Towards Symbolic Reinforcement Learning with Common Sense

Artur d'Avila Garcez[1] **Aimore Resende Riquetti Dutra[2]** **Eduardo Alonso[3]**

Department of Computer Science
City, University of London
London EC1V 0HB, U.K.
a.garcez@city.ac.uk[1], aimorerrd@hotmail.com[2], e.alonso@city.ac.uk[3]

Abstract

Deep Reinforcement Learning (deep RL) has made several breakthroughs in recent years in applications ranging from complex control tasks in unmanned vehicles to game playing. Despite their success, deep RL still lacks several important capacities of human intelligence, such as transfer learning, abstraction and interpretability. Deep Symbolic Reinforcement Learning (DSRL) seeks to incorporate such capacities to deep Q-networks (DQN) by learning a relevant symbolic representation prior to using Q-learning. In this paper, we propose a novel extension of DSRL, which we call Symbolic Reinforcement Learning with Common Sense (SRL+CS), offering a better balance between generalization and specialization, inspired by principles of common sense when assigning rewards and aggregating Q-values. Experiments reported in this paper show that SRL+CS learns consistently faster than Q-learning and DSRL, achieving also a higher accuracy. In the hardest case, where agents were trained in a deterministic environment and tested in a random environment, SRL+CS achieves nearly 100% average accuracy compared to DSRL's 70% and DQN's 50% accuracy. To the best of our knowledge, this is the first case of near perfect zero-shot transfer learning using Reinforcement Learning.

1 Introduction

The combination of classical Reinforcement Learning (RL) with deep neural networks has achieved human-level competence at solving some difficult problems, especially with the use of Deep Q-Networks (DQN) at solving games[15, 17, 20]. There is no doubt that Deep Reinforcement Learning (DRL) has offered new perspectives to the areas of automation and Artificial Intelligence (AI), but despite their success, DRL seems unable to tackle a number of problems that are considered relatively simple for humans to solve. DRL requires large training data sets, thus learning slowly. A trained deep network performing very well in one task will often perform poorly in another, analogous (sometimes very similar) task. Finally, DRL lacks interpretability, being criticized frequently for being a "black-box" model.

Some authors have tried to address the above shortcomings by adding prior knowledge to neural networks in general, and more recently to RL [4, 8, 11, 24]. In [5], it is argued that combining aspects of symbolic AI with neural networks and Reinforcement Learning could solve at once all of the above shortcomings. Such ideas are then instantiated in a computational system called Deep Symbolic Reinforcement Learning (DSRL). DSRL uses convolutional neural networks which, applied to navigation tasks typically consisting of negotiating obstacles (objects) and finding optimal paths or collecting other objects, can learn relevant object types from images, thus building a more abstract symbolic representation of the state-space, which is followed by a Q-learning algorithm for learning a policy on this state-space. Despite the fact that DSRL does not learn as effectively as DQN in a deterministic environment, DSRL was shown to outperform DQN when the policy learned in a deterministic environment is transferred to a random environment, thus indicating the value of using a more abstract, symbolic representation.

In this paper, we seek to reproduce the DSRL system with some simplifications. We have implemented this simplified, but also more generic system, which we refer to as Symbolic Reinforcement Learning (SRL). For the sake of analysis, it is desirable to separate within SRL, learning and decision-making from the object recognition part of the system, this latter part carried out in DSRL by convolutional neural networks. Such a separation allows us to focus the analysis of performance on the learning and decision making by providing the symbols and positions of the objects directly to the SRL system. Aspects of type transition and symbol interaction dealt with by DSRL are not analyzed in this paper.

We also propose an extension of SRL called Symbolic Reinforcement Learning with Common Sense (SRL+CS). SRL+CS makes one modification to the learning of SRL and one modification to the decision-making. This is then shown empirically to be sufficient for producing a considerable improvement in performance by comparison with various other algorithms. Differently from SRL, in SRL+CS only the states containing objects with which an agent interacts have their Q-values updated when rewards are non-zero (let us call this *principle one*), and the relative position of an object with respect to an agent is considered in the decision of which action to take (call this *principle two*).

More specifically, *principle one* is implemented through Equation (3) by making specific updates to the Q-values according to the specific interactions that an agent has with the (objects in the) environment; *principle two*, which seeks to give more importance to the objects that are nearer to the agent, is implemented through Equation (4) by simply dividing the Q-values by a factor proportional to the relative position of the agent w.r.t. the objects for the purpose of choosing an action. The implementation of these two principles within a standard RL set-up such as SRL+CS is only made possible by the adoption of a more abstract symbolic representation of the state-space, as put forward by the DSRL approach.

For the sake of comparison between the various RL systems already mentioned, we chose to reproduce a benchmark game used by DSRL in [5]. In this game, SRL+CS achieves 100% accuracy when tested in a random environment, compared to DQN's 50% and DSRL's 70% accuracy. In general, SRL+CS learns faster and more effectively than SRL by simply incorporating the above principles one and two to Q-learning.

The remainder of the paper is organized as follows: In Section 2, the Q-Learning, DQN and DSRL algorithms are reviewed briefly. In Section 3, the Symbolic Reinforcement Learning (SRL) algorithm is explained in three parts: state-space representation, learning, and decision-making. In Section 4, the Symbolic Reinforcement Learning with Common Sense algorithm is introduced through simple modifications to the SRL algorithm's learning and decision making. The setup of the experimental environment is introduced in Section 5 using several variants of the benchmark game where an agent is expected to collect an object type while avoiding another. Experimental results are also presented in Section 5. We evaluate how Q-learning, DQN, DSRL, SRL and SRL+CS generalize and scale with changes to the size of the environment, as well as how the systems perform at zero-shot transfer learning when trained and tested in deterministic or random setups. SRL+CS outperforms all the other systems in all but one of these experiments. We then conclude and discuss directions for future work.

2 Background

In this section, we briefly recall the algorithms of Q-learning, Deep Q-Networks (DQN) and Deep Symbolic Reinforcement Learning (DSRL).

2.1 Q-Learning

Q-learning is a model-free RL algorithm (although model-based variations have been proposed). In Q-learning, an agent's goal is to maximize by trial and error its total future reward thus learning which action is optimal in a given state [26]. By learning a function $Q(s, a)$ mapping states to actions, Q-learning can under certain conditions find an optimal action-selection policy for any given finite Markov Decision Process (MDP). $Q(s, a)$ denotes the quality of a state-action pair, which ultimately gives the expected utility of an action a in a given state s. The Q-learning algorithm can be summarized as follows (below, an ϵ-greedy policy is the most commonly used policy, which with probability ϵ, selects an action randomly, and with probability $1-\epsilon$, selects an action that gives the maximum reward in the current state):

```
Starting at state:  $s_0$,
For                 $t = 1, 2, 3, \ldots$
                    Choose an action $a_t$ using e.g. an $\epsilon$-greedy policy w.r.t. $Q(s,a)$;
                    Execute action $a_t$;
                    Update Q, as follows:
                    $Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left[R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$,
                    where $\alpha_t$ is the learning rate at time $t$,
                    $R$ is the reward observed for the current state and choice of action,
                    $\gamma$ is a temporal discount factor, and
                    $\max_a Q(s_{t+1}, a)$ is an estimate of the optimal future value.
```

At each step, the Q-Learning algorithm updates the Q-value function considering the current state (s_t) and the action performed (a_t). After learning, the policy Π dictating the action to be taken at any given state will be given by: $\Pi(s) = arg \max_a Q(s,a)$. A full introduction to Q-learning can be found in [23].

2.2 Deep Q-Networks (DQNs)

Deep Q-Networks (DQNs) were introduced in [17] and ignited the field of deep Reinforcement Learning for three main reasons: first, DQNs were designed as an end-to-end RL approach when applied to games having only the pixel values and the game score as inputs, thus requiring minimal domain knowledge; second, it used deep convolutional neural networks (CNN) and experience replay to learn a Q-value approximation function successfully; third, it showed very good performance in a wide range of games - in some of them achieving performance higher than human-level - with the use of the same hyper-parameters [13]. In a nutshell, DQNs are composed of a convolutional neural network to reduce the state space and seek to generalize states, and a Q-learning algorithm for mapping states to actions. Despite their success, DQN training can be computationally inefficient, lacking explainability and hierarchical reasoning.

2.3 Deep Symbolic Reinforcement Learning (DSRL)

DSRL was introduced in [5]. In DSRL, first, object types are classified with objects having their locations detected by a low-level symbol generator which uses a convolutional auto-encoder and a spectrum comparison technique. Second, a spatio-temporal representation is built of all the objects, using the relative positions of the objects at two consecutive time frames. Third, this spatio-temporal representation is provided to a group of Q-learning functions, one for each pair of type of objects. The update rule for the interaction between objects of types i and j is given by:

$$Q^{ij}(s_t^{ij}, a_t) \leftarrow Q^{ij}(s_t^{ij}, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q^{ij}(s_{t+1}^{ij}, a) - Q^{ij}(s_t^{ij}, a_t)\right] \tag{1}$$

where α is the learning rate, γ is a temporal discount factor, and each state s_t^{ij} represents an interaction between object types i and j at time t. After learning, the values of the Q^{ij} functions are added, and the action with the largest sum is chosen as shown in Equation 2.

$$a_{t+1} = argmax_a \sum_Q (Q(s_{t+1}, a)) \tag{2}$$

3 Symbolic Reinforcement Learning (SRL)

SRL is our implementation of the DSRL model with some simplifications. For ease of analysis, in SRL we separate the Reinforcement Learning problem into three parts: State-Space Representation, Learning, and Decision-Making, as detailed in what follows.

3.1 State-Space Representation

Naive approach: The simplest way of representing the state space of an RL environment derived from an image is to associate each possible set of pixel values with a state. A black and white image is composed of a matrix of pixels that can assume values from 0 to 255. In a black and white image where n is the number of rows and m is the number of columns of pixels in the image, the number of possible states will then be 256^{n*m}. Such a very large state space presents obvious difficulties for learning. In this setting, each combination of assignments of values to the pixels in the image can be considered a relevant state for the purpose of RL.

CNN approach: In order to reduce the number of states in an image, the use of a convolutional neural network (CNN) was proposed in [16], leading to Deep Q-networks. DQNs seek to generalize the states provided by pixel values using learned filters and a fully-connected feedforward neural network as a regressor for the Q-function value approximation. However, the states generalized by the CNN are by construction dependent on all the pixels in the raw image, which can hinder generalization across different images. In [21], it is shown for example, that varying one pixel value can cause the classification of a CNN to change, or in the case of DQN, the state to change. Furthermore, the states are not fully translation-, rotation-, and size-invariant, requiring additional manipulation to become more robust, as shown in [7]. States generalized by CNNs do not carry explicit information about the objects' relative positions, and are in general restricted by the fixed dimensions of the input image. Experiments reported in [5] and reproduced here will highlight this problem in the context of Q-learning.

Symbolic approach: Another way to compress and generalize the number of states in an image that does not incur the above loss of information is to recognize such objects (possibly using a CNN) and represent them by symbols with their respective locations. In the DSRL approach of [5] this is done by using a low-level symbol generation technique, denoted by the letter C in Figure 1, which illustrates the process. An advantage of DSRL, therefore, is that now the state-space can be represented not by the combination of all pixel values or learned filters but by object types (e.g. '+' and '-' as in Figure 1) and their positions (X and Y numerical values in Figure 1), represented in symbolic form.

The use of a symbolic approach allows one to separate the detection and classification of objects in an image, which have been shown to work well with the use of CNNs and other deep learning and hybrid approaches [1, 5, 6, 12, 19], from the ways that an agent may learn and reason about its environment. In our current implementation of SRL, used to obtain the results reported in this paper, the object detection and classification part is not included. Instead, we focus on the agent's learning and reasoning. Each object's type and position is provided directly to the system in symbolic form as shown on the right-hand side of Figure 2, where we use the letter A to denote *abstraction*. Each box in Figure 2 can be seen as a sub-state, which can in turn be composed with other sub-states in different ways depending on the choice of abstraction.

Sub-states that are represented only by a single object often do not carry enough relevant information when it comes to reasoning about the properties of the problem at hand such as size invariance. In the experiments carried out in this paper, which use the same game as used in [5] where a single agent represented by a star is expected to collect all the positive objects in the environment while avoiding any negative objects, the four objects shown in Figure 1 are combined to create the six abstract sub-states shown on the right hand side of Figure 2.

We have observed empirically that the choice of abstraction can facilitate learning considerably by representing the relevant parts of the environment in which an agent is expected to focus. Although it is likely that the *best* abstraction will depend on the problem at hand, and therefore on background knowledge about the problem domain, in what follows we identify some generally desirable (common-sense) abstractions, such as the relative distance between objects, which allows learning to scale well to larger environments, and which can be implemented still by a model-free variation of Q-learning.

Both in DSRL and SRL, abstraction is carried out by the combination two by two of sub-states. In DSRL, this is done as shown in Figure 2 by taking the relative position of each object w.r.t. every other object. In SRL, the same is done but then, in addition, any state where an agent (the star-shaped object) is not included is removed (states $s'4$ to $s'6$ in the example of Figure 2). This is simply because any state without an agent is obviously irrelevant to our game. In general, ideally, the system would know how to select only sub-states which are relevant to solving the problem.

An alternative choice of abstraction would essentially dictate how general or specific the resulting state-space should be in relation to the learning task at hand. We leave such an analysis of generality vs. specificity as future work. Notice that the outcome of abstraction in SRL or DSRL is different from the outcome of a neural network-based approach such as DQN. While a CNN is capable of distinguishing one state from another, (symbolic) abstraction is capable of creating entirely new states from the original input. It is likely however that such abstraction can be achieved by stacking end-to-end differentiable neural networks, although this too is left as future work.

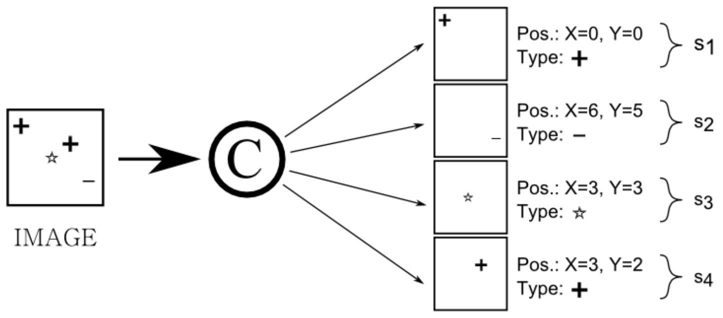


Figure 1: The letter C represents the tasks of object recognition and object type assignment typically carried out with the use of a convolutional neural network [5]. In our approach, such object types and their location are seen as sub-states of the state of an image.

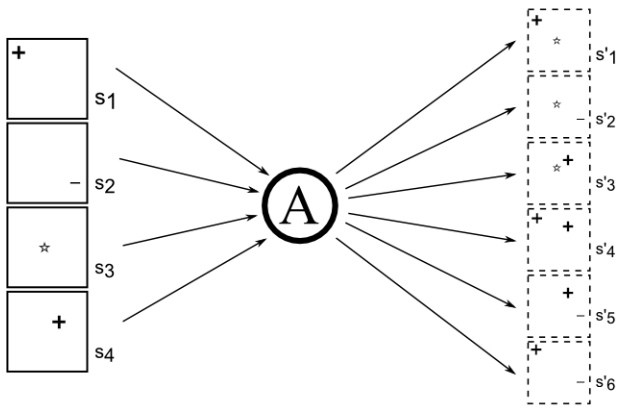


Figure 2: Each sub-state can be combined with certain other sub-states to form a richer, and hopefully more meaningful symbolic representation for the learning task at hand. We call this process an abstraction, here denoted by the letter **A**. In this particular example, **A** simply calculates the relative position of each object w.r.t. every other object. The boxes drawn using dotted lines indicate that the positions are relative, while the boxes drawn using solid lines indicate that the positions are absolute.

In summary, several advantages can be found in the symbolic way of representing the world:

(a) It simplifies the problem by reducing the state-space and limiting the number of relevant objects in each sub-state;
(b) It is compositional, allowing the state-space to be represented in different ways which may facilitate learning;
(c) Sub-states can reappear at different regions of an image thus contributing to better generalization;
(d) It is independent of the size of the images; once a task is learned it should scale to larger environments;
(e) It should be possible at least in principle to learn relations among objects.

3.2 Learning

Learning with multiple sub-states: A potential drawback of having sub-states is that, instead of solving one Markov Decision Process (MDP), our learning algorithm now has to solve N Partially Observable Markov Decision Processes (POMDPs), where N is the number of abstract sub-states (s'). In DSRL and SRL, Q-Learning was chosen for this task, in which case there are two options:

use a single Q-value function which embeds the type and position of objects in its sub-states (s'), or use one Q-value function for each pair of object types, embedding only their position in the sub-states. The alternatives are illustrated in Figure 3. In DSRL and SRL, the second option was chosen, since having separate Q-value functions should make learning from each specific combination of object types easier.

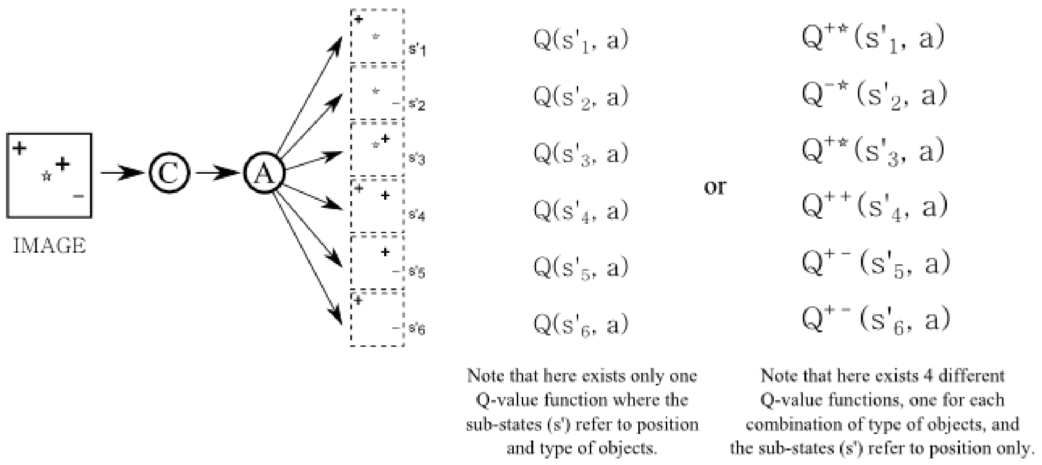


IMAGE
C
A
s'1
s'2
s'3
s'4
s'5
s'6
Q(s'1, a)
Q(s'2, a)
Q(s'3, a)
Q(s'4, a)
Q(s'5, a)
Q(s'6, a)
or
Q+*(s'1, a)
Q-*(s'2, a)
Q+*(s'3, a)
Q++(s'4, a)
Q+-(s'5, a)
Q+-(s'6, a)
Note that here exists only one Q-value function where the sub-states (s') refer to position and type of objects.
Note that here exists 4 different Q-value functions, one for each combination of type of objects, and the sub-states (s') refer to position only.

Figure 3: Two options for encoding abstract sub-states into a Q-value function: use a single Q-value function or use as many as the number of pairs of object types. By choosing the latter, another choice will be required on how to use the Q-value functions for decision making.

In summary, when abstractions (s') are used in the place of entire images, an agent is required to learn in a partially observable environment. In such cases, no guarantees exist of finding an optimal policy, as in the case of a fully-observable environment. Nevertheless, it should still be posible for the learning algorithm to find optimal policies for sub-tasks which may be described fully within the sub-spaces. These sub-tasks often have shorter horizons and may appear more than once, making learning in such cases converge faster, and hopefully helping an agent to succeed in its main task.[1]

Rewarding multiple sub-states: In Q-learning, a reward is obtained when an action takes place in a given state. An alternative is to associate rewards with the resulting state that an action may produce, or even with reaching a state independently of the action. In SRL with the use of multiple Q-learning functions, therefore, most of the time that a reward is obtained it will be impossible to know to which sub-states this reward corresponds. An agent will perform poorly if rewards are associated with sub-states that are not relevant, or are not the *cause* of the reward.[2]

In SRL, the same rewards go into the updates of all the Q-value functions. As discussed, this is undesirable because certain sub-states may be rewarded for an unrelated action. We have confirmed this empirically by creating an example where rewards are assigned incorrectly, leading to low performance at test time. SRL+CS will address this problem by limiting reward updates to specific sub-states based on valid object interactions, as detailed in the next section.

3.3 Decision Making

Another issue that arises when considering multiple sub-spaces is how to aggregate the Q-value functions learned in order to make a decision, i.e. choose a single action at a given state.

[1]An example would be playing the Atari game of Breakout by focusing first only on how the paddle moves left or right as a result of the corresponding key strokes, then focusing on certain objects only, such as the paddle and the ball, and how to hit the ball every time, and so on. At some point, e.g. to end the game with a win, an agent could choose to combine all sub-states in its abstraction to form the full image and solve the problem as an MDP. That would require changing the focus from paddle and ball only to the entire state space and probably considering longer horizons, although at this point fewer objects will have been left, and therefore the state space will be smaller than at the start of the game.

[2]For example, in the game of Chess it may be counter-productive to reward for wining the game the positions of the pieces that do not take any part in the King's checkmate.

DSRL selects the highest value among the sum of each Q-value function, as shown in Equation 2. In SRL, we use the same approach. However, this may not work in all situations because the actions associated with the Q-value functions may be contradictory, especially as the number of Q-value functions (i.e. objects) increases. Simply adding the Q-values seems insufficient to guarantee that *argmax* will maximize the accumulated future reward.

Another way to aggregate the Q-value functions would be to simply take the *argmax* among all the Q-value function values. This does not work well either because it ignores Q-value functions with lower values, which may be relevant nevertheless (e.g. for learning from multiple sub-states). SRL+CS will address this problem by assigning a degree of importance to the different Q-functions, as detailed in the next section.

4 Symbolic Reinforcement Learning with Common Sense (SRL+CS)

Symbolic Reinforcement Learning with Common Sense is based on SRL with two modifications, one in the learning and another in the decision-making part, with the objective of solving the two issues identified in the previous section: assigning rewards and aggregating Q-values adequately. In what follows, first, we formalize the state-space representation, then we present the two modifications made in the learning and decision-making algorithms.

STATE-SPACE REPRESENTATION:

SRL+CS creates a state-space exactly as done by SRL. Formally, it builds a representation of the world (abstraction) by creating sub-states s^k which consist of the relative position of an agent w.r.t. an object. Each sub-state represented by this pair (agent, object) is denoted by k. Let agent m and object n have absolute positions (x^m, y^m) and (x^n, y^n), respectively. Then, s^k is a tuple $(x^m - x^n, y^m - y^n)$.

LEARNING:

In order to assign rewards adequately, SRL+CS restricts the Q-value function updates to specific sub-states, based on the interactions between an agent and the objects. Specifically, when a reward that is non-zero is received, only the Q-value function with a sub-state s^k with a value of (0,0) is updated. Recall that a value of (0,0) indicates that an agent is in direct contact with the object. Equation 3 shows the Q-value function update:

$$Q^{ij}(s_t^k, a_t) \leftarrow Q^{ij}(s_t^k, a_t) + \alpha \left[r_{t+1} + \gamma \max_A Q^{ij}(s_{t+1}^k, A) - Q^{ij}(s_t^k, a_t) \right] \tag{3}$$

where α is the learning rate, γ is the temporal discount factor, and A is the set of all possible actions for that state s_{t+1}^k. Notice that more than one update to the Q-value function with indices i and j can take place (i.e. for different sub-states) at each time point t due to the presence of multiple objects of the same type in the image. Equation 3 is very similar to the standard Q-learning approach, with the difference that now usually the update does not occur once, but for all sub-states (k). When the reward is different from zero, an update occurs only for the sub-state with value (0,0). Intuitively, this is expected to indicate *where a reward has been coming from*, given a set-up consisting of multiple sub-states.

DECISION MAKING:

In order to aggregate Q-values adequately, SRL+CS assigns a degree of importance to each Q-value function, based on how far the objects are from an agent. The calculation of the distances from an agent to the objects is straightforward given the above choice of symbolic state-space representation. This allows one to give a priority to the decisions that are spatially close and therefore generally more relevant. It should also help an agent concentrate on a relevant subtask; in the case of the game used in the experiments that will follow, a relevant sub-task may be to collect the positive objects which are above or below. This idea is in line with that of a common sense *core* whereby our actions are influenced mostly by our surroundings [10], giving more importance in general to the objects that are nearer or can be seen.

Formally, the next action is chosen by Equation (4), where d_t^k is the Euclidean distance between the agent and the object's position at time t.

$$a_{t+1} = arg\max_{A}\left[\sum^{k}\frac{Q^{ij}(s_t^k, A)}{(d_t^k)^2}\right] \tag{4}$$

Notice the similarity with Equation 2. As in SRL, Q-values are summed and the action associated with the function with the largest value is taken. It is worth pointing out that, as before, this is not guaranteed to work in all situations, although it works very well in most situations of the game used in our experiments, reported in the next section.

5 Experimental Results

We have implemented a game, also used in [5], to evaluate and compare the performances of the following RL algorithms: Q-Learning, DQN, DSRL, SRL and SRL+CS, as introduced and discussed in the previous sections. In this game, an agent (a star-shaped object) can move up, down, left or right to collect as many positive objects as possible, denoted by a plus sign, while avoiding as many negative objects as possible, denoted by a minus sign. The objects are at fixed positions. Every time that the position of the star-shaped agent coincides with that of an object the object disappears, yielding a reward of 1 in the case of positive objects and -1 in the case of negative objects. A set of bricks restricts the passage of the agent, as shown in Figure 4. The game finishes when all positive objects are collected or after 100 movements of the agent.

The environment is fully-observable (the agent can see all the objects), sequential, static, discrete and finite, with no model of the world provided in advance.

All algorithms (Q-Learning, DQN, DSRL, SRL and SRL+CS) used an ϵ-greedy exploration strategy with a 10% chance of choosing a different action at random during training. In addition, actions were chosen randomly when two or more actions shared the same Q-value, i.e. no predefined preference was assigned to any action. The learning rate (alpha) was set to 1.0 since the state transitions were deterministic, and the discount-factor rate was set to 0.9 as a standard value.

The metric chosen for measuring the performance of the star-shaped agent was a score calculated as the sum of accumulated rewards. This tells us how much an agent has learned to move correctly by collecting positive objects while avoiding negative ones.

5.1 Training in Larger State-spaces

Experiment 1 - Increasing the Size of the Environment.

In this first experiment, agents were trained on the three grids shown in Figure 4 of increasing sizes. Starting from the center of the grid, an agent has to collect a positive object placed at a random location within the grid. No negative objects were used in this experiment.

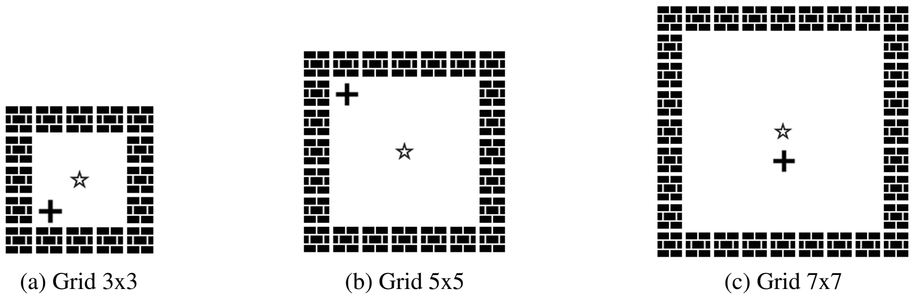
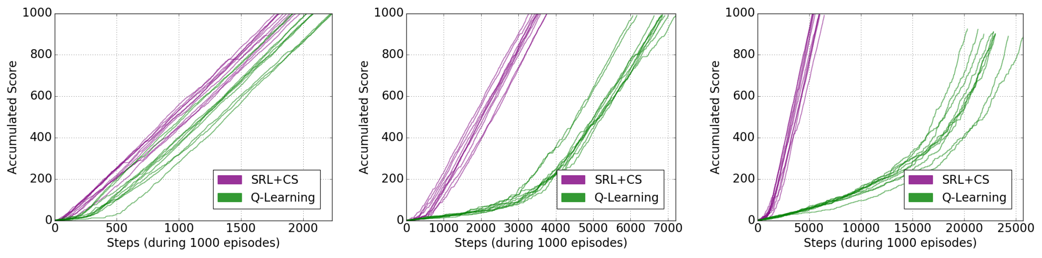

(a) Grid 3x3 (b) Grid 5x5 (c) Grid 7x7

Figure 4: Environments of increasing sizes used in Experiment 1 with positive objects only.

In SRL (and DSRL and SRL+CS), an abstraction is responsible for building a state space that takes into consideration the relative positions of the objects. As a result, an increase in the size of the environment should not imply the same increase in the size of the state-space. If the relative position between agent and object is the same in both the small and large grids, the agent's performance should be roughly the same regardless of the size of the environment. This is not normally the case with the use of Q-learning.

Figure 5 shows, in each plot, the accumulated scores obtained over 10 runs of 1000 games each, using both SRL+CS and Q-learning. It can be seen that, as the size of the grid increases, Q-learning requires many more training steps (i.e. movements of the agent within the grid) than SRL+CS. The results confirm those reported in [5].

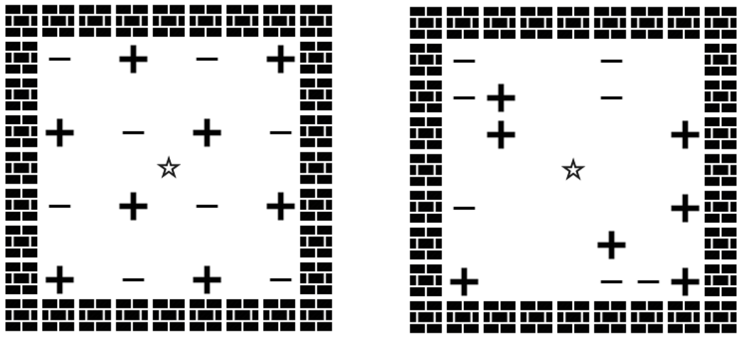


(a) Results of grid 3x3 (b) Results of grid 5x5 (c) Results of grid 7x7

Figure 5: Experiment 1 - These three plots, each containing 10 runs of 1000 games (each game is also referred to as an *episode*; each *step* is one movement of the agent in the grid) show that the symbolic approach (in this case SRL+CS) learns to collect the positive objects much faster than Q-learning as the size of the grid increases; SRL obtained very similar results to SRL+CS, but these were not plotted to avoid occlusion.

5.2 Training and Testing in Deterministic and Random Configurations

We now run three experiments using the two configurations shown in Figure 6 where the environment is initialized at each episode either deterministically or at random with the same number of both positive and negative objects. These configurations are the same as used in the experiments reported in [5].



(a) Deterministic Configuration (b) Random Configuration

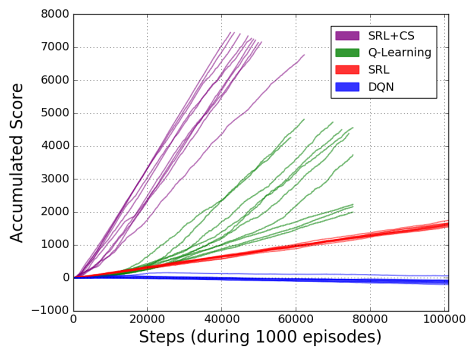
Figure 6: Two examples of initial configurations of the environment used in Experiments 2, 3 and 4, where each episode starts with eight (a) and seven (b) objects of each type.

In addition to the *accumulated scores* metric already used to measure agent performance, here we also use the percentage of positive objects collected over the total number of objects collected: a value of 100% shows that an agent has collected only positive objects, but not necessarily all positive objects.

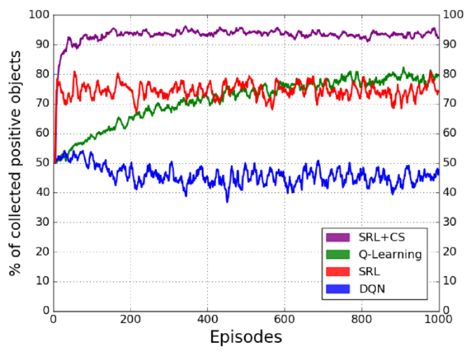
In [5], for some experiments, only this second metric is used. We argue that it should be used in connection with the accummulated scores, as done here, as otherwise it may boast the performance of a very conservative agent, one which collects very few positive objects.

Experiment 2 - Training in a Deterministic Configuration.
In this experiment, starting from the center of the grid, an agent has to collect positive objects while avoiding negative ones, all positioned in the grid by following a deterministic pattern as illustrated in Figure 6a.

The results of this experiment, shown in Figure 7, indicate that SRL+CS learns a policy which is better than that chosen by any of the other algorithms, faster than any of the other algorithms. SRL+CS collects an average rate of more than 90% of positive objects. DQN did not achieve a reasonably good performance in our experiments, probably because it requires a large number of hyper-parameter fine tunning, which was not done extensively. In this same task, results reported in [5] show that DQN can achieve a rate of almost 100% of positive objects collected, and that DSRL can achieve a rate of 70% of positive objects collected, although accumulated scores were not reported.



(a) Accumulated Scores for 10 runs of 1000 episodes each for each algorithm.



(b) Rolling mean (using a window of ten episodes) of the average percentage of positive objects collected over 10 runs.

Figure 7: Experiment 2 - Fig. 7(a) shows that SRL+CS learns faster to collect positive objects and avoid negative objects. Some Q-learning curves start to show the same slope as SRL+CS but much later. Fig. 7(b) shows that the rate of positive objects collected using SRL+CS is more than 90%. Notice that SRL+CS cannot reach 100% as an average due to the effect of 10% random exploration built into the learning algorithm.

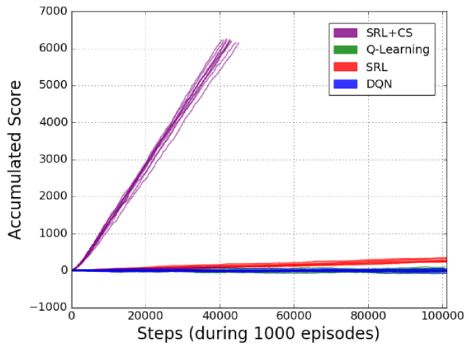
Experiment 3 - Training in a Random Configuration.
In this experiment, the positions of the objects to be placed on the grid are chosen randomly at the start of each episode, except for the agent's position, which is always the center of the grid, as exemplified in Figure 6b.

The results presented in Figure 8 show that only SRL+CS is capable of learning an effective policy in this case. It suggests that SRL+CS can learn certain sub-tasks, despite the randomness of the environment, unlike the other algorithms. SRL+CS must have learned sub-tasks independently of where the objects were located, for example, *move up if a positive object is above*, even though SRL+CS is not provided with any explicit knowledge.

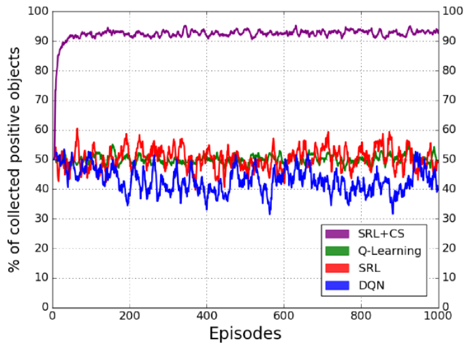
Experiment 4 - Training in a Deterministic Configuration and Testing in a Random Configuration (Zero-shot Transfer Learning).

In this experiment, we evaluate the system's ability to re-use knowledge learned in a situation (Fig. 6a) in a different yet related situation (Fig. 6b). All systems were trained on the deterministic configuration and tested on the random configuration. During testing, there was no further learning involved; hence, there was no exploration either.

The results of this experiment (Figure 9) show that only SRL+CS is able to transfer its learning correctly to a similar situation; SRL+CS collects almost 100% of the positive objects in the random environment. Such results are impressive and deserve further investigation. Figure 9b shows that SRL collects only 15% of positive objects, but SRL achieves a higher score than Q-Learning (c.f. Figure 9a). This is because in many of the runs SRL did not collect any object at all, making the average of the percentage of positive objects collected lower than a random walk (where the same



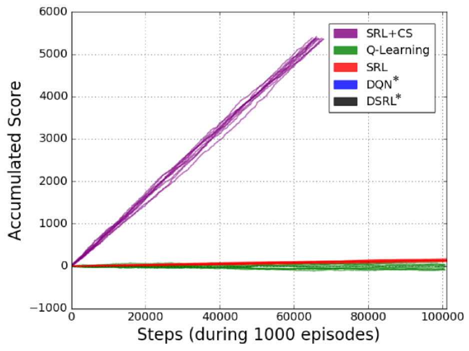
(a) Accumulated Scores for 10 runs of 1000 episodes each for each algorithm.



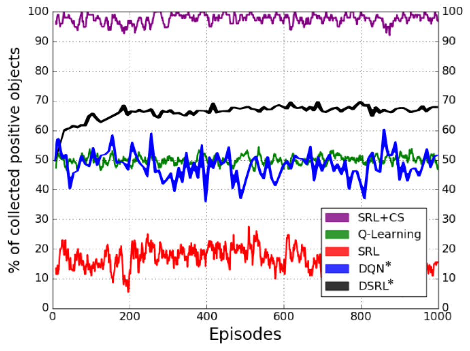
(b) Rolling mean (using a window of ten episodes) of the average percentage of positive objects collected over 10 runs.

Figure 8: Experiment 3 - Figs. 8(a) and 8(b) show that only SRL+CS learns an effective policy for the random environment. This indicates that, independently of the configuration of the environment, SRL+CS learns how to interact with the objects and to move correctly in order to collect as many positive objects as possible avoiding negative objects.

amount of negative and positive objects are collected). DQN and Q-learning were unable to transfer knowledge learned in the deterministic environment to the new situation. Additionally, DSRL was able to collect only 70% of positive objects.



(a) Accumulated Scores for 10 runs of 1000 episodes each, for each algorithm. (*) Results for DQN and DSRL are not available.



(b) Rolling mean (using a window of ten episodes) of the average percentage of positive objects collected over 10 runs. The results reported for DSRL and DQN are obtained from [5].

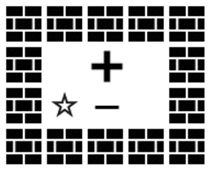
Figure 9: Experiment 4 - In this zero-shot transfer learning experiment, SRL+CS does not receive any training in the random environment task, but is the only algorithm to solve the task.

We also ran other experiments training and testing SRL+CS in a number of different game configurations, achieving similar results as reported here (apart from one case where the possible movements of the agent are restricted by the environment, e.g. in a 3x2 environment with one positive and one negative object). This is discussed next.

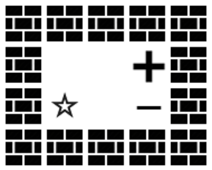
5.3 Further Transfer Learning Experiments

Finally, we run two transfer learning experiments using the three configurations shown in Figure 10. For the purpose of highlighting in the plots if a negative object is collected, in these experiments a different reward function is chosen. If a negative object is collected, a reward of -10 is obtained, and if a positive object is collected, a reward of +1 is obtained. We use 10 steps, instead of 100, as the maximum number of steps in each episode, since the environments are much smaller now.

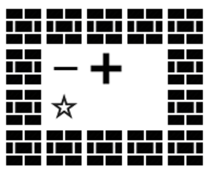
Figure 10 shows the three configurations used. The systems were trained in configuration (a) and tested in configurations (b) and (c). Figure 11 shows the algorithms' training performances in configuration (a). Out of the 10 runs for each algorithm, the best models (the ones achieving the highest scores) were selected for testing in configurations (b) and (c). Testing in these configurations was done once per algorithm since the configurations were deterministic and there was no further learning.



(a) Train



(b) Test



(c) Test

Figure 10: Initial configurations used in Experiments 5 and 6.

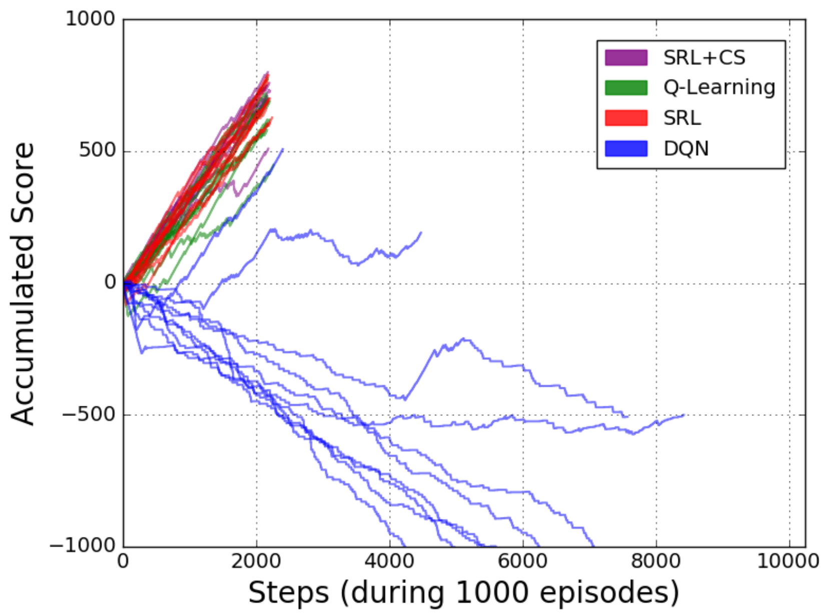


Figure 11: Scores for training in configuration (a), showing 10 runs for each algorithm, each run containing 1000 episodes. For all algorithms, apart from DQN, agents have learned to collect the positive object and to avoid the negative object. DQN, however, is quite inconsistent, although the best DQN run, whose model is selected for the rest of the experiments, achieves performance similar to the other algorithms.

Experiment 5 - Transferring Undefeasible Knowledge.
In this experiment, after training in configuration (a) the agents were tested in configuration (b). Figure 12 shows that DQN is unable to collect the positive objects, Q-learning shows a random behavior, and the symbolic algorithms were able to generalize to the new situation, thus avoiding the negative object while collecting the positive object. By inspecting configurations (a) and (b), a reasonable assumption is that the symbolic approaches can learn *to move up and to the right*, although no explicit knowledge is used (recall that Equations 3 and 4 are the only changes made to Q-learning). Comparing configurations (a) and (c) though, it is clear that knowledge learned in (a) needs to be revised in (c), as discussed next.

Experiment 6 - Negative Transfer.
In this experiment, following training in configuration (a), the systems are tested in configuration (c). Figure 13 shows the results. Now, DQN does not collect any object, as before, Q-Learning walks

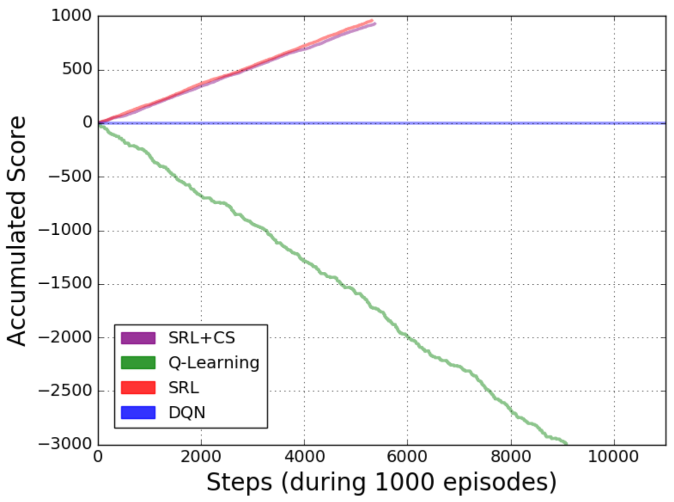


Figure 12: Experiment 5 - The symbolic algorithms are able to transfer learning correctly from environment (a) to environment (b), while Q-learning behaves randomly, and DQN never collects any object.

randomly, as before, but SRL and SRL+CS collect the negative object before collecting the positive object in every episode.

Experiment 6 was the only case where SRL+CS failed to generalize. By inspecting configurations (a) and (c), it is clear that if an agent learns *to move up and to the right* then it shall collect the negative object always in configuration (c). This is a case of negative transfer, where the system has no means of revising the knowledge learned. Notice that the problem is not in the decision-making, which should distinguish positive and negative objects, but in the learning phase. Symbolic RL algorithms use Q-learning, which learns from (state, action) pairs. This may not be desirable, as seen here, leading to poor generalization. In other words, Q-learning cannot generalize over states only. Because of this, an agent will not learn to simply *avoid a negative object* (recall that during learning this agent did not encounter a situation where it was below a negative object and executed the action of moving up). In many real-world situations, it should not matter how a state is reached, and the simple fact of being in a state can be sufficient to make a decision on a relevant action. This limitation of the generalization capacity of Q-learning is well-known.

6 Conclusion and Future Work

In this paper, we have presented a symbolic RL algorithm with features of common sense (SRL+CS) which can outperform Q-learning, DSRL and DQN in environments with deterministic and random configurations, and at transfer learning. Building on the knowledge representation proposed by deep symbolic RL, SRL+CS can make two simple modifications to the standard Q-learning algorithm, implementing common sense features such that attention can be focused normally on the objects that are of most relevance to an agent.

SRL+CS is model free: the choice of an appropriate abstraction for the representation of the state space allows for simple changes to learning and decision making which enhance the generalization capacity of RL. The experiments reported in this paper were also run with only one of the two modifications made at a time. Although each modification has led to an improvement in performance, only the combination of the two has produced the high levels of performance seen in all the tasks.

The results reported reiterate the value of the symbolic approach to RL, which through the use of sub-state abstractions can reduce the size of the state space considerably. However, we have shown that two important issues arise: the assignment of rewards to sub-states during learning and the aggregation of Q-values for decision making. SRL+CS addresses both issues adequately.

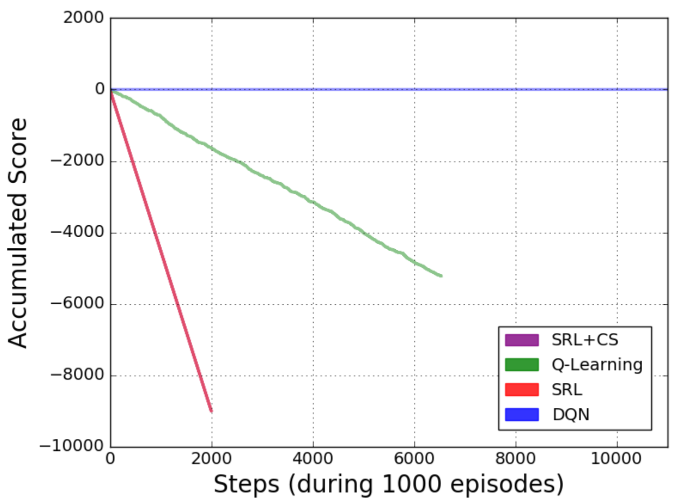


Figure 13: Experiment 6 - It this case of negative transfer, the symbolic algorithms fail, while Q-Learning behaves randomly, performing better than SRL and SRL+CS as a result, and DQN never collects any object. In this experiment, SRL and SRL+CS share exactly the same behaviour of collecting the negative object before collecting the positive object.

Various features of common sense have been argued to be relevant for achieving an adequate balance between generalization and specialization [2, 14]. Symbolic RL offers the possibility of adding to standard RL features of common sense which go beyond those considered in this paper. Despite the impressive results of SRL+CS at zero-shot learning, the experiments reported also highlight the limitations of Q-learning at generalizing from state-action representations. As future work, therefore, we aim to build an end-to-end SRL+CS architecture capable of learning common sense domain knowledge and applying it to solving more complex problems. A promising direction for further investigation is the interplay between SRL+CS and related work on the integration of planning and RL [9, 22, 25] and Relational RL [3, 18].

References

[1] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[2] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

[3] Saso Dzeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

[4] Artur S. d'Avila Garcez, Lus C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[5] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*, 2016.

[6] Artur d'Avila Garcez Ivan Donadello, Luciano Serafini. Logic tensor networks for semantic image interpretation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1596–1602, 2017.

[7] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[8] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[9] Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv preprint arXiv:1706.04317*, 2017.

[10] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[12] Guiying Li, Junlong Liu, Chunhui Jiang, and Ke Tang. Relief impression image detection: Unsupervised extracting objects directly from feature arrangements of deep cnn. *arXiv preprint arXiv:1601.06719*, 2016.

[13] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[14] John McCarthy. *Programs with common sense*. RLE and MIT Computation Center, 1960.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[18] Matthias Nickles. Integrating relational reinforcement learning with reasoning about actions and change. In *International Conference on Inductive Logic Programming*, pages 255–269. Springer, 2011.

[19] Amaia Salvador, Xavier Giró-i Nieto, Ferran Marqués, and Shin'ichi Satoh. Faster r-cnn features for instance search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16, 2016.

[20] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[21] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

[22] Richard S Sutton. First results with dyna, an integrated architecture for learning, planning and reacting. *Neural Networks for Control*, pages 179–189, 1990.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[24] S. N. Tran and A. S. d'Avila Garcez. Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–13, 2016.

[25] Harm Vanseijen and Rich Sutton. A deeper look at planning as learning from replay. In *International Conference on Machine Learning*, pages 2314–2322, 2015.

[26] Christopher Watkins and Peter Dayan Q-learning. In *Machine Learning* 8:279-292, 1992.