decision making is concerned with realizing a goal by executing an operation which most likely, and sufficiently likely leads to its fulfillment under current circumstances. And when no such candidate exists to get this done in a single step, subgoals are derived from which a candidate will fulfill this requirement or again lead to further subgoaling, which is like backward planning from a goal to current circumstances, but while taking event uncertainties and uncertainties of the implications into account. This process can be summarized as follows:

Input: Goal G **Result:** Execution of Op, or subgoaling
subgoals = {}, bestDesire = 0.0
forall $((X, Op) \Rightarrow G) \in memory$ **do**
  $subgoals = subgoals \cup \{X\}$
  if $desire(Op) > bestDesire$ **then**
    bestDesire = $desire(Op)$, bestOp = Op
  end
end
if $bestDesire > DECISION\_THRESHOLD$ **then**
  execute(bestOp)
else
  forall $s \in subgoals$ **do**
    derive s (for potential selection in next inference step, leading to recursion)
  end
end

Algorithm 1: Decision and subgoaling

Also to make use of implications effectively in implementations, the procedural implications should be indexed by their consequent, where only a constant amount of implications is allowed for each consequent. This can be achieved by ranking them according to their truth expectation, so that the weakest implications are removed while these which predict

successfully are kept (similarly as in Hammer and Lofthouse (2018)), keeping the resource requirements bounded (Wang, 2009a). Also, through the indexing, the competing hypothesis to lead to the goal don't need to be searched for, they only need to be iterated and compared in the way the pseudo-code describes.

Exploration Additionally, sometimes the operation to execute is ignored and a random one is executed instead, which can be considered a form of exploration through motor babbling. This is also common for reinforcement learners, and for the reasoner is necessary especially in the beginnings where no procedural implication does exist thus far, hence no decision can be derived to lead to the desired outcome. Yet sometimes an action should be tried so that the first implications will form and "informed decision" can increasingly replace random trial (exploitation taking over exploration). A key difference is that, in accordance with the considerations regarding natural intelligence, there cannot be a time-dependent exploration rate decay, instead the reasoner reduces motor babbling when the truth expectation about how to reach a goal, $desire(Op)$, increases.

4. Reinforcement learning comparison with Q-Learning

In this section we will compare ONA and Q-Learning, from a theoretical perspective and on concrete examples both techniques can be applied on.

Differences between the decision-making models Before we move on to comparison on concrete experiments, there are some relevant differences in both decision making models which we will need to address to allow for a fair comparison. Since ONA is a NARS implementation design, many properties of the ones described in Wang and Hammer (2015) are inherited by it. Compared to reinforcement learning formalizations some of the most significant differences are:

- **Statements instead of states** ONA, as a NARS, does not assume states which fully describe the current situation, instead events are usually partial descriptions of the current situation as perceived by the agent, consistently with this idea also the Markov property is not assumed to hold. To make the practical comparison possible, the events however will hold the same information as the corresponding states the Q-Learner will receive in the simulated experiments. However, in the last example, a robotics use case, we will see events from different sources, coming from different modalities without blowing up the state space as would be the case when simply combining their values into a single state vector.

- **Unobservable information** Related to the previous point, there is a major difference between unobservable states which are not known (not just their values being unknown), and known unobservable states which values can be estimated from observable state due to known observation probabilities as commonly handled by a POMDP (Spaan, 2012) in a model-based RL setting. While ONA inherits some limited capability to address both via its uncertainty reasoning machinery, partial observability is outside of the scope of this manuscript.

- **Hierarchical abstraction** Complex environments often demand a higher level of abstraction of behaviors to allow for data-efficient learning of policies or hypotheses

in general. This is by far not fully solved yet by any AI model, though attempts like Nachum et al. (2018) and Zhou et al. (2019), which arrange multiple reinforcement learners in a hierarchical way, do exist. However also abstraction of state matters, here, deep learning, especially convolutional neural networks (Khan et al., 2020), allow to deal with high-dimensional image input and models which work in real-time (Shanahan and Dai, 2019) have become an essential technology in robotics and many real-world applications, including self-driving cars (Do et al., 2018; Nugraha et al., 2017; Farag, 2018; Zhang et al., 2019). In the last use case we will see YOLOv4 (Bochkovskiy et al., 2020) being utilized for object detection, additionally we will see further abstraction of behavior (abstracting away from particular object types) happening via inductive reasoning.

- **One action in each step** ONA does not assume that in every step an action has to be chosen. To make the techniques comparable, we will hence add an additional *ĥothing* action for the Q-Learner in each example.

- **Multiple objectives** ONA can work on multiple goals simultaneously (which will be demonstrated in a separate publication in the future). A common approach to deal with multi-objectives in reinforcement learning is to combine together the individual objectives into a single reward function (Sutton and Barto, 2018; Yang et al., 2019). This is most commonly done by formulating a scalarisation function (Van Moffaert et al., 2013), measuring the utility of a linear combination of expected return values of the objectives. In Zintgraf et al. (2015) however it is argued that the approach does not work if the parameters of the scalarisation function are not known in advance, and that in such cases a model that expresses the multiple objectives explicitly is required. The latter is common also in other Constructivist AI systems other than ONA, such as Leela (Kommrusch, 2020).

- **Changing objectives** Most reinforcement learning solutions are not designed to allow dealing with changing objectives / changing utility function. For game-playing this is fine, RL had a lot of success as the objective of a game usually does not change while playing it (Schrittwieser et al., 2020). However, in robotics scenarios the situation is different, behavior of robots (such as household robots and robotic explorers) is usually expected to satisfy dynamic user goals. In this case planning methods (route planning, motion planning, etc.) remain to be crucial in autonomous robotics and self-driving cars (Karpas and Magazzeni, 2020; Badue et al., 2021; Aradi, 2020), but can for example also be combined with RL-based path tracking approaches (You et al., 2019).

Reasoner and background The particular implementation we will use for comparison purposes and follows these principles is 'OpenNARS for Applications' (ONA, see Hammer and Lofthouse (2020)), an implementation of a Non-Axiomatic Reasoning System (Wang, 2013) developed by the author. While ONA has been compared with actor-critic (AC) and double-deep Q-learning (DDQ) on variants of the cart-pole task (Eberding et al., 2020), the input representations were not the same between the compared methods (mostly because ONA does not accept numeric inputs without preprocessing), lowering the strength of the

results. Additionally, more tasks are required to make the case to establish ONA as an additional technique to address reinforcement learning problems stronger. In this section we will compare ONA with a standard table-based Q-Learner (Watkins, 1989) implementation with eligibility taces (Sutton, 1988), while ensuring that both the Q-Learner and ONA receive the exact same input. For completeness, and to allow to relate the hyperparameter choices with the Q-Learning model used for the experiments:

$$Q(s_{t+1}, a_{t+1}) = max_a Q(s_{t+1}, a)$$

$$\delta Q_t = r + \gamma * Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

$$e_t(s_t, a_t) = e_t(s_t, a_t) + 1$$

$\forall s$ and a :

$$Q(s, a) \leftarrow Q(s, a) + \alpha * \delta Q_t * e_t(s, a)$$

$$e_t(s, a) \leftarrow \gamma \lambda e_t(s, a)$$

where α is the learning rate, γ controls how much to favour future rewards over short-term reward, and λ controls the decay speed of egilibility traces. Additionally ϵ which is not mentioned here, is the exploration rate. It encodes the chance to select a random action as a_{t+1} instead of the one with the highest expected reward.

Goal achievement as reward While ONA can deal with the goals in the described way, the Q-Learner needs a reward signal. Hence if both should receive the exact same input to make comparison more meaningful, there needs to be a mapping from goal achievement to reward. The way this is achieved is the following way: when an event X is input it is interpreted by ONA as event, and by the Q-Learner simply as current state. If X however corresponds to the outcome to achieve, and the reward for the Q-Learner will be 1 (while ONA receives event $goodNar$), and else 0. This of course assumes that the goal does not change, as else the Q-table entries would have to be re-learned, meaning the learned behavior would often not apply anymore. But for reinforcement learning problems it is most common that the objective is fixed, so for the purposes of this paper, and for a fair comparison, the examples will include a fixed objective.

Setup The experiments chosen for comparison are two typical reinforcement learning examples, Space invaders and Pong, plus a grid robot experiment where the agent has to find food while maneuvering around obstacles underway. Both techniques (Fig. 3) are run multiple times in each experiment, and the example-specific success measure is kept track of for each time point across 10000 iterations, together with the average summarized over all runs of the particular technique. Furthermore, the ONA parameters are chosen to be the ones in the default config of the ONA v0.8.8 master branch (Hammer and Lofthouse, 2020) across all experiments (with a FIFO window size of 20, increasing it did not increase the scores further). The motor babbling rate is chosen to be the same as ϵ of the Q-Learner in the experiments.

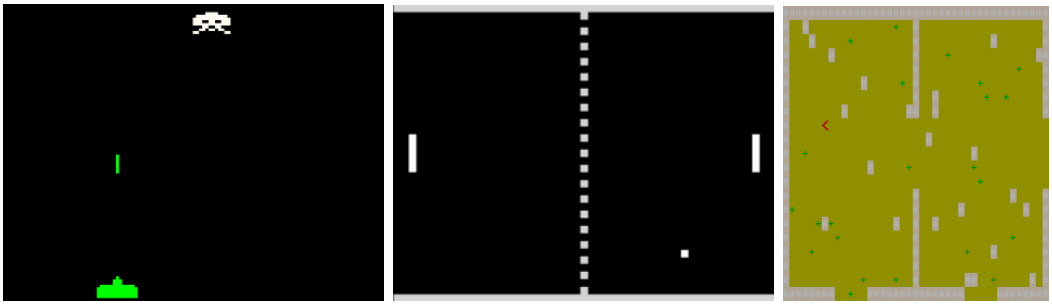

Figure 3: From left to right: Space Invaders, Pong and Robot.

Space Invaders In this game (Fig. 3) the player controls a spaceship and is supposed to shoot down aliens, whereby the success ratio we will use is the amount of hits over the total shots. Both models receive the enemy location as either $enemyLeft$, $enemyRight$, or $enemyAligned$ when aligned with the player. Additionally the actions the agent can take are $\hat{}left, \hat{}right$ which move the agent to the left/right side by some step size (set to be 5 percent of the game screen width, so 20 actions to get from one side to the other), and $shoot$. Additionally the $\hat{}nothing$ action exists for fair comparison between both techniques, as the reasoner can decide to do nothing according to NAL decision theory (Wang, 2013). To hit the enemy, the $shoot$ action needs to be taken when aligned with the enemy. The hyperparameters for the Q-Learner are $\alpha = 0.1, \gamma = 0.1, \lambda = 0.8, \epsilon = 0.3$. Each technique is run 10 times with different random seeds each, meaning also variations in starting positions in addition to outcomes of when to choose exploratory actions.

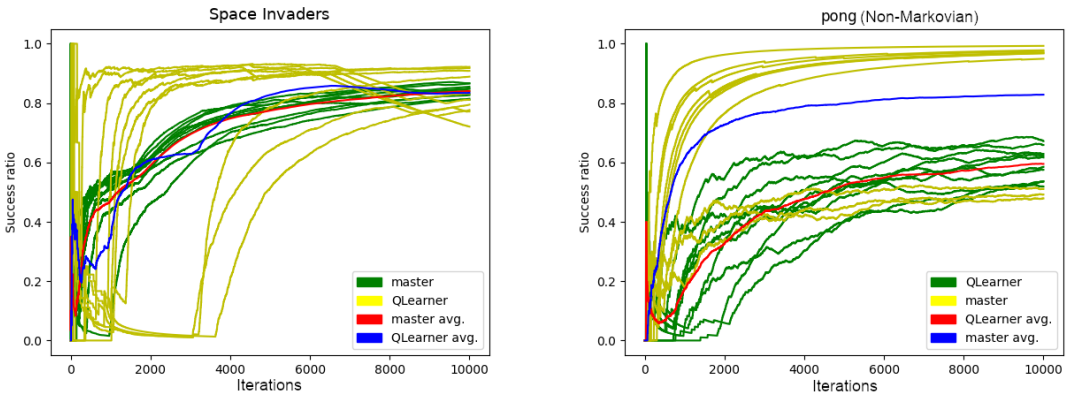

Figure 4: Success ratio in Space invaders and Pong.

The results as seen on the left side in Fig. 4 indicate that both techniques (Q-Learning and ONA) converge to the same capability on average, and on average with approximately the same learning speed. However, the learning behavior of ONA is more consistent, while Q-Learning revealed both cases where it learns quicker and slower than most ONA runs, corresponding to a higher variance in learning performance.

Pong In Pong (Fig. 3), the location of the ball, which is able to reflect at the walls, is encoded as relative $ballLeft$, $ballRight$, $ballEqual$ in the same fashion as in Space invaders. This time $\hat{}left$ and $\hat{}right$ initiate left/right movement of the bat, and an additional operator $\hat{}stop$ stops the movement. Like before $\hat{}nothing$ exists as alternative action to make the comparison fair. The goal is for the ball to hit the bat repeatedly, and the success rate represents the hits over the sum of hits and misses. This time the hyperparameters of the Q-Learner are $\alpha = 0.1, \gamma = 0.1, \lambda = 0.8, \epsilon = 0.2$, meaning a slightly lower epsilon than before.

Surprisingly, while the Q-Learner learned the use of $moveUp$ and $moveDown$, it consistently failed to learn to use the stop operator and hence reached lower success ratios than ONA (Fig. 4) which only failed to do so in 3 runs, while learning the more optimal non-Markovian behavior in the other cases.

When visualizing the reasoner's knowledge in a behavior graph as in Fig. 5, we can see why it can be tricky to learn the better policy. In this graph, nodes are the event, and links encode transitions via some operation (representing implication statements as introduced earlier). The operations are drawn at the outgoing side of the edge. Additionally, the edge colors are of interest here, where red encodes positive evidence and blue encodes negative evidence, hence violet is a mixture thereof:

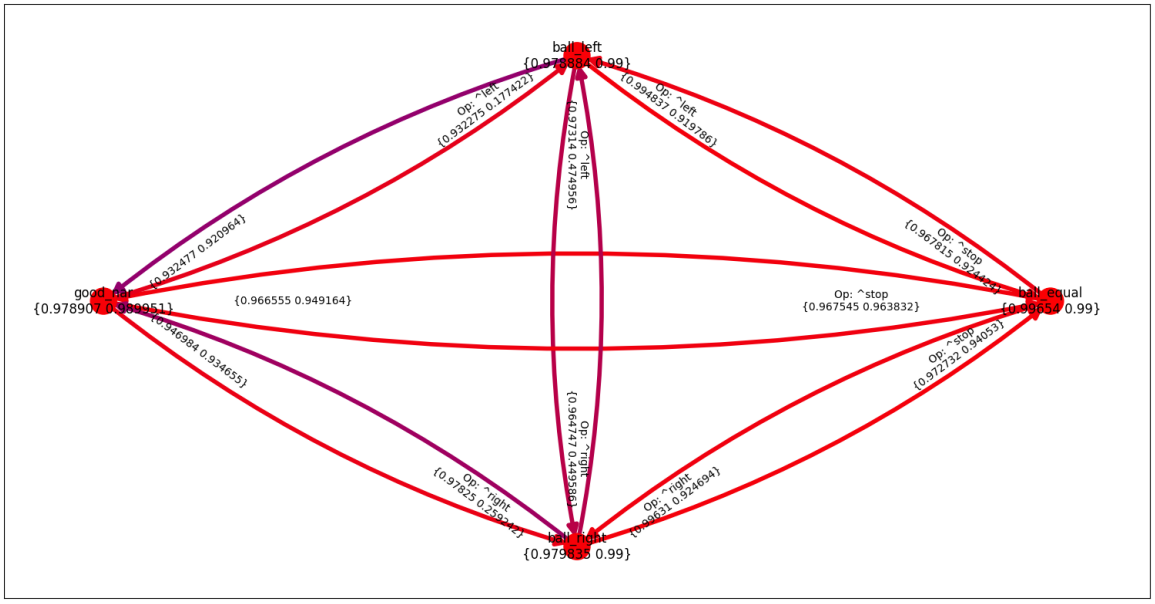

Figure 5: ONA behavior graph after learning Pong.

There is a violet rather than red connection from $ballLeft$ to $goodNar$ via $\hat{}left$, which indicates that this link has received an significant amount of negative evidence. This makes sense, as moving left when the ball is left is often not by alone sufficient to hit the ball, also the stop operator has to be invoked when reaching the ball location (same for the other direction). The edge from $ballLeft$ to $ballEqual$ via the $\hat{}left$ operation on the other hand is very successful (indicated by red color), and so is the edge from $ballEqual$ to $goodNar$ via $\hat{}stop$.

This makes the issue at heart clear: an operator that initiates movement instead of performing a step of a certain size once breaks the Markov property of the reward, that is, that the last state and action determines fully what state the agent will observe next and the reward it will obtain. With action $\hat{}left$ being used lastly, the bat continues to move left when $\hat{}nothing$ is invoked, hence the state transition and also reward in this case can depend on a state-action combination which happened prior to the last one. In this case also, state merging heuristics like in Gaon and Brafman (2020) cannot resolve the issue, as the side effect is caused by the action and not directly part of an observable state.

To confirm that this is indeed the explanation for the worse performance, the same experiment was tried without the $\hat{}stop$ action. Hereby, everything else (the overall setup) remained to be the same. In this case the Q-Learner turned out to perform comparably well like ONA on average in end performance (Fig. 6), while on average ONA learned faster:

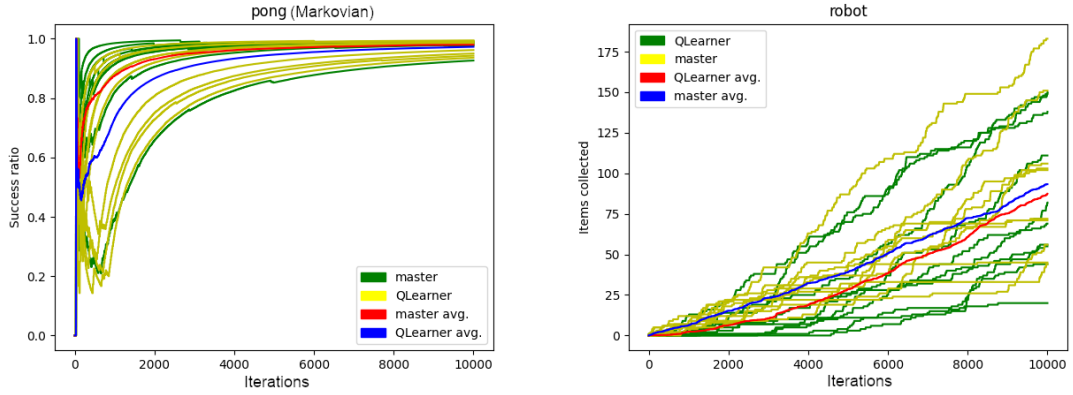

Figure 6: Success ratio in simplified Pong and Grid Robot.

though in this case the problem regresses to learning to invoke only $moveUp$ and $moveDown$, which is a simpler problem to solve. Nevertheless, what this suggests is that ONA can deliver the same learning performance as Q-Learning, while additionally delivering better results when the Markov property does not hold. The sliding window approach to mine for patterns in input event stream, although local in time, is less restricted than only considering the current and last state in Q-table updates.

Robot This experiment (Fig. 3) features a robot in a grid with walls and food objects to collect, whereby the amount of food objects collected we will directly use as success measure. Perceived events are from the perspective of the robot, which can turn around by using $\hat{}rotateLeft$ and $\hat{}rotateRight$. Additionally it can move forward by one grid cell by invoking $\hat{}moveForward$. The robot can see what is in front, left and right to it within a 10 cells distance. It can either be, in this preference order, $foodCentered$, $foodLeft$, $foodRight$, $wallCentered$, $wallLeft$, or $wallRight$. The only outcome positive of interest is the agent colliding with food, but of course to achieve this the agent also has to learn to avoid obstacles in order not to get stuck.

In this experiment, and with the same parameters as in Space Invaders, both techniques performed comparably well, with a slight leap of ONA in initial performance, and slight

leap of Q-Learning in the end performance. This example can easily be extended to multi-objective scenarios and scenarios with changing objectives, showing merits in these areas will be part of our future work.

Overall, as Table 1 suggests, except of the issue with the Markov property of states and rewards being violated for Q-Learning which demanded a simplification of the Pong example to get similar results, both techniques were comparable in performance on average. Hence the reasoning-based approach provides a viable alternative for such problems, while performing better whenever the Markov property is violated, since it does not explicitly depend on this property.

Table 1: End performance results of all experiments.

Success measure	**ONA**	**Q-Learner**	**Number of trials**
Space invaders	0.86	0.85	20 (10 each)
Pong (non-Markovian)	0.80	0.61	20 (10 each)
Pong (simplified)	0.98	0.97	20 (10 each)
Grid robot	91	87	20 (10 each)

5. Conclusion

We have presented an alternative to reinforcement learning, using Non-Axiomatic Logic for reasoning under uncertainty, and detailed how it relates to the phenomenon of natural intelligence. By allowing a reasoner to learn (and not just by an external mechanism like in inductive logic programming (Muggleton and De Raedt, 1994), which does not work in real-time), principles like goal derivation and planning based on goal-independent representations can also be used to tackle reinforcement learning problems. It has been shown that on the particular set of tasks (Pong, Space Invaders, and Grid Robot), 'OpenNARS for Applications' performs comparably well like Q-Learning with eligibility traces on average, and while providing better results when the Markov property is violated. This suggests the utilization of uncertainty reasoning (non-axiomatic reasoning in particular) to be an alternative, and sometimes better, approach to deal with various reinforcement learning problems. Additionally it has the potential to get around some of the inherent limitations in common reinforcement learning techniques, such as to exploit the Markov property of states and rewards, which as we saw can be easily violated even in relatively simple environments.

In the future we will show that the reasoning-based approach naturally allows for the flexibility means-end reasoning (practical reasoning) approaches are typically known and valued for (as we have started to show in our more recent robotics work (Hammer et al., 2023) and curriculum learning setups). This includes the ability to change behaviors immediately when goals change, to be able to plan to reach outcomes which have not been observed before in the same manner, to pursue multiple goals, and to take background knowledge into account effectively. Combining this with an ability to learn behaviors comparably well like reinforcement learners can allow to more easily build intelligent agents

which are expected to achieve various mission goals in an autonomous or semi-autonomous way.

References

Szilárd Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):740–759, 2020.

Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Rafael H Bordini and Jomi F Hübner. Bdi agent programming in agentspeak using jason. In *Computational Logic in Multi-Agent Systems: 6th International Workshop, CLIMA VI, London, UK, June 27-29, 2005, Revised Selected and Invited Papers 6*, pages 143–164. Springer, 2006.

Daniel A Cristol, Paul V Switzer, Kara L Johnson, and Leah S Walke. Crows do not use automobiles as nutcrackers: putting an anecdote to the test. *The Auk*, pages 296–298, 1997.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad, 2007.

Truong-Dong Do, Minh-Thien Duong, Quoc-Vu Dang, and My-Ha Le. Real-time self-driving car navigation using deep neural network. In *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, pages 7–12. IEEE, 2018.

Leonard M Eberding, Kristinn R Thórisson, Arash Sheikhlar, and Sindri P Andrason. Sage: task-environment platform for evaluating a broad range of ai learners. In *Artificial General Intelligence: 13th International Conference, AGI 2020, St. Petersburg, Russia, September 16–19, 2020, Proceedings 13*, pages 72–82. Springer, 2020.

Wael Farag. Recognition of traffic signs by convolutional neural nets for self-driving vehicles. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 22(3): 205–214, 2018.

Alexander Ferrein, Gerald Steinbauer, and Stavros Vassos. Action-based imperative programming with yagi. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Maor Gaon and Ronen Brafman. Reinforcement learning with non-markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3980–3987, 2020.

Olivier L Georgeon, Rémi C Casado, and Laetitia A Matignon. Modeling biological agents beyond the reinforcement-learning paradigm. *Procedia Computer Science*, 71:17–22, 2015.

Ben Goertzel, Matthew Iklé, Izabela Freire Goertzel, and Ari Heljakka. *Probabilistic logic networks: A comprehensive framework for uncertain inference*. Springer Science & Business Media, 2008.

Patrick Hammer and Tony Lofthouse. Goal-directed procedure learning. In *Artificial General Intelligence: 11th International Conference, AGI 2018, Prague, Czech Republic, August 22-25, 2018, Proceedings 11*, pages 77–86. Springer, 2018.

Patrick Hammer and Tony Lofthouse. 'opennars for applications': architecture and control. In *Artificial General Intelligence: 13th International Conference, AGI 2020, St. Petersburg, Russia, September 16–19, 2020, Proceedings 13*, pages 193–204. Springer, 2020.

Patrick Hammer, Peter Isaev, Tony Lofthouse, and Robert Johansson. Ona for autonomous ros-based robots. In Ben Goertzel, Matt Iklé, Alexey Potapov, and Denis Ponomaryov, editors, *Artificial General Intelligence*, pages 231–242, Cham, 2023. Springer International Publishing. ISBN 978-3-031-19907-3.

Pascal Hitzler, Aaron Eberhart, Monireh Ebrahimi, Md Kamruzzaman Sarker, and Lu Zhou. Neuro-symbolic approaches in artificial intelligence. *National Science Review*, 9(6):nwac035, 2022.

Hanqing Hu, Mehmed Kantardzic, and Tegjyot S Sethi. No free lunch theorem for concept drift detection in streaming data classification: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1327, 2020.

Can Kabadayi and Mathias Osvath. Ravens parallel great apes in flexible planning for tool-use and bartering. *Science*, 357(6347):202–204, 2017.

Erez Karpas and Daniele Magazzeni. Automated planning for robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:417–439, 2020.

Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53:5455–5516, 2020.

Steve Kommrusch. Self-supervised learning for multi-goal grid world: Comparing leela and deep q network. In *International Workshop on Self-Supervised Learning*, pages 72–88. PMLR, 2020.

Olli J Loukola, Cwyn Solvi, Louie Coscos, and Lars Chittka. Bumblebees show cognitive flexibility by improving on an observed complex behavior. *Science*, 355(6327):833–836, 2017.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Brilian Tafjira Nugraha, Shun-Feng Su, et al. Towards self-driving car using convolutional neural network and road lane detector. In *2017 2nd international conference on automation, cognitive science, optics, micro electro-mechanical system, and information technology (ICACOMIT)*, pages 65–69. IEEE, 2017.

Judea Pearl. Causal inference. *Causality: objectives and assessment*, pages 39–58, 2010.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62: 107–136, 2006.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

James G Shanahan and Liang Dai. Realtime object detection via deep learning-based pipelines. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2977–2978, 2019.

Matthijs TJ Spaan. Partially observable markov decision processes. *Reinforcement learning: State-of-the-art*, pages 387–414, 2012.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Kristinn R Thórisson, Jordi Bieger, Xiang Li, and Pei Wang. Cumulative learning. In *Artificial General Intelligence: 12th International Conference, AGI 2019, Shenzhen, China, August 6–9, 2019, Proceedings 12*, pages 198–208. Springer, 2019.

Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–199. IEEE, 2013.

Pei Wang. Insufficient knowledge and resources—a biological constraint and its functional implications. In *2009 AAAI Fall Symposium Series*, 2009a.

Pei Wang. Formalization of evidence: A comparative study. *Journal of Artificial General Intelligence*, 1(1):25–53, 2009b.

Pei Wang. *Non-axiomatic logic: A model of intelligent reasoning*. World Scientific, 2013.

Pei Wang and Patrick Hammer. Assumptions of decision-making models in agi. In *Artificial General Intelligence: 8th International Conference, AGI 2015, AGI 2015, Berlin, Germany, July 22-25, 2015, Proceedings 8*, pages 197–207. Springer, 2015.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32, 2019.

Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Autonomous planning and control for intelligent vehicles in traffic. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2339–2349, 2019.

Lotfi A Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, 1988.

Hanbo Zhang, Xuguang Lan, Site Bai, Lipeng Wan, Chenjie Yang, and Nanning Zheng. A multi-task convolutional neural network for autonomous robotic grasping in object stacking scenes. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6435–6442. IEEE, 2019.

Ye Zhou, Erik-Jan van Kampen, and Qiping Chu. Hybrid hierarchical reinforcement learning for online guidance and navigation with partial observability. *Neurocomputing*, 331: 443–457, 2019.

Luisa M Zintgraf, Timon V Kanters, Diederik M Roijers, Frans Oliehoek, and Philipp Beau. Quality assessment of morl algorithms: A utility-based approach. In *Benelearn 2015: proceedings of the 24th annual machine learning conference of Belgium and the Netherlands*, 2015.