Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference

Towards an Understanding of What Is Learned: Extracting Multi-Abstraction-Level Knowledge from Learning Agents

Daan Apeldoorn, Gabriele Kern-Isberner
daan.apeldoorn@tu-dortmund.de, gabriele.kern-isberner@cs.tu-dortmund.de
Technische Universität Dortmund

Abstract

Machine Learning approaches used in the context of agents (like Reinforcement Learning) commonly result in weighted state-action pair representations (where the weights determine which action should be performed, given a perceived state). The weighted state-action pairs are stored, e. g., in tabular form or as approximated functions which makes the learned knowledge hard to comprehend by humans, since the number of state-action pairs can be extremely high. In this paper, a knowledge extraction approach is presented which extracts compact and comprehensible knowledge bases from such weighted state-action pairs. For this purpose, so-called Hierarchical Knowledge Bases are described which allow for a top-down view on the learned knowledge at an adequate level of abstraction. The approach can be applied to gain structural insights into a problem and its solution and it can be easily transformed into common knowledge representation formalisms, like normal logic programs.

1 Motivation

Machine Learning (ML) approaches like Reinforcement learning (RL) which are used in the context of learning agents, work quite well on many different tasks for decades now. But there is still a lack in understanding *what* was learned and *how* a learning agent solves a previously learned task. This paper tackles the issue of extracting knowledge on multiple levels of abstraction from weighted state-action pair representations that are learned in the context of agents. The resulting knowledge bases offer a compact view on the learned knowledge on different levels of abstraction, such that the knowledge can be inspected at an appropriate degree of detail. The main contributions of the paper are:

- A formal definition of *Hierarchical Knowledge Bases* (HKBs) with multiple levels of abstraction, which allows the inspection of the contained knowledge in a top-down manner on an adequate degree of detail.
- An improved knowledge extraction algorithm, based on our preliminary work (Apeldoorn and Kern-Isberner 2016), which is able to extract such knowledge bases faster from sub-symbolic weighted state-action pair representations (independent of the underlying ML approach used for learning, given that it results in a weighted state-action pair representation).

2 Related Work

In (Sun 2002), two extraction approaches are proposed to gain simple rules or plans from RL. However, this approach does not extract an HKB with multiple levels of abstraction, as will be done here, with the possibility to inspect knowledge at different levels of abstraction. In (Leopold, Kern-Isberner, and Peters 2008) RL was combined with belief revision to support the learning process of an agent and it was shown on a object recognition task that the considered agent can benefit from the belief revision mechanism and it is also possible to incorporate additional background knowledge. However, the focus of (Leopold, Kern-Isberner, and Peters 2008) lies on the incorporation of the two paradigms rather than on the possibility of making the learned knowledge explicit.

We build on our preliminary work (Apeldoorn and Kern-Isberner 2016), where we showed the performance gain of learning agents by incorporating HKBs extracted from a sub-symbolic representation during the learning process. The knowledge extraction algorithm used in (Apeldoorn and Kern-Isberner 2016) suffered from a computational drawback which will be improved here by incorporating ideas of the APRIORI algorithm (Agrawal et al. 1996).

3 Hierarchical Knowledge Bases

3.1 Preliminaries

As a preliminary for the following sections, we consider an agent which is learning a task by acting autonomously in an unknown environment. The agent is equipped with n sensors through which it can perceive its current state in the environment. The agent is able to perform actions from a predefined action space and can furthermore perceive, whether or not the performed actions were good (e. g., in form of a numeric reward).

More formally, in such a representation, a state s is an element of a multi-dimensional state space $\mathbb{S} = \mathbb{S}_1 \times ... \times \mathbb{S}_n$ where n is the number of the agent's sensors (through which the agent is able to perceive its state in the environment) and every $\mathbb{S}_i$ is a set of possible sensor values of the corresponding sensor. Furthermore, the agent selects actions from a predefined action set $\mathbb{A}$ and the learned weights are stored in a multi-dimensional matrix $\hat{Q} = (q_{s_1,...,s_n,a})$ with $s_i \in \mathbb{S}_i$ and $a \in \mathbb{A}$. The weights can be learned by diverse ML approaches, provided that the learning approach converges such that given a state, the highest weight determines the best action to be selected (i. e., $a^{\max}_{s_1,...,s_n} = \arg\max_{a' \in \mathbb{A}} q_{s_1,...,s_n,a'}$).

3.2 Definition of HKBs

This section introduces the concept of Hierarchical Knowledge Bases (HKBs) and provides the corresponding definitions. The HKBs consist of rules which are organized on different levels of abstraction. To be able to define these rules, two different kinds of states and two different kinds of rules will be distinguished:

Definition 1 (Complete States/Partial States) *A complete state is a conjunction $s := s_1 \wedge ... \wedge s_n$ of all values s_i currently perceived by an agent's sensors, where n is the number of sensors (and every perceived sensor value $s_i \in \mathbb{S}_i$ of the corresponding sensor value set $\mathbb{S}_i$ is assumed to be a fact in the agent's current state). A partial state is a conjunction $s := \bigwedge_{s' \in S} s'$ of a subset $S \subset \{s_1, ..., s_n\}$ of the sensor values of a complete state.*

Definition 2 (Complete Rules/Generalized Rules) *Complete rules and generalized rules are of the form $p_\rho \Rightarrow a_\rho\ [w_\rho]$, where p_ρ is either a complete state (in case of an complete rule) or a partial state (in case of a generalized rule), the conclusion $a_\rho \in \mathbb{A}$ is an action of an agent's action space $\mathbb{A}$ and $w_\rho \in [0, 1]$ is the rule's weight.*[1]

Thus, complete rules map complete states to actions and generalized rules map partial states to actions. An HKB can now be defined as follows:

Definition 3 (Hierarchical Knowledge Base) *A Hierarchical Knowledge Base (HKB) is an ordered set $\mathcal{KB} := \{R_1, ..., R_{n+1}\}$ of $n + 1$ rule sets, where n is the number of sensors (i. e., the number of state space dimensions). Every set $R_{i<n+1}$ contains generalized rules and the set R_{n+1} contains complete rules, such that every premise $p_\rho = \bigwedge_{s \in S_\rho} s$ of a rule $\rho \in R_i$ is of length $|S_\rho| = i - 1$.*

According to Definition 3, the set R_1 contains the most general rules (with empty premises) and the set R_{n+1} contains the most specific (i. e., complete) rules.

For the relations of rules, the terms of *exception* and *needed exception* are used in the following:

Definition 4 (Exception/Needed Exception) *A rule $\rho \in R_{j>1}$ 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 $w_\upsilon > w_\tau$.*

[1]Note that in (Apeldoorn and Kern-Isberner 2016), complete rules are called *elementary rules*.

4 Knowledge Extraction

4.1 Extraction Algorithm

To be able to extract rule-based symbolic knowledge from a sub-symbolic representation on multiple levels of abstraction, adequate *representation criteria* have been defined in (Apeldoorn and Kern-Isberner 2016), following the idea of a human, explaining a previously learned task to another person. Usually, one is interested in a compact representation of the knowledge which explains how to solve a previously learned task (i. e., how to get from a starting state to a solution). Considering only the relevant weights in this sense can be easily achieved by letting the agent store the best state-action sequence which was found according to the weights contained in $\hat{Q}$ as a set $\mathcal{SA}^{\hat{Q}} = \{(\mathbf{s}_1, a_1), ..., (\mathbf{s}_m, a_m)\}$ (with every $\mathbf{s}_i \in \mathbb{S}_1 \times ... \times \mathbb{S}_n$ and every $a_i = \arg\max_{a' \in \mathbb{A}} \hat{q}_{\mathbf{s}_i,a'}$).

Following these ideas, the knowledge extraction algorithm takes a set of state-action pairs $\mathcal{SA}^{\hat{Q}}$ (based on the weights contained in $\hat{Q}$) as input and returns an HKB $\mathcal{KB}^{\hat{Q}}$ which reflects the knowledge contained in $\hat{Q}$ by performing the following steps:

1. *Initial creation of rule sets:* In the first step, the multiple abstraction levels $R_1, ..., R_{n+1}$ of the knowledge base are initially filled with rules. Since the number of possible premises of the rules grows exponentially in the number of sensors, an adequate preselection of the rules will be done here using adapted ideas from the APRIORI algorithm (Agrawal et al. 1996). Details on the adaption of the APRIORI algorithm used here will be provided later in Section 4.2.
2. *Removal of worse rules:* 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).
3. *Removal of worse more specific rules:* In all sets $R_{j>1}$, a rule $\rho \in R_j$ with premise $p_\rho = \bigwedge_{s \in S_\rho} s$, 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 $S_\sigma \subset S_\rho = \{s_1, ..., s_{j-1}\}$ and weight $w_\sigma \geq w_\rho$.
4. *Removal of too specific rules:* In all sets R_j, a rule $\rho \in R_{j>1}$ with premise $p_\rho = \bigwedge_{s \in S_\rho}$ 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 $S_\sigma \subset S_\rho = \{s_1, ..., s_{j-1}\}$ and if ρ is not a *needed exception* to a rule $\tau \in R_{j-1}$.
5. *Optional filter step:* Optionally, filters may be applied to filter out further rules which are helpful to explain the knowledge contained in $\hat{Q}$ through $\mathcal{SA}^{\hat{Q}}$, but which are not needed for reasoning later.

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.

4.2 Adapting APRIORI

An adaption of the APRIORI algorithm (Agrawal et al. 1996) is used in the first step of the knowledge extraction algorithm (Section 4.1) to initially fill the rules sets $R_1, ..., R_{n+1}$. Given a set $\mathcal{SA}^{\hat{Q}}$ of (best) state-action pairs, the adapted APRIORI starts with short premises having a minimum support $supp_{\min}$ (i. e., those partial states that are involved to some degree in $\mathcal{SA}^{\hat{Q}}$). The premises are then successively extended to longer premises by keeping only those which are still having at least the minimum support of $supp_{\min}$. The support of a premise p_ρ with corresponding (ordered) premise set $S_\rho \subseteq \{s_1, ...s_n\}$ is calculated as

$$supp(S_\rho) := \frac{|\{(\mathbf{s}, a) \in \mathcal{SA}^{\hat{Q}} \mid S_\rho \subseteq \mathbf{s}\}|}{|\mathcal{SA}^{\hat{Q}}|} \tag{1}$$

Furthermore, the weight w_ρ of a corresponding rule ρ of the form $p_\rho \Rightarrow a_\rho\ [w_\rho]$ with premise set S_ρ is calculated as the confidence

$$conf(\rho) = \frac{|\{(\mathbf{s}, a) \in \mathcal{SA}^{\hat{Q}} \mid S_\rho \subseteq \mathbf{s}, a_\rho = a\}|}{|\{(\mathbf{s}, a) \in \mathcal{SA}^{\hat{Q}} \mid S_\rho \subseteq \mathbf{s}\}|} \tag{2}$$

More detailed, the adapted APRIORI takes a set of state-action pairs $\mathcal{SA}^{\hat{Q}}$ as input and outputs an initial set of rule sets with potentially relevant rules $R^{\hat{Q}} = \{R_1, ..., R_{n+1}\}$ by proceeding as follows:

1. Create set R_1 and add all rules ρ with an empty premise p_ρ, $a_\rho \in \mathbb{A}$ and $w_\rho = conf(\rho) > 0$.
2. Create a set of premise sets $\mathcal{S}_1$ and add all (ordered) premise sets S_ρ of length $|S_\rho| = 1$ with support $supp(S_\rho) \geq supp_{min}$.
3. Create set R_2 and add for all premise sets $S_\rho \in \mathcal{S}_1$ all rules ρ with $p_\rho = \bigwedge_{s \in S_\rho}$, $a_\rho \in \mathbb{A}$ and $w_\rho = conf(\rho) > 0$.
4. Set $k := 2$.
5. Create the set $\mathcal{S}_k$ of premise sets of length k. Combine every two premise sets $S_\rho, S_\sigma \in \mathcal{S}_{k-1}$ having the first $k-1$ elements in common to create a new premise set $S_\tau = S_\rho \cup S_\sigma$.[2] Add the new combined premise set S_τ to $\mathcal{S}_k$ if
   - all $(k-1)$-elementary subsets of S_τ occur in sets of $\mathcal{S}_{k-1}$ and,
   - $supp(S_\tau) \geq supp_{\min}$.
6. Create set R_{k+1} and add for all premise sets $S_\rho \in \mathcal{S}_k$ all rules ρ with $p_\rho = \bigwedge_{s \in S_\rho}$, $a_\rho \in \mathbb{A}$ and $w_\rho = conf(\rho) > 0$.
7. Set $k := k + 1$.
8. If $k \leq n$, continue with step 5.

[2]Note that an additional performance gain may be achieved here in practice since (in contrast of the original APRIORI-algorithm) only those pairs of sets that do not have any sensor values of the same sensor value set in common have to be considered for combination.

After performing these steps, the rule sets contained in $R^{\hat{Q}}$ are initially filled with preselected rules that could potentially be relevant for the knowledge to be extracted, given a minimum support of $supp_{\min}$.

5 Examples

5.1 Example Scenarios

Three example scenarios are provided, where an agent (e. g., a robot) based on the agent model from Section 3.1 learned to get from a starting point A to a target point B.[3] Figure 1 shows the scenarios with learned policies (indicated by the arrows) together with the corresponding extracted knowledge bases.

5.2 Extraction Example

In this section, the knowledge extraction algorithm from Section 4 is applied to Example 2 from Figure 1:

1. Starting from the learned state-action pairs $\mathcal{SA}^{\hat{Q}} = \{((x_0, y_0), \text{North}), ((x_0, y_1), \text{East}), ..., ((x_7, y_1), \text{East}), ((x_7, y_1), \text{South})\}$, the adaption of the APRIORI algorithm (Section 4.2) is performed with $supp_{\min} = 0$ to initially fill the knowledge base $\mathcal{KB}^{\hat{Q}}$ with potential rules. After that, $\mathcal{KB}^{\hat{Q}}$ contains the following rules (rules to be removed in the *subsequent* step are marked by * in the following):

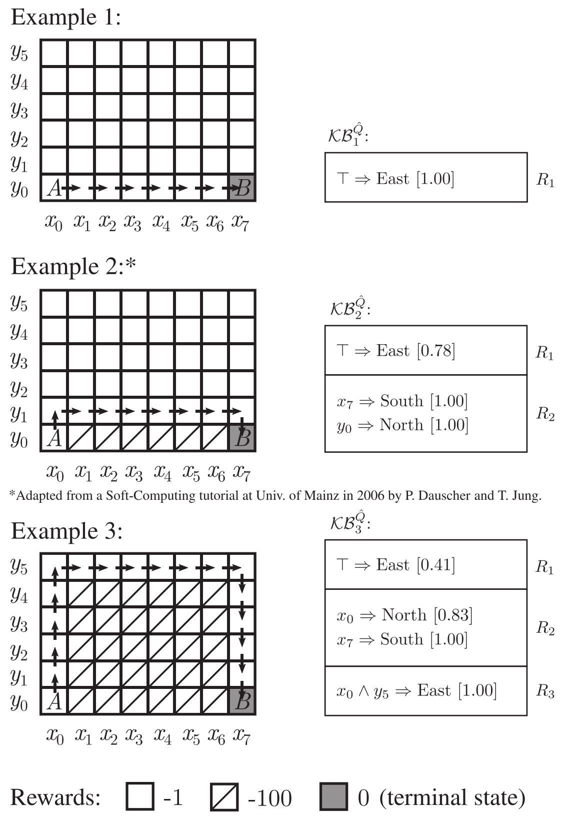
$\top \Rightarrow$ East [0.778]	
*$\top \Rightarrow$ South [0.111]	
*$\top \Rightarrow$ North [0.111]	
$y_0 \Rightarrow$ North [1.0]	$x_2 \Rightarrow$ East [1.0]
$y_1 \Rightarrow$ East [0.875]	$x_3 \Rightarrow$ East [1.0]
*$y_1 \Rightarrow$ South [0.125]	$x_4 \Rightarrow$ East [1.0]
$x_0 \Rightarrow$ East [0.5]	$x_5 \Rightarrow$ East [1.0]
*$x_0 \Rightarrow$ North [0.5]	$x_6 \Rightarrow$ East [1.0]
$x_1 \Rightarrow$ East [1.0]	$x_7 \Rightarrow$ South [1.0]
$x_0 \wedge y_0 \Rightarrow$ North [1.0]	$x_6 \wedge y_1 \Rightarrow$ East [1.0]
$x_7 \wedge y_1 \Rightarrow$ South [1.0]	$x_5 \wedge y_1 \Rightarrow$ East [1.0]
$x_2 \wedge y_1 \Rightarrow$ East [1.0]	$x_4 \wedge y_1 \Rightarrow$ East [1.0]
$x_1 \wedge$ y1 $\Rightarrow$ East [1.0]	$x_3 \wedge y_1 \Rightarrow$ East [1.0]
$x_0 \wedge y_1 \Rightarrow$ East [1.0]	

2. After removing the *worse rules*, $\mathcal{KB}^{\hat{Q}}$ contains the following rules:[4]

$\top \Rightarrow$ East [0.778]	
$y_0 \Rightarrow$ North [1.0]	$x_3 \Rightarrow$ East [1.0]
$y_1 \Rightarrow$ East [0.875]	$x_4 \Rightarrow$ East [1.0]
*$x_0 \Rightarrow$ East [0.5]	$x_5 \Rightarrow$ East [1.0]
$x_1 \Rightarrow$ East [1.0]	$x_6 \Rightarrow$ East [1.0]
$x_2 \Rightarrow$ East [1.0]	$x_7 \Rightarrow$ South [1.0]
*$x_0 \wedge y_0 \Rightarrow$ North [1.0]	*$x_6 \wedge y_1 \Rightarrow$ East [1.0]
*$x_7 \wedge y_1 \Rightarrow$ South [1.0]	*$x_5 \wedge y_1 \Rightarrow$ East [1.0]
*$x_2 \wedge y_1 \Rightarrow$ East [1.0]	*$x_4 \wedge y_1 \Rightarrow$ East [1.0]
*$x_1 \wedge$ y1 $\Rightarrow$ East [1.0]	*$x_3 \wedge y_1 \Rightarrow$ East [1.0]
$x_0 \wedge y_1 \Rightarrow$ East [1.0]	

[3]To learn the optimal policy, a standard *Q-Learning* approach (Watkins 1989) was used here in every scenario.

[4]In case of equivalent rules (i. e., rules having the same premise and an equal weight), only one of these rules need to be kept. Here the selection is done according to the lexicographic order of the conclusions.



Example 1:
y_5
y_4
y_3
y_2
y_1
y_0
A
B
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7
$\mathcal{KB}_1^{\hat{Q}}$:
$\top \Rightarrow$ East [1.00]
R_1
Example 2:*
$\mathcal{KB}_2^{\hat{Q}}$:
$\top \Rightarrow$ East [0.78]
R_1
$x_7 \Rightarrow$ South [1.00]
$y_0 \Rightarrow$ North [1.00]
R_2
Example 3:
$\mathcal{KB}_3^{\hat{Q}}$:
$\top \Rightarrow$ East [0.41]
R_1
$x_0 \Rightarrow$ North [0.83]
$x_7 \Rightarrow$ South [1.00]
R_2
$x_0 \wedge y_5 \Rightarrow$ East [1.00]
R_3
Rewards: □ -1 ▨ -100 ■ 0 (terminal state)

*Adapted from a Soft-Computing tutorial at Univ. of Mainz in 2006 by P. Dauscher and T. Jung.

Figure 1: Example Scenarios with Extracted HKBs

3. After removing the *worse (or equivalent) more specific rules*, the following rules are remaining in $\mathcal{KB}^{\hat{Q}}$:

$\top \Rightarrow$ East [0.778]	
$y_0 \Rightarrow$ North [1.0]	$*x_4 \Rightarrow$ East [1.0]
$*y_1 \Rightarrow$ East [0.875]	$*x_5 \Rightarrow$ East [1.0]
$*x_1 \Rightarrow$ East [1.0]	$*x_6 \Rightarrow$ East [1.0]
$*x_2 \Rightarrow$ East [1.0]	$x_7 \Rightarrow$ South [1.0]
$*x_3 \Rightarrow$ East [1.0]	
$*x_0 \wedge y_1 \Rightarrow$ East [1.0]	

4. The final $\mathcal{KB}^{\hat{Q}}$ after removing the *too specific rules*:

$\top \Rightarrow$ East [0.778]
$y_0 \Rightarrow$ North [1.0]
$x_7 \Rightarrow$ South [1.0]

Such an HKB can also be transformed, e. g., to a *normal logic program*, by considering the rules on each level $R_{j>1}$ as exceptions of the rules of level R_{j-1}. In this example, this will result in the program: $\mathcal{P}^{\hat{Q}} = \{\text{East} \leftarrow$ *not* y_0, *not* $x_7., \text{North} \leftarrow y_0., \text{South} \leftarrow x_7.\}$.[5]

5.3 Interpretation of the Results

The knowledge contained in the extracted HKBs in Figure 1 can be read in a top-down manner: In the case of Example 1, the agent simply learned to go east, which is reflected by the only provided abstraction level in $\mathcal{KB}_1^{\hat{Q}}$. In the case of Examples 2 and 3, in general, the agent also learned to go east (since the target is still located in the east in both scenarios). This can easily be seen on the top-level of $\mathcal{KB}_2^{\hat{Q}}$ and $\mathcal{KB}_3^{\hat{Q}}$, respectively. However, when interested in more details, one can have a closer look on the lower levels of abstraction: In case of Example 2, the agent learned that when perceiving y_0 it should go north and when perceiving x_7 it should go south to avoid the highly negative rewarded area in the south. In case of Example 3, an additional third level of abstraction is provided in $\mathcal{KB}_3^{\hat{Q}}$, where it can be seen that the agent learned to go to east when perceiving x_0 and y_5 as an exceptional case.

In general, every level $R_{j>1}$ can be considered to contain the exceptions of the level R_{j-1}, which allows to comprehend what was learned on the respective level of detail.

6 Conclusion and Future Work

This paper presented a knowledge extraction approach which allows to comprehend what is learned through ML techniques in the context of learning agents at different levels of abstraction. For this purpose, the concept of HKBs was described and an improved extraction algorithm was introduced which is able to extract an HKB faster from a state-action pair sequence (which was retrieved before from a weighted state-action pair representation): We were able to overcome a computational bottleneck of a preliminary version of the algorithm used in (Apeldoorn and Kern-Isberner 2016) by introducing a variant of the APRIORI algorithm (Agrawal et al. 1996) to the extraction algorithm.

Future research could, e. g., comprise a closer incorporation of the adapted APRIORI approach with the removal strategies of the extraction algorithm.

References

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. 1996. *Fast Discovery of Association Rules*. Advances in Knowledge discovery and Data Mining. Cambridge, MA, USA: MIT Press. 307–328.

Apeldoorn, D., and Kern-Isberner, G. 2016. When should learning agents switch to explicit knowledge? In Benzmüller, C.; Sutcliffe, G.; and Rojas, R., eds., *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, volume 41 of *EPiC Series in Computing*, 174–186. EasyChair Publications.

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

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

Watkins, C. 1989. *Learning from Delayed Rewards*. England: University of Cambridge.

[5]Thanks to Corinna Krüger for the exchange of ideas on this.