EPiC Series in Computing

Volume 41, 2016, Pages 174–186

GCAI 2016. 2nd Global Conference on Artificial Intelligence

When Should Learning Agents Switch to Explicit Knowledge?

Daan Apeldoorn[1] and Gabriele Kern-Isberner[2]

[1] Technische Universität Dortmund, Dortmund, Germany
daan.apeldoorn@tu-dortmund.de
[2] Technische Universität Dortmund, Dortmund, Germany
gabriele.kern-isberner@cs.tu-dortmund.de

Abstract

According to psychological models, learned knowledge can be distinguished into implicit and explicit knowledge. The former can be exploited, but cannot be verbalized easily (e. g., to explain it to another person). The latter is available in an explicit form, it often comprises generalized, rule-based knowledge which can be verbalized and explained to others. During a learning process, the learned knowledge starts in an implicit form and gets explicit as the learning process progresses, and humans benefit from exploiting such generalized, rule-based knowledge when learning. This paper investigates how learning agents can benefit from explicit knowledge which is extracted during a learning process from a learned implicit representation. It is clearly shown that an agent can already benefit from explicit knowledge in early phases of a learning process.

1 Introduction

When learning new tasks, a common and general approach for learning agents is to gain knowledge by starting with random actions and by perceiving their consequences (e. g., the resulting subsequent states and their rewards). This quickly leads to the idea of representing the gained knowledge in an implicit, sub-symbolic way, e. g., as weighted pairs of states and actions. Reinforcement Learning (RL) [15, 11], as an instance of this general approach, is known to work quite well for many applications (e. g., [12, 6]) and recent research also outlines its psychological foundation [14]. However, it is also known from psychological experiments, that human reasoning incorporates the ability of generalization and distillation of (rule-based) knowledge [2] (pp. 137–139). This kind of knowledge is *explicit* and can be verbalized and explained to others.

This paper investigates the incorporation of implicit and explicit knowledge, as it is known from human learning processes, in the context of agents. For this purpose, symbolic rule-based knowledge is created during the learning process of an agent using RL as learning paradigm.

The main contribution of the paper is the empirical evidence that learning agents can clearly benefit from creating and exploiting explicit knowledge, similar to the way humans do when learning complex tasks. It is shown that learning to solve a task can be significantly accelerated by creating and exploiting explicit symbolic knowledge. As a side-contribution, a multi-

abstraction-level knowledge base and its extraction from implicit sub-symbolic knowledge will be outlined which offers an explanatory view on the structure of the learned task and its solution.

After discussing related work in Section 2, the creation of a multi-abstraction-level knowledge base from implicit sub-symbolic knowledge is outlined in Section 3 as preliminary work to the experimental set-up. Subsequently, the experimental set-up is presented in Section 4 and the empirical evaluation is done in Section 5. Finally, a conclusion and remarks on future work are provided in Section 6.

2 Related Work

Different attempts have already been made to incorporate symbolic knowledge (e. g., rule-based knowledge) with sub-symbolic learning approaches.

In [1], an object-oriented representation of the environment is integrated with RL to link environment states with symbols. Although this is an interesting approach, in our case, the linkage between states and symbols follows a very simple and generic approach by simply associating one symbolic variable with every dimension of the state space (see Section 3.1).

In [10], extraction approaches are proposed to gain both simple rules and plans from RL and in [4] decision trees are created from learned weighted state-action representations with the primary goal of supporting agent developers in the implementation of adequate agent behavior. These works focus on the extraction of knowledge in different forms but no attempts were made to evaluate whether agents themselves can benefit from the extracted explicit knowledge.

In [5] sub-symbolic learning was successfully combined with belief revision to support the learning process and it was shown on a object recognition task that the considered learning agent can benefit from the belief revision mechanism (with and without additional background knowledge). The focus of [5] lies on the incorporation of the two paradigms rather than on making learned knowledge explicit. However, it was not systematically investigated, in which phase during the learning process the agent should rely its behavior more on the explicit, symbolic part of the approach rather than on the implicit representation of the weights learned through RL.

In general, there seems to be no attempt to investigate the point during a learning process, when agents benefit most from creating generalized rule-based knowledge as a model of their environment and relying their behavior on it, although this is an interesting question since humans are doing this *intuitively*.

Nowadays, different sub-symbolic learning approaches exist, and especially in the context of agents, RL is a widely used paradigm. RL approaches differ in the way how weights are updated when perceiving subsequent states and rewards (e. g., *Q-Learning* [15], *SARSA* [8]), how many previous states are considered by the update, how the action selection is realized to balance exploration and exploitation (e.g., *VDBE* [13]) and how the problem of high dimensional state-action spaces in larger problems can be tackled (e. g., [3]). Also in recent works, RL has been combined with other modern learning approaches like *Deep Neural Networks* [6] to increase the learning performance.

The knowledge extraction approach which is introduced in this paper to be used for the experiments, can be combined with both the classical and the newer learning approaches mentioned here, given that the underlying learning approach provides a weighted mapping of inputs (e. g., percepts) to corresponding outputs (e. g., actions). The performance of our approach should actually increase with the performance of the underlying learning approach.

Unlike Inductive Logic Programming (ILP) [7], no background knowledge is involved here and the presented approach focuses on the weights of the state-action pairs.

The focus of this paper lies on the investigation how learning agents can benefit from symbolic knowledge representation—not in the sense of *a priori knowledge* that is provided to an agent to support the learning process (as has been done e. g. in [9, 5])—but in the sense of generalizing, explicit rule-based knowledge that will be created by the learning agent itself during the learning process, similar to the way humans do when learning (cf. [2], pp. 137–139).

3 Preliminaries

This section presents the preliminaries needed for the experiments done in Section 5. It starts with a short introduction to the multi-abstraction-level knowledge base (representing the explicit knowledge) and how it is extracted from the implicit representation of weighted state-action pairs (Sections 3.1). After that, the reasoning algorithm will be outlined, which is used by the agent to exploit the explicit knowledge created during the learning process (Section 3.2).

3.1 Knowledge Extraction

As basic preliminary, the previously learned knowledge is assumed to be available in an implicit form, i. e., as weighted state-action-pairs which are stored in an *extended* weight matrix $\hat{Q}$: The matrix comprises $n+1$ dimensions, where the first n dimensions represent states in the state space $\mathbb{S} = \mathbb{S}_1 \times ... \times \mathbb{S}_n$ of the task to be learned and the last dimension represents actions in the action space $\mathbb{A}$ which comprises the agent's possible actions.[1] In principle, it is not relevant how the weights were learned and any learning paradigm can be used to fill the weight matrix: The more efficient the learning approach (i. e., the faster the weights represent the adequate behavior of the agent), the better the knowledge extraction should perform.

To represent the knowledge contained in the learned weights of the extended $\hat{Q}$-matrix adequately in a rule-based way, some *representation criteria* have to be determined first. As a model, we consider a human explaining a previously learned task to another person, since such explainable knowledge can be considered *explicit*. The explicit, explanatory knowledge is expected to be

- **Criterion 1:** *adequately relevant*
  (the knowledge should be restricted to the relevant parts only), and
- **Criterion 2:** *adequately generic*
  (equivalent, more general knowledge should be preferred over more specific knowledge).

To be able to fulfill these criteria, the following definitions are helpful which will be used later to outline the knowledge extraction approach:

Complete states and elementary rules. A *complete state* s can be constructed from the conjunction of all *partial states* $s_1 \wedge ... \wedge s_n$ (with $s_i \in \mathbb{S}_i$) and an *elementary rule* has the form $s_1 \wedge ... \wedge s_n \Rightarrow a\ [w]$ (where a is an action and $w = q_{s,a}$ is the weight determining the quality of the action given that state s is perceived).

Elementary and generalized rules. Elementary rules are the most specific rules possible, but to get compact knowledge bases, *generalized rules* have to be found (where possible) which do not contain all of the s_i in the premises. These more general rules can be considered *aggregations* of the more specific rules (i. e., the weights are aggregated over all values of the missing

[1] Note, that the matrix will be implemented in a sparse form, since usually only a few state-action-pairs are relevant for performing the knowledge extraction.

s_i to create a weight for the disjunction of these values[2]). An example on how elementary rules will be aggregated to generalized rules will be provided by Example 1.

The knowledge extraction algorithm takes an extended weight matrix $\hat{Q}$ as input (which contains the implicit knowledge in form of the learned weights) and returns a knowledge base $\mathcal{KB}^{\hat{Q}}$ which reflects $\hat{Q}$ according to the preliminaries introduced in the beginning of this section. Since we want to create the explicit knowledge as a compact representation of the knowledge implicitly contained in $\hat{Q}$, and due to reasons of computational complexity, only the weights of the best state-action sequence found until the learning process was stopped will be considered.[3]

The knowledge extraction algorithm performs the following steps:

1. *Normalization of weights:*
   Every weight $w_{\text{raw}} \in \hat{Q} = (\hat{q}_{s_1,\ldots,s_n,a})$ is normalized over the action dimension to
   $$w = \frac{w_{\text{raw}}}{\max_{a' \in \mathbb{A}}(\hat{q}_{s_1,\ldots,s_n,a'})}$$

2. *Creation of rule sets:*
   All generalized rules (i. e., all rules $\bigwedge_{i \in I} s_i \Rightarrow a\ [w]$ with $s_i \in \mathbb{S}_i$ and with $I \subset \{1,\ldots,n\}$ where n is the number of state space dimensions) are created by aggregating the average weight $\bar{w}$ over all missing state space dimensions (i. e., over those state space dimensions $\mathbb{S}_i$ of which no value is contained in the premise of the respective rule). Note, that (as already mentioned) only the state-action pairs contained in the best found state-action sequence will be considered in this step.

Example 1. *Consider a 2-dimensional state space $\mathbb{S} = \mathbb{T} \times \mathbb{U}$ with $\mathbb{T} = \{t_1, t_2\}$ and $\mathbb{U} = \{u_1, u_2\}$, an action space $\mathbb{A} = \{a_1, a_2\}$ and an extended weight matrix $\hat{Q} = (\hat{q}_{t,u,a})$. Then the following 10 generalized rules are potentially created (depending on what states are actually contained in the sparse matrix $\hat{Q}$):*

$$\left.\begin{array}{l} \top \Rightarrow a_1\ [\bar{w}_1] \quad with \quad \bar{w}_1 = \frac{1}{4} \sum\limits_{t \in \mathbb{T}, u \in \mathbb{U}} \hat{q}_{t,u,a_1} \\ \top \Rightarrow a_2\ [\bar{w}_2] \quad with \quad \bar{w}_2 = \frac{1}{4} \sum\limits_{t \in \mathbb{T}, u \in \mathbb{U}} \hat{q}_{t,u,a_2} \end{array}\right\} \textit{most general rules}$$

$$\left.\begin{array}{l} t_1 \Rightarrow a_1\ [\bar{w}_3] \quad with \quad \bar{w}_3 = \frac{1}{2} \sum_{u \in \mathbb{U}} \hat{q}_{t_1,u,a_1} \\ t_1 \Rightarrow a_2\ [\bar{w}_4] \quad with \quad \bar{w}_4 = \frac{1}{2} \sum_{u \in \mathbb{U}} \hat{q}_{t_1,u,a_2} \\ t_2 \Rightarrow a_1\ [\bar{w}_5] \quad with \quad \bar{w}_5 = \frac{1}{2} \sum_{u \in \mathbb{U}} \hat{q}_{t_2,u,a_1} \\ t_2 \Rightarrow a_2\ [\bar{w}_6] \quad with \quad \bar{w}_6 = \frac{1}{2} \sum_{u \in \mathbb{U}} \hat{q}_{t_2,u,a_2} \\ u_1 \Rightarrow a_1\ [\bar{w}_7] \quad with \quad \bar{w}_7 = \frac{1}{2} \sum_{t \in \mathbb{T}} \hat{q}_{t,u_1,a_1} \\ u_1 \Rightarrow a_2\ [\bar{w}_8] \quad with \quad \bar{w}_8 = \frac{1}{2} \sum_{t \in \mathbb{T}} \hat{q}_{t,u_1,a_2} \\ u_2 \Rightarrow a_1\ [\bar{w}_9] \quad with \quad \bar{w}_9 = \frac{1}{2} \sum_{t \in \mathbb{T}} \hat{q}_{t,u_2,a_1} \\ u_2 \Rightarrow a_2\ [\bar{w}_{10}] \quad with \quad \bar{w}_{10} = \frac{1}{2} \sum_{t \in \mathbb{T}} \hat{q}_{t,u_2,a_2} \end{array}\right\} \textit{more specific rules}$$

[2] Only the relevant states contained in the sparse matrix are considered here.

[3] Combined with the implementation of $\hat{Q}$ in sparse form, this significantly reduces the state-action pairs that have to be considered by the knowledge extraction algorithm.

The resulting rules will be grouped according to their generality into different sets $R_1, ..., R_{n+1}$ where R_1 contains the most general rules and R_{n+1} contains the most specific, the elementary rules. The elementary rules are derived directly from the $\hat{Q}$-matrix.

3. *Removal of worse rules:*
   Intuition: "Restrict to relevant knowledge" (corresponds to Criterion 1, Section 3.1).
   In all sets R_j, a rule $\rho \in R_j$ is removed, if there exists another rule $\sigma \in R_j$ with the same partial state as premise having a higher weight (i. e., in every set R_j only the best rules for a given partial state are kept).

4. *Removal of worse more specific rules:*
   Intuition: "Prefer general over specific knowledge" (corresponds to Criterion 2, Sec. 3.1).
   In all sets R_j, a rule $\rho \in R_j$ with premise $p_\rho = s_1 \wedge ... \wedge s_n$, conclusion a_ρ and weight w_ρ is removed, if there exists a more general rule $\sigma \in R_{j'<j}$ with premise $p_\sigma = \bigwedge_{s \in S_\sigma} s$ where $S_\sigma \subset S_\rho = \{s_1, ..., s_n\}$ and with weight $w_\sigma \geq w_\rho$.

5. *Removal of too specific rules:*
   Intuition: "Prefer general over specific knowledge, if the more specific knowledge is not necessarily needed/relevant" (corresponds to Criteria 1 and 2, Section 3.1).
   In all sets R_j, a rule $\rho \in R_j$ with premise $p_\rho = s_1 \wedge ... \wedge s_n$ and conclusion a_ρ is removed, if there exists a more general rule $\sigma \in R_{j'<j}$ with the same action $a_\sigma = a_\rho$ as conclusion and with premise $p_\sigma = \bigwedge_{s \in S_\sigma} s$ where $S_\sigma \subset S_\rho = \{s_1, ..., s_n\}$ and if ρ is not a *needed exception* to a rule $\tau \in R_{j-1}$.
   A rule σ is an *exception*, to a rule $\tau \in R_{j-1}$ with premise $p_\tau = \bigwedge_{s \in S_\tau} s$, action a_τ as conclusion and weight w_τ, if $S_\tau \subset S_\rho$ and $a_\rho \neq a_\tau$. The exception is *needed*, if there exists no other rule $\upsilon \in R_{j-1}$ with premise $p_\upsilon = \bigwedge_{s \in S_\upsilon} s$ and action a_υ as conclusion where $S_\upsilon \subset S_\rho$, $a_\upsilon = a_\rho$ and $w_\upsilon > w_\tau$.

6. *Optional filter step:*
   Optionally, filters may be applied to filter out further rules which are helpful to explain the knowledge contained in $\hat{Q}$, but which are not needed for reasoning later (e. g., since they will never fire given all states contained in $\hat{Q}$, or since other rules exist on the same level of abstraction which would lead to the same result when reasoning is performed on the extracted knowledge base, see Section 3.2). Since the optional filter step only serves to simplify the resulting knowledge base $\mathcal{KB}^{\hat{Q}}$ without having any effect on the reasoning behavior, no further details on this step will be provided here.

After performing these steps on $\hat{Q}$, the knowledge base $\mathcal{KB}^{\hat{Q}}$ comprises all sets $R_j \neq \emptyset$ with the extracted rules representing the implicit knowledge contained in the learned weights of $\hat{Q}$ in a compact way. Algorithm 1 summarizes the knowledge extraction algorithm.

The algorithm presented here creates nice and compact knowledge bases with multiple abstraction levels, which explain the structure of the underlying problem and its solution in an easy and comprehensible way (examples will be shown in Section 5.3).

3.2 Reasoning

The reasoning algorithm takes the knowledge base $\mathcal{KB}^{\hat{Q}}$ (created by Algorithm 1) and the current perceived state of the agent as input and it outputs a set $A \subseteq \mathbb{A}$ of inferred actions.[4] The

[4]Usually the set A contains only one single action. Only in case of multiple equivalent rules with the same maximum weight and different conclusions exist on a level R_j, more than one action could be contained in A.

```
01  % Normalization
02  % (according to Section 3.1, Step 1)
03  normalize( Q̂ )
04
05  % Initial creation of the rule sets
06  % (according to Section 3.1, Step 2)
07  R^Q̂ := {R_1, ..., R_{n+1}}   % Ordered set of rule sets
08
09  % Removal of worse rules
10  % (according to Section 3.1, Step 3)
11  for each R ∈ R^Q̂ do
12      for each ρ ∈ R do
13          if ∃σ ∈ R: p_σ = p_ρ, w_σ > w_ρ then
14              R := R \ {ρ}
15          end if
16      end for
17  end for
18
19  % Removal of worse more specific rules
20  % (according to Section 3.1, Step 4)
21  for j := 2 to n + 1 do
22      R := R_j ∈ R^Q̂
23      for each ρ ∈ R do
24          if ∃σ ∈ R_{j'<j}: S_σ ⊂ S_ρ, w_σ ≥ w_ρ then
25              R := R \ {ρ}
26          end if
27      end for
28  end for
29
30  % Removal of too specific rules
31  % (according to Section 3.1, Step 5)
32  for j := 2 to n + 1 do
33      R := R_j ∈ R^Q̂
34      for each ρ ∈ R do
35          if ∃σ ∈ R_{j'<j}: a_σ = a_ρ, S_σ ⊂ S_ρ and
36              (∄τ ∈ R_{j-1}: a_ρ ≠ a_τ, S_τ ⊂ S_ρ or
37               ∃υ ∈ R_{j-1}: a_υ = a_ρ, S_υ ⊂ S_ρ, w_υ > w_τ) then
38              R := R \ {ρ}
39          end if
40      end for
41  end for
42
43  % Perform optional filter steps
44  % (according to Section 3.1, Step 6)
45  filter( R^Q̂ )
46
47  KB^Q̂ := {R ∈ R^Q̂ | R ≠ ∅}
```

Algorithm 1: Knowledge extraction

```
01  % Initialize the set of all partial states of
02  % the current perceived state s, the inferred
03  % actions A and index j
04  S := {s_1, ..., s_n}
05  A := ∅
06  j := |KB^Q̂|
07
08  % Search for most specific rules whose premise
09  % is satisfied by the current perceived state s
10  % with maximum weight among all satisfied rules
11  while A = ∅ and j > 1 do
12      R := R_j ∈ KB^Q̂
13      for each ρ ∈ R do
14          if S_ρ ⊆ S and ∄σ ∈ R: S_σ ⊆ S, w_σ > w_ρ then
15              A := A ∪ {a_ρ}
16          end if
17      end for
18      j := j − 1
19  end while
```

Algorithm 2: Reasoning

knowledge base $\mathcal{KB}^{\hat{Q}} = \{R_1, ..., R_{n+1}\}$ consists of an ordered set of rule sets, where R_1 contains the most general and R_{n+1} contains the most specific, the elementary rules (see Section 3.1).

When a state $s = s_1 \wedge ... \wedge s_n$ is perceived, the reasoning algorithm searches for the most specific rules ρ whose premises are satisfied by s and which have the highest weight among all satisfied rules on the same level of specificity (i. e., rules ρ with premise $p_\rho = \bigwedge_{s' \in S_\rho} s'$ where S_ρ is a subset of the set $S = \{s_1, ...s_n\}$ and no other rule σ exists with premise $p_\sigma = \bigwedge_{s' \in S_\sigma} s'$ where $S_\sigma \subseteq S$ and $w_\sigma < s_\rho$). The actions A will then be returned (in case $|A| > 1$ the returned actions are equivalent and the agent may select randomly among them). Algorithm 2 shows the reasoning algorithm.

Example 2. *Consider again the 2-dimensional state space* $\mathbb{S} = \mathbb{T} \times \mathbb{U}$ *and the action space* $\mathbb{A}$ *from Example 1. Furthermore, assume that Algorithm 1 resulted in the extracted knowledge base* $\mathcal{KB}^{\hat{Q}} = \{R_1, R_2\}$ *with* $R_1 = \{\top \Rightarrow a_1\ [\bar{w}_1]\}$ *and* $R_2 = \{t_1 \Rightarrow a_2\ [\bar{w}_3]\}$ *and that state* $s = t_2 \wedge u_1$ *is perceived. Then, the reasoning algorithm will return* $A = \{a_1\}$*, since* $\{t_1\} \not\subseteq \{t_2, u_1\}$.

4 From Implicit to Explicit Knowledge

After having provided the basic algorithms for knowledge extraction and reasoning in the previous section, we will now explain the main idea of our approach to connect learning and rule-based reasoning, and describe the test scenarios for the experimental set-up. As learning approach, an instance of RL (classical Q-Learning [15]), will be used to fill the extended weight matrix $\hat{Q}$. The approach is slightly modified to be able to handle the multi-dimensional state-spaces where every dimension represents the values of one kind of percepts (i. e., the values of one of the agent's sensors).

4.1 Basic Idea

Humans seem to incorporate both sub-symbolic learning with symbolic (rule-based) knowledge representation and reasoning capabilities: When being confronted with a new, previously unknown environment (e. g., a new task to be solved), there seems to be a *learning process* until we are able to create a (simplified) model of the new environment which can be represented explicitly by symbolic knowledge (e. g., by simple but generalizing rules describing how to solve the task). At this point in the learning process, we can make the learned knowledge *explicit* and we are able to explain the learned task to someone else based on the explicit model. In addition, we are also able to exploit the model and to benefit from its simplified representation. This can also be observed in psychological experiments (as has been done e. g. in [2]).

From this observation, two central questions arise:

- Is a learning agent also able to benefit from extracted rule-based knowledge?
- If so, in which phase of the learning process the agent benefits most from relying its behavior on the explicit knowledge?

  If the model is created too early during the learning process, the extracted knowledge will be of low quality (and even comprise wrong rules); if the model is created too late during the learning process, the benefit from exploiting the simplified model will be relatively low.

In the following, we make a first attempt to answer these questions by investigating them in the context of two typical RL test scenarios. This will be done by letting an agent interact with an environment, until the learned optimal policy can be considered *stable* (see Section 5.2). This whole learning process will be repeated multiple times and the point when the knowledge is extracted though Algorithm 1 and the agent relies its action selection on Algorithm 2 will be varied. The details of these experiments and the results are provided in the following sections.

4.2 Test Scenarios

We consider two simple and typical RL test scenarios which will be used for the experiments to investigate at which point during a learning process learning agents can benefit most from relying their behavior on extracted symbolic knowledge rather than on the knowledge implicitly contained in the learned extended weight matrix $\hat{Q}$.

Scenario 1. In this scenario an agent (e. g., a robot) has to learn to get from a starting point A to a target point B in an unknown environment represented by a 2-dimensional grid world of size 8×6. The agent is able to perceive its current x- and y-position. Thus, its state space is given by $\mathbb{S} = \mathbb{S}_x \times \mathbb{S}_y$ with $\mathbb{S}_x = \{x_0, ..., x_7\}$ and $\mathbb{S}_y = \{y_0, ..., y_5\}$. The agent is able to perform four different actions corresponding to the four cardinal directions. Thus, the action space is given by $\mathbb{A} = \{\text{North}, \text{South}, \text{East}, \text{West}\}$. The agent should reach the target point B efficiently in as few as possible steps (i. e., all states except the target point B are rewarded negatively). The target point represents a terminal state, thus reaching this state determines the end of a *learning episode*. The left part of Figure 1 shows the Scenario.

Obviously, in this scenario, the optimal policy is easy to explain: The agent simply has to go to east to reach the point B. Nevertheless, the optimal policy is not that easy to find, since the agent does not know anything in advance about the environment and therefore has to explore the environment (which is much larger than the optimal policy). As a consequence, storing the knowledge about the way from A to B would comprise $8 \cdot 6 \cdot 4 = 192$ weights, instead of a knowledge base with a single symbolic rule which would be sufficient here.

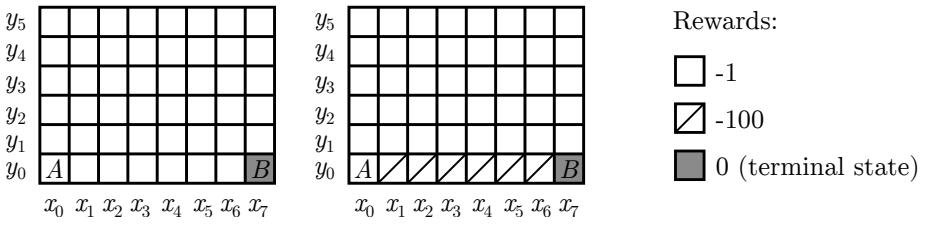

Figure 1: Test scenarios 1 (left) and 2 (right) with corresponding rewards

Scenario 2. The second scenario is quite similar to Scenario 1. In addition, in this scenario, there is a large area of negative reward (e. g., water or swamp) which must be taken into account by the agent when getting from A to B. The right part of Figure 1 shows the scenario.[5]

Compared to Scenario 1, in this scenario, the optimal policy is more complex, since the agent must not only learn to reach the target point B, but also to avoid the highly negative rewarded area. Nevertheless, both scenarios have in common that a larger area has to be explored, even though the tasks themselves are easy to describe.

5 Experiments

At the beginning of this section, the learning approach used in the experiments to learn the implicit knowledge will be described (Section 5.1). Subsequently, *stable policies* will be defined for the test scenarios to determine (approximately) when the agent completed the learning process in the corresponding scenario (Section 5.2). After that, the knowledge extraction approach will be illustrated, by showing to which knowledge bases the approach converges in case of the two test scenarios (Section 5.3). Finally, it will be investigated at which point during the learning process the agent can benefit most from relying its behavior on the extracted explicit knowledge rather than on the learned weights of the extended $\hat{Q}$-matrix (Section 5.4).

5.1 Learning Approach

As sub-symbolic learning approach for the implicit knowledge, a classical RL algorithm (Q-Learning [15]) will be used. During the learning process, the weights are updated according to the following formula [15]:

$$q'_{s,a} = (1-\alpha) \cdot q_{s,a} + \alpha \cdot (r_s + \gamma \cdot \max_{a' \in \mathbb{A}} q_{s',a'}) \tag{1}$$

where $q_{s,a}$ is the old weight and $q'_{s,a}$ is the new weight determining the quality of performing action a in state s and s' is the subsequent state. $\alpha \in [0,1]$ is the *learning rate* which determines how much of a currently perceived reward is used for learning and $\gamma \in [0,1]$ is the *discount factor* which determines to which degree future states are taken into account.

Furthermore, to explore the environment, an *exploration rate* $\epsilon \in [0,1]$ determines the probability that a random action is chosen, instead of the best action already learned according to the current weights.

The approach is only slightly modified to be able to handle the multi-dimensional state-spaces of the extended $\hat{Q}$-matrix. In addition, the current best state-action sequence is kept

[5] This scenario is taken from a Soft-Computing tutorial at University of Mainz in 2006 by Peter Dauscher and Tobias Jung.

	optimal policy length l	no. of subsequent opt. policy runs k	average no. of runs $\bar{r}$
Scenario 1	7	6	≈196
Scenario 2	9	7	≈219

Table 1: Parameters of the test scenarios

after every learning episode (i. e., after every run during the learning process), to be able to apply the knowledge extraction algorithm to the relevant state-action pairs only (see Section 3.1).

In the following experiments, a learning rate of $\alpha = 0.1$, a discount factor of $\gamma = 0.9$ and an exploration rate of $\epsilon = 0.1$ will be used.

5.2 Stable Policies

The agent learns a policy π while interacting with the environment through multiple iterations of the learning approach described in Section 5.1. Whenever the agent reaches the terminal state (point B of the test scenarios) the current *run* ends and the agent's position is reset to the starting point (point A). After a certain number of runs, the optimal policy π^* is learned.

An important question is now, at which point during the learning process (i. e., after how many runs) the learned optimal policy π^* can be considered to be *stable*. Since the agent explores its environment randomly (see Section 5.1), we cannot consider π^* to be stable, when the agent found it for the first time. Instead, π^* should be considered to be stable, if the probability, that the found optimal policy π^* will be changed again, is smaller than a certain threshold. Since an important preliminary for changing an already found policy is *exploration* (i. e., performing actions which are suboptimal according to the current weights of the extended $\hat{Q}$-matrix), π^* is considered to be stable if the following inequation holds for a minimal k:

$$(1 - (1 - \epsilon\phi)^l)^k \leq \beta \tag{2}$$

where ϵ is the random action probability used in Algorithm 1, $\phi = 1 - \frac{1}{|\mathbb{A}|} = 0.75$ is the conditional probability that a suboptimal action is performed given that a random action is performed, l is the number of steps of the optimal policy π^*, k is the number of (subsequent) runs and β is the threshold for the probability that exploration is involved in performing π^* in k subsequent runs.

For the following experiments, we choose $\epsilon = 0.1$, $\beta = 0.01$ and by solving (2), we get the respective values of a minimal k for the test scenarios. Having these parameters, the average number $\bar{r}$ of runs that are needed to learn the optimal policy π^* stably, can be determined for every test scenario. The learned optimal policy is then considered to be stable after $\bar{r}$ runs (averaged over 200 repetitions). Table 1 summarizes the parameters of the test scenarios for the following experiments.

5.3 Illustrating the Knowledge Extraction Algorithm

In this section, the knowledge extraction algorithm (Algorithm 1) will be demonstrated in the context of the two test scenarios: This will be done by running the learning approach from Section 5.1 in every scenario for $\bar{r}$ runs (according to Table 1). After that, Algorithm 2

$\mathcal{KB}_1^{\hat{Q}}$:

$\top \Rightarrow$ East [1.00]	R_1

$\mathcal{KB}_2^{\hat{Q}}$:

$\top \Rightarrow$ East [0.78]	R_1
$x_7 \Rightarrow$ South [1.00] $y_0 \Rightarrow$ North [1.00]	R_2

Figure 2: Extracted knowledge bases for the test scenarios normally obtained after $\bar{r}$ runs, according to Table 1 (weights are rounded to two decimals)

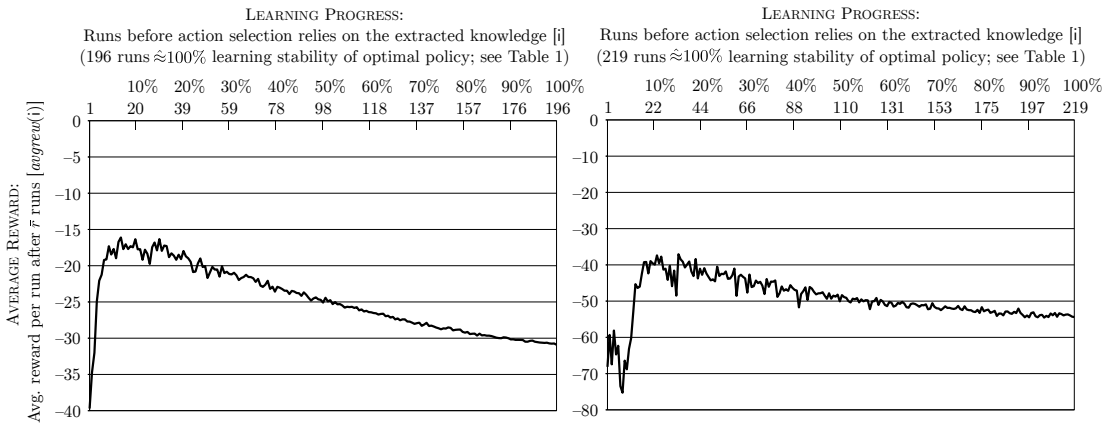

Figure 3: Results for Scenarios 1 (left) and Scenario 2 (right)

is performed to extract the knowledge from the learned weights of the extended $\hat{Q}$-matrix. Figure 2 shows the resulting knowledge bases for Scenario 1 and Scenario 2.

5.4 Results

This section presents the results of the experiments, which are shown in Figure 3.

The results show clearly that the agent benefits from relying its behavior on the extracted explicit knowledge even in early stages of the learning process. This is due to the fact, that the agent has to explore the whole state-action space, but the environment of the scenario is structured according to rather simple rules (e. g., in case of Scenario 1, the goal can be reached by simply going to the east). This generalization is found quickly by already exploiting the explicit knowledge in early stages of the learning process.

At the beginning of the learning process, the explicit knowledge represented by the extracted knowledge bases is of rather low quality, since the agent did not yet collect enough experience for these kinds of generalization. This can lead to knowledge bases comprising over-generalized and wrong rules. This effect is reflected in the early phase of the learning process (up to approximately 15% learning stability of the optimal policy π^*) by the fluctuations of the curve. This seems to be related to psychological findings, which show that the human ability of learning and generalizing from few examples can also lead to over-generalization [2]. The optimal amount

of learning stability found here for relying the action selection on the extracted knowledge is around 15%.

In case of Scenario 2 (right part of Figure 3), the effect that the extracted explicit knowledge is of low quality in early phases of the learning process is even more visible (which is reflected by the stronger fluctuations of the curve), since the environment of the scenario is structured according to more complex rules and thus early generalizations are more probable to be misleading.

6 Conclusion and Future Work

In this paper, the benefit of exploiting explicit symbolic knowledge in the context of learning agents was investigated. The explicit knowledge was created by the introduced knowledge extraction algorithm from a sub-symbolic implicit representation of learned state-action pairs and the explicit knowledge was represented using a multi-abstraction-level knowledge base. The algorithm was implemented and tested in a learning cycle of a RL agent using Q-Learning to learn the weights of the implicit representation and the benefit has been empirically studied in the context of two typical RL test scenarios. It was shown that an agent can clearly benefit from explicit symbolic knowledge even in early stages of the learning process.

In addition, by making use of the presented approach, the learned knowledge implicitly contained in the weights of the extended $\hat{Q}$-matrix can be represented in a very compact way focusing on the relevant parts of the learned knowledge. Besides the compression aspect (once the knowledge base is extracted, it can replace the extended $\hat{Q}$-matrix which can comprise thousands of weights in real world scenarios), the extracted knowledge is easy to comprehend which can serve to explain the behavior learned by an autonomous agent.

Humans are able to learn from few examples by generalizing their experiences adequately. Thereby, on the one hand, humans can cope with large state-action spaces. On the other hand, it is also known from psychological experiments, that humans tend to over-generalization which can result in bad decisions [2]. The experimental results shown in Figure 3 seem to reflect these effects adequately: Creating explicit generalizing knowledge at the right time during a learning process increases the learning performance significantly. Done too early, this can even decrease the performance (which is reflected by the larger fluctuations in the early phases of the leaning process in Figure 3). However, overall, there is a clear benefit from exploiting the explicit generalized knowledge.

Furthermore, humans are able to explain complex learned tasks in an easy way on an adequate level of abstraction. This is well reflected by the multi-abstraction-level knowledge base which was proposed here to model the explicit knowledge.

For future work, it would be interesting to also investigate the reverse direction: When learning agents should give up such created explicit knowledge in case of new percepts (e. g., from a new environment) which do not seem to fit to the previously created explicit representation. Furthermore, it would be interesting to apply the results to transfer learning, e. g., how the multi-abstraction-level knowledge bases can be adapted to new tasks (e. g., to avoid complete relearning) and to incorporate the approach to a sound cognitive architecture. Future work could also comprise applications to real world scenarios, e. g., to make knowledge about learned tasks and their solutions explicit and comprehensible for humans.

Acknowledgements. Thanks to Viola Gauß for helpful discussions and to Jan Eric Lenssen and Henrik Heimbürger for providing computational resources for some experiments.

References

[1] C. Diuk, A. Cohen, and M. L. Littman. *An Object-Oriented Representation for Efficient Reinforcement Learning*, pages 240–247. Proceedings of the 25th International Conference on Machine Learning. Omnipress, Madison, Wisconsin, 2008.

[2] D. Dörner. *Die Logik des Mißlingens – Strategisches Denken in komplexen Situationen.* Rowohlt Taschenbuch Verlag, Reinbek bei Hamburg, 1992.

[3] T. Jung. *Reinforcement Lernen mit Regularisierungsnetzwerken.* Johannes Gutenberg-Universität Mainz, Mainz, 2007.

[4] R. Junges and F. Klügl. Learning tools for agent-based modeling and simulation. *KI – Künstliche Intelligenz*, 27(3):273–280, 2013.

[5] T. Leopold, G. Kern-Isberner, and G. Peters. *Belief Revision with Reinforcement Learning for Interactive Object Recognition*, pages 65–69. ECAI 2008 – 18th European Conference on Artificial Intelligence Proceedings. IOS Press, Amsterdam, 2008.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[7] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19–20(1):629–679, 1994.

[8] G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. CUED/F-INFENG/TR 166, Cambridge University Engineering Department, England, 1994.

[9] D. Shapiro, P. Langley, and R. Shachter. *Using Background Knowledge to Speed Reinforcement Learning in Physical Agents*, pages 254–261. AGENTS '01 Proceedings of the fifth international conference on Autonomous agents. ACM, New York, 2001.

[10] R. Sun. *Knowledge Extraction from Reinforcement Learning*, pages 170–180. New Learning Paradigms in Soft Computing. Springer, Berlin Heidelberg, 2002.

[11] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, Massachusetts; London, England, 1998.

[12] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[13] M. Tokic. *Adaptive ϵ-Greedy Exploration in Reinforcement Learning Based on Value Differences*, pages 203–210. KI 2010: Advances in Artificial Intelligence. Springer, Berlin Heidelberg, 2010.

[14] M. Tokic. Reinforcement Learning: Psychologische und neurobiologische Aspekte. *KI – Künstliche Intelligenz*, 27(3):213–219, 2013.

[15] C.J.C.H. Watkins. *Learning from Delayed Rewards.* University of Cambridge, England, 1989.