ADC: Concept Discovery and Learning of Actions for Unknown Environments

Ana C. Tenorio-González CATANACE17@INAOEP.MX
Eduardo F. Morales EMORALES@INAOEP.MX
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro 1, Sta. Ma. Tonantzintla, Puebla,72840, Mexico
http://www.inaoe.mx

Abstract

How to represent adequate knowledge for solving a problem is a challenging task in the design of artificial intelligent agents. Ideally we would like an agent to create its own representations. In this paper, we describe the ADC (Automatic Discovery of Concepts) algorithm for learning relational concepts and actions to represent states of the environment and solve tasks, using inductive logic programming and reinforcement learning. Our experiments show that an intrinsic motivation function and automatically learning new representations can reduce the learning time and produce useful concepts for solving a task.

1. Introduction

Artificial intelligent agents should perceive and interact with its surrounding world. An autonomous agent should be able to represent, learn and reason about its environment using its basic background knowledge (BK) and experience (Russell & Norvig, 2008). It should adapt its behavior automatically when changes in the environment occur, improving its skills increasing its independence (from human programmers) and precision, and also reducing the time in the execution of tasks.

The definition of abstract concepts can be used to simplify the representation of the world and improve the machine learning (ML) process. On the other hand, the learning of new actions can be used to acquire new skills. In this paper a concept is understood as a set of objects, attributes and relations among them. In ML, Inductive Logic programming (ILP) (Muggleton, 1991) has been successful in the learning of concepts because of its powerful representation based on logic and its robust inductive methods. ILP can also provide mechanisms to add new vocabulary to the initial BK as demand-driven predicate invention (Muggleton & Road, 1994; Stahl, 1996; Wrobel, 1994; Li et al., 2008; Stanley & Domingos, 2007; Kosmerlj et al., 2011). Reinforcement learning (RL) (Sutton & Barto, 1998) has been useful to acquire skills from experience when external rewards are provided. Recently, intrinsic motivation functions (Merrick & Mahler, 2013; Baldassarre & Mirolli, 2013) have been used in RL to guide the learning process automatically (Singh et al., 2005; Zhang & Weng, 2002; Merrick & Mahler, 2006; 2013).

In this paper, we describe ADC, a method for intelligent agents to automatically learn relational concepts and actions during the exploration of unknown environments. Our approach is designed to discover and collect examples of potential concepts, using ILP, that are used to represent the environment, while actions are learned through RL with intrinsic motivation based on an asymmetrical Wundt's curve (Berlyne, 1960). The main contributions of this research are: a) Automatic, incremental and simultaneous learning of relational concepts by predicate invention, construction of descriptions of states based on those relational concepts and learning of actions over that state representation; and b) Automatic intrinsic guide for exploration of environments in a concept discovery framework. In the next section related work is described, Section 3 introduces the proposed ADC algorithm, Section 4 describes experiments where we illustrate the functionality of ADC with a mobility domain, and Section 5 presents conclusions and future research work.

2. Related work

Abstraction in RL has been performed in different components of RL algorithms, as value functions, actions and state representations. Much of work has been done in temporal abstractions for actions, as the discovery of useful subgoals based on options (Sutton et al., 1999). Methods based on options to identify subgoals that helps to reach new, strategic, parts of the environment have been

Abstraction in RL Workshop at the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

proposed using different techniques as relative novelty (based on novelty calculated by number of visits to close states) (Barto & Simsek, 2004) or diverse density (detection of regions with most successful trajectories (sequences of states) to reach other strategic states) (McGovern & Barto, 2001). State abstractions have been also performed with different techniques, safe state abstractions have been learned based on decomposition of value functions in additive parts based on expected rewards (Andre & Russell, 2002), deep neural networks have been used to deal with high dimensional learning (Kulkarni et al., 2016), other works have been used abstractions to deal with continuous spaces and learn functional concepts by unsupervised clustering (Mousavi et al., 2015), and ILP with relational learning in RL has been used to create relational representations of states, actions and specific goals arising relational reinforcement learning (RRL) (Džeroski et al., 2001).

Regarding to this last technique, RRL allows to use more expressive representations because of using ILP, reduce search space, create structural representations and make easier the reuse of basic knowledge acquired in simple tasks in other more complex (Tadepalli et al., 2004). Reinforcement learning over abstractions of space has been addressed defining relational properties of the state space and relational actions over the states, building hierarchical representations (Morales, 2004). Also, RRL with guidance has been addressed based on human controllers and demonstrations applied to games and robotics (Driessens & Džeroski, 2004; Martínez et al., 2015). RRL has been used in extensions of RL as Inverse RRL (Munzer et al., 2015), where a reward function which explains an expert policy should be learned, and Interactive RL (Nickles & Rettinger, 2014), where an agent interacts with human users to understand concepts.

Our proposed method described in this paper is most related to RRL abstraction techniques, since ADC represents the knowledge as logic clauses. It automatically creates an incremental and hierarchical representation of the environment, where states are represented by relational concepts learned by ILP, over which policies are learned by RL.

3. ADC: Concept discovery and learning of actions

The learning of relational concepts and actions during the exploration of unknown environments, involves several challenges, mainly because, agents should be as autonomous as possible. Agents should be able to drive their own learning process, figuring out how to learn concepts and actions automatically. The exploration can be exploited to learn to perform tasks, identify what situations cause the learning of new concepts, and also, evaluate the benefits of introducing that new knowledge.

With ADC (for Automatic Discovery of Concepts), an agent learns relational concepts and actions through subgraph discovery, inductive logic programming and reinforcement learning. The acquired knowledge represented by relational concepts helps to simplify the representation of the world and the reasoning process. The acquired knowledge can be reused in different tasks and can also be easily understood by humans. The relational concepts are used to represent states of the world for learning actions. The learning of actions helps to evaluate the utility of the relational concepts. Moreover, learning of actions through experience helps to improve the skills of an agent for the accomplishment of tasks. The next subsections describe how these processes are performed by ADC.

3.1. How to represent knowledge in concept discovery

In our approach, ADC represents all knowledge provided and acquired using Horn clauses (concepts, definitions of states and actions). First-order logic (FOL) provides robust reasoning mechanisms (as induction) and a representation easier to understand than other common representations (e.g., function approximators, neural networks, vectors). For the exploration of an environment, an agent should start with some basic knowledge. In ADC, initial background knowledge is formed by predicates representing simple objects, attributes and relations, and also, some basic actions.

3.2. How to discover potential concepts

In ADC, demand-driven predicate invention with an automatic collection of examples of potential concepts is performed. The agent uses the basic actions to explore the environment. During the exploration, it uses its background knowledge and its sensors to identify elements of the environment and incrementally construct a graph-based representation. Each predicate identified in the environment about an object and/or attribute is mapped as a vertex, and a relation is mapped as an edge in the graph. The growth of this graph is limited to speed up its subsequent processing. ADC assumes that frequent representations identified in the environment can be potentially useful new concepts.

If some elements are identified more than once in the environment, they are potentially relevant features of the environment that can provide significant information about how to define that place. In ADC, the agent performs a frequent substructure search in the constructed graph-based representation of the environment, through the system Subdue (Holder et al., 1994). Each discovered frequent sub-graph is an example of a potential concept. The frequent subgraphs are represented also by a FOL clause. ADC clusters these examples based on two similarity measures based on relevant elements and inexact matching.

With the first measure, a sub-graph g_i is added to a group C_k if certain percentage of the objects and relations of the clause cl_{g_i} of g_i can also be found in the common objects and relations R_k found in the instances of that group. This measure allows to cluster sub-graphs which share the same objects and relations although their structures and sizes could be different. Let cl_{g_i} be a new instance clause and L_g be its set of literals. Let C_k be a group of instances and R_k be the set of common literals in the clauses of the instances in C_k according to Equation 3. A sub-graph g_i is added to a group C_k, if a large proportion of the literals L_g in cl_{g_i} are common to R_k (see Equation 1) and if a large proportion of the most common literals in C_k (i.e., R_k) are common to the literals in cl_{g_i} (see Equation 2).

$$g_i \in C_k \text{ iff } \frac{|L_g \cap R_k|}{|L_g|} \geq th_1 \tag{1}$$

$$g_i \in C_k \text{ iff } \frac{|L_g \cap R_k|}{|R_k|} \geq th_2 \tag{2}$$

$$l \in R_k \text{ iff } \sum_{i=1}^{S_{C_k}} |\{l\} \cap L_{g_i}| \geq \frac{\sum_{j=1}^{|L_{C_k}|} \sum_{i=1}^{S_{C_k}} |\{l_j\} \cap L_{g_i}|}{|L_{C_k}|} \tag{3}$$

where S_{C_k} = number of sub-graphs of C_k, L_{C_k} = set of literals of all graphs in C_k and th_1, th_2 = threshold values. Let $N(l)$ = name of literal l and L a set of literals:

$$\{l\} \cap L = \begin{cases} l & \text{if } N(l) = N(k) \text{ and } k \in L \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

The second measure takes into account the size and structure of the sub-graphs. A sub-graph g_i is added to a group of sub-graphs C_k if the average cost of structural changes to make $g_i = g_k \in C_k$ for all the sub-graphs in C_k is smaller than a threshold value (see Equation 5). The $matchcost(g_i, g_k)$ corresponds to the inexact matching measure used in Subdue, this function calculates the minimum cost to convert one graph to another, considering a cost based on deletion, insertion and substitution of vertexes and edges, through a branch-and-bound tree search algorithm (Holder et al., 1994).

$$g_i \in C_k \text{ iff } \frac{\sum_{j=1}^{S_{C_k}} matchcost(g_i, g_j)}{S_{C_k}} \leq th_3 \tag{5}$$

When an example cannot be added to a group a new group is created with this example. As each group have a set of similar examples of a potential concept, a definition is induced by the ILP system Progol (Muggleton, 1995), using the grouped examples as positive examples and a set of negative examples. The negative examples set are created using the examples gathered in other groups and producing synthetic examples creating variations of the positive examples of the group where the definition is being induced. These variations are produced by deleting relations from original positive examples. This process to produce negative examples avoids over-generalization during the induction. Each $cl_g \in Ex^+$ has the form $h_g \leftarrow O, R$ where h_g is the head of the clause, $O = \{o_1, o_2, \ldots, o_n\}$ is the set of literals in the clause that describes objects, and $R = \{r_1, r_2, \ldots, r_m\}$ is the set of literals that describes relations among objects. Artificial negative examples are (if they are not members of Ex^+):

$h_g \leftarrow O$ (a clause without relations)
$h_g \leftarrow R$ (a clause without explicit objects)
$h_g \leftarrow O, R' | R' \subset R$ (clauses with a smaller number of relations

The formation of groups of examples is incremental during the learning process. The definitions of the learned concepts are used to learn new concepts (hierarchical concepts), added to the current background knowledge and organized in a lattice.

3.3. How to recognize the environment with concepts to learn actions

Concepts learned by ADC are used to define states of the environment in ADC. Each time the agent performs an action, it checks which of the learned concepts are present in its current situation. If the current situation is not a known state, the agent constructs a new clause definition for such state as a conjunction of the identified concepts. If the state is known and additional learned concepts can be identified in that situation, ADC updates the definition of the state. New states are added to the background knowledge of the agent. So, the agent constructs an incremental representation of the environment, that can be updated during the exploration of an environment.

3.4. How to guide the exploration and learn actions

In ADC, the learning of actions is performed with traditional rewards (r_e) using Sarsa$-\lambda$ with eligibility of traces (Sutton & Barto, 1998). The agent performs basic actions to explore the environment. Actions are learned for the states defined by concepts. Actions are defined as STRIPS-like operators (Fikes & Nilsson, 1971), with preconditions, actions and post-conditions. Also, an agent using ADC drives its own exploration of the environment using an intrinsic motivation function. The agent is self-attracted to few known situations and repelled by well-known states by intrinsic motivation. This motivation is based on the Wundt's curve (Berlyne, 1960) but defined asymmetrically to reduce the time of exploration (see

Equation 6), and given as a reward r_i, $r = r_e + r_i$,

$$r_i = \frac{maxReward}{(1 + \exp(C_1 * (-novelty + minNovelty_R)))} + \frac{maxPunish}{(1 + \exp(-C_2 * (novelty - minNovelty_P)))} \tag{6}$$

where $maxReward$ = maximum value for reward, $maxPunish$ =maximum value for punishment, $minNovelty_R$ =minimum value of novelty for reward, $minNovelty_P$ =minimum value of novelty for punishment, C_1 and C_2 are the parameters to define the Wundt's curve, $C_1 < C_2$ defines the asymmetry of the curve. The highest value of novelty ($novelty = maxValueNovelty$) is given when a new state is reached, otherwise, $novelty = numVisits_s * noveltyMostVisited_s / maxNumVisits_S$. $numVisits_s$ = number of views of the current state during exploration (initially 0). $maxNumVisits_S$ = number of views of the most viewed state (among all the states) until then (initially 0), for all the states, and $numVisits_s$ = value of the most viewed state. $noveltyMostVisited_s$ = value for novelty of the most viewed state in the past. A state stops receiving intrinsic rewards when all possible actions have been performed in this state, or when the number of visits $numVisits_s$ is larger than the number of the current actions multiplied by a threshold value.

3.5. Scope and limitations of ADC

ADC requires an initial background knowledge represented in first-order language, relative to the domain, to identify elements in the environment, explore and learn tasks. Continuous spaces can be discretized by the BK of ADC. The performance of ADC depends on: initial background knowledge and capability of the agent to recognize basic elements of the environment, presence or absence of frequent substructures that could been used for the same algorithm to identify the environment, size of the explored environment, size of the graphs constructed (number of vertexes and edges), number of discovered potential concepts, number of characterized and unidentifiable states, and time to perform actions. The inexact matching process used in the frequent sub-graph search for a graph 1 with *n* vertices and a graph 2 with *m* vertices, where $m \geq n$, has a complexity of $\mathcal{O}(n^{m+1})$, although its performance can be improved by the use of a branch-and-bound search algorithm (Cook & Holder, 1994). The growth of the graphs constructed by ADC is limited to speed up the learning in large spaces. It is expected that with large background knowledge and environments, the performance of ADC can be seriously affected.

4. Experiments

This section shows the results obtained using ADC in a domain of mobility of objects. A Nao (Aldebaran-Robotics, 2012) robot was used in a room ($3m^2$) formed by four walls simulated through Webots for Nao (Michel, 2004), see Figure 1. The robot explored the environment performing four basic actions provided in its initial background knowledge. States were identified by only four colors (green, blue, yellow and red) and relative positions between the objects and the robot. The initial background knowledge provided is shown in Table 1.The task of the robot was *Go to the red balls and discover the hidden green wall, pushing the balls.*

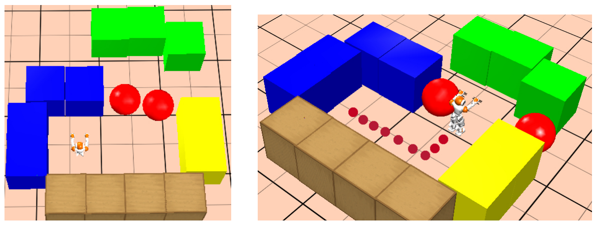

Figure 1. Environment of *Mobility of objects*: initial state and example of goal state. A simulated humanoid in a room with walls of boxes and movable balls.

The incremental representation of states based on concepts was compared against a traditional RL algorithm with predefined states provided by the user. We also evaluated the effectiveness of the intrinsic motivation function and biased actions in RL. The general parameters of ADC used in the experiments were: $th1 = 0.6$, $th2 = 0.7$, $th3 = 2.0$, $r_e = 10$, $\gamma = 0.9$, $\lambda = 0.9$, $\alpha = 0.1$, $\epsilon = 0.2$, $maxReward = 1$, $minNovelty_R = 0.3$, $maxPunish = -1$, $minNovelty_P = 1.2$, $noveltyMostVisited_s = 1.7$, $minNumVisits_{a-s} = 1.5$, $C_1 = 8$, $C_2 = 12$, $inc = 0.1$ and $dec = 0.2$. The values for thresholds of substructure discovery were chosen following the theory provided in (Holder et al., 1994) and (Cook & Holder, 1994), examples of working of the parameters in different databases can be found in (Cook & Holder, 1994). The thresholds for clustering were chosen to cluster elements in a same group with more than a half of similar components among them. The values for the RL parameters were chosen according to the experimental data and recommendations for traces of eligibility without replacement reported in (Singh & Sutton, 1996). Experiments were conducted on Debian Linux 7.1 (OS) with hardware based on Intel Core i7-3610QM 2.3GHz (CPU) and 6Gb of RAM at 1600MHz.

We performed two experiments (repeated 10 times each): (i) Use pre-defined states and actions and compared RL (SARSA with eligibility traces) with/without the intrinsic motivation function (PS: Extrinsic/Extrinsic + Intrinsic)

Table 1. Background knowledge provided to learn concepts about *Mobility of objects*.

Background knowledge
object/4 Object: instance of an object of the world with its attributes. Arguments: identifier of the instance of the object and its three attributes (color, previous position, current position). *robot/3* Object: agent/robot who is exploring the environment and its attributes. Arguments: identifier of the robot and its two attributes (previous position, current position).
in_front_of_me/2 Relation: a relative position (in front of) between an object of the world and the agent. *equal/2* Relation: approximate equality of two positions. Arguments: identifiers of two instances (object or robot).
different/2 Relation: approximate inequality of two positions. Arguments: identifiers of two instances (object or robot).
go_Right/2 (/0) Action: turning right (90°) and going forward (while obstacles are not perceived). *go_Left/2 (/0)* Action: turning left (90°) and going forward (while obstacles are not perceived). *go_Forward/2 (/0)* Action: going forward (while obstacles are not perceived). *push/2 (/0)* Action: moving the arms to push. Two arguments (/2): identifiers of two instances (robot) in two different time instants.

and (ii) allow to learn states and actions while exploring the environment (ADC) and compare RL with/without the intrinsic motivation function (Extrinsic/ADC: Extrinsic + Intrinsic).

The results obtained with predefined states are shown in Figure 2 and Table 2. The results obtained using ADC (incremental representation based on concepts) are shown in Figure 2 and Table 3. The total number of concepts learned with each configuration is shown, but also the number of general, singleton and hierarchical concepts are shown. General concepts have induced definitions, singleton concepts define groups with only one instance and hierarchical concepts have in their definitions other concepts previously learned. In unknown environments it is difficult to know *a priori* what number of concepts could be discovered or what concepts are useful. So, it is preferred a moderate number of concepts, with large numbers of general and hierarchical concepts, learned in short time. Also those concepts should be representative of the environment and useful to learn tasks.

Table 2. Average convergence time and number of episodes for each experiment of *Mobility of objects* with predefined states.

Experiment	Time (hr:min:secs) AVG (SD)	No. episode AVG (SD)
PS: Extrinsic (RL traditional)	03:34:09 (00:47:11)	17 (6)
PS: Extrinsic + Intrinsic	01:21:14 (00:18:01)	10 (2)

Table 3. Average convergence time and number of episodes for each experiment of *Mobility of objects* with ADC (*G=General, S=Singleton, H=Hierarchical* concepts).

Experiment	Time (hr:min:secs) AVG (SD)	No. episode AVG (SD)	No. states AVG (SD)	No. concepts {G, S, H} AVG (SD) {AVG (SD), AVG (SD), AVG (SD)}
Extrinsic (RL traditional)	06:44:54 (03:16:00)	21 (3)	12 (3)	168 (13) {27 (6), 142 (16), 145 (5)}
ADC: Extrinsic + Intrinsic	06:23:27 (01:41:00)	20 (6)	16 (5)	153 (1) {33 (10) 122 (11) 136 (2)}

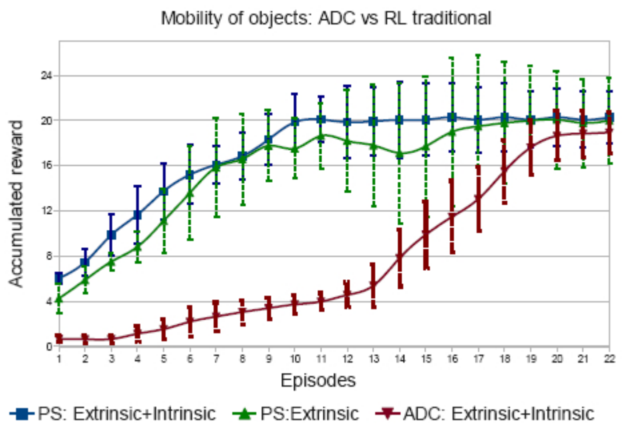

Figure 2. Learning curves with predefined and incremental states with the ADC algorithm with intrinsic motivation and traditional RL with/without intrinsic motivation in *Mobility of objects* with predefined states.

In predefined states, the best results are obtained when the intrinsic reward is added to the traditional RL algorithm. The intrinsic motivation helps to focus on constant exploration of few known states avoiding to revisit well known states and reducing the learning time by 2 hours (see Table 2). The novel strategy of exploration, designed to balance the concept discovery and the learning of actions in unknown environments simultaneously, seems not to affect the learning behavior obtaining better results than the traditional RL algorithm.

In the complete ADC (incremental representation based on concepts), the best results are also obtained when ADC uses intrinsic motivation to guide the exploration (according to the learning time, and consistency in number of concepts, states and episodes). The intrinsic motivation keeps the agent focused in learning both, concepts and behavior policy. Using intrinsic motivation allows a more consistent learning than with the traditional strategy of exploration (a lower standard deviation in number of concepts and learning time). Similar behavior policies, states and concepts could be learned by the two strategies of exploration. The learning times were greater than those obtained with predefined states. However, in these experiments, concepts were learned and states were characterized using those concepts automatically, with an accumulated reward and number of episodes similar to those obtained with predefined states.

Table 4. Mobility of objects (Extrinsic + Intrinsic). Learned action, state descriptions where actions were learned and concept definitions which are used to describe the states.

% The robot learned to go to the right when it was in front of blue walls, go to the left when it was in front of yellow walls and push red balls to discover a green wall. do_Action(4):-state(1),go_Right(),state(3),go_Left(),state(8), push(),state(9).
state(1):-c72(A,B,C,blue,D,E,F),c7(A,B,C,blue,D,E,F). state(3):-c59(A,B,C,yellow,D,E,F),c146(A,B,C,yellow,D,E,F). state(8):-c123(A,B,C,red,D,E,F). state(9):-c148(A,B,C,green,D,E,F).
c72(A,B,C,blue,D,E,F):-object(A,B,C,blue),robot(D,E,F), in_front_of_me(A,D),different(E,F). c7(A,B,C,blue,D,E,F):-object(A,B,C,blue),robot(D,E,F), in_front_of_me(A,D). c59(A,B,C,yellow,D,E,F):-object(A,B,C,yellow),robot(D,E,F), in_front_of_me(A,D),equal(B,C). c146(A,B,C,yellow,D,E,F):-object(A,B,C,yellow),robot(D,E,F), in_front_of_me(A,D). c123(A,B,C,red,D,E,F):-object(A,B,C,red),robot(D,E,F), in_front_of_me(A,D),equal(B,C),different(E,F). c148(A,B,C,green,D,E,F):-object(A,B,C,green),robot(D,E,F), in_front_of_me(A,D).

Table 4 shows examples of a behavior policy (a sequence of three actions and four states to reach the goal), states (sets of concepts involving the colored walls and balls, and the robot) and concepts (different relative positions between the robot and the colored objects) all of them automatically learned with ADC with intrinsic motivation (Extrinsic + Intrinsic). Four states were identified by the robot to accomplish the task describing: *the robot has reached a blue object, it is in front of him* (*state(1)* and concepts *c72,c7*), *the robot is in front of a yellow object* (*state(3)* and concepts *c59,c146*), *the robot has reached a red object, it is in front of him* (*state(8)* and concept *c123*) and *the robot is in front of a green object* (*state(9)* and concept *c148*). The robot learned to go to the balls and move them to find the green wall.

Table 5 shows additional useful concepts related with mobility learned by ADC aside from the learned concepts to solve the task. ADC could describe which objects were fixed and which were movable. The blue objects could not be pushed by the robot (*c16, c17*), so these objects could be labeled as fixed. The red objects could be pushed by the robot, so these objects could be labeled as movable (*c123, c129*).

Table 5. Mobility of objects (Extrinsic + Intrinsic). Examples of definitions of concepts learned by ADC. For presentation purposes, the number of arguments of previously learned concepts used in the definition of new concepts is reduced to one.

Mobility of objects	
Concepts	Description
c16(A,B,B,blue,C,D,D):- object(A,B,B,blue),robot(C,D,D), in_front_of_me(A,C), equal(B,B),equal(D,D).	*A robot is standing in front of a blue object*
c17(A,B):- c16(A),c16(B),push(A,B).	*A robot tries to push a blue object, the robot and the object do not change its position, the object is fixed*
c123(A,B,C,red,D,E,F):- object(A,B,C,red), robot(D,E,F), in_front_of_me(A,D), equal(E,F),different(B,C).	*A robot is in front of a red object, the red object changed its position*
c129(f0,f1):- c123(f0),c123(f1),push(f0,f1).	*A robot tries to push a red object, the robot does not change its position, the object changes it position, the object is movable*

5. Conclusions and future work

The ADC algorithm has been introduced for simultaneously learning concepts and actions in unknown environments. Experiments in a domain of mobility of objects were performed to illustrate the functionality of ADC. The results shown that a self-motivated agent is able to learn useful relational concepts, build state descriptions based on concepts and learn actions in this domain. It was shown that self-motivation, based on an asymmetrical Wundt's curve, accelerates and guides the learning process of the agent.

As future work, the representation of ADC can be improved to include functional symbols, negation and recursivity. A method to filter useful and not useful concepts can be included. Also we need to perform more tests and evaluate ADC in other more challenging environments with robust and more complex actions to facilitate the exploration (for reducing accumulated errors when performing actions and for speeding up the learning process). We also need to assess the sensitivity of the algorithm to its parameters.

References

Aldebaran-Robotics, 2012. URL www.aldebaran.com/en/humanoid-robot/nao-robot.

Andre, D. and Russell, S. State abstraction for programmable reinforcement learning agents. In *Eighteenth National Conference on Artificial Intelligence*, pp. 119–125, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0.

Baldassarre, G. and Mirolli, M. *Intrinsically motivated learning in natural and artificial systems*. SpringerLink : Bücher. Springer-Verlag Berlin Heidelberg, 2013. ISBN 9783642323751.

Barto, A. and Simsek, Ö. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 751–758. ACM Press, 2004.

Berlyne, D. *Conflict, Arousal and Curiosity*. McGraw-Hill, New York, 1960.

Cook, D. and Holder, L. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1(1):231–255, 1994. ISSN 1076-9757.

Driessens, K. and Džeroski, S. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57 (3):271–304, 2004. ISSN 1573-0565.

Džeroski, S., Raedt, L. De, and Driessens, K. Relational reinforcement learning. *Machine Learning*, 43(1):7–52, 2001. ISSN 1573-0565.

Fikes, R. and Nilsson, N. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pp. 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

Holder, L., Cook, D., and Djoko, S. Substructure discovery in the subdue system. In *Proceedings of The AAAI Workshop on Knowledge Discovery in Databases*, pp. 169–180. AAAI Press, 1994.

Kosmerlj, A., Bratko, I., and Zabkar, J. Embodied concept discovery through qualitative action models. *International journal of uncertainty, fuzziness and knowledge-based systems*, 19(3):453–475, 2011. ISSN 0218-4885.

Kulkarni, T., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *CoRR*, abs/1604.06057, 2016. URL http://arxiv.org/abs/1604.06057.

Li, N., Stracuzzi, D., and Langley, P. Learning conceptual predicates for teleoreactive logic programs. In *Proceedings of The Late-Breaking Papers Track at the Eighteenth International Conference on Inductive Logic Programming*, 2008.

Martínez, D., Alenya, G., and Torras, C. Relational reinforcement learning with guided demonstrations. *Artificial Intelligence*, 2015. ISSN 0004-3702.

McGovern, A. and Barto, A. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1.

Merrick, K. and Mahler, M. Modelling motivation as an intrinsic reward signal for reinforcement learning agents. *Machine Learning*, 2006.

Merrick, K. and Mahler, M. Novelty and beyond: towards combined motivation models and integrated learning architectures. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, pp. 235–259. Springer-Verlag Berlin Heidelberg, 2013. ISBN 978-3-642-32374-4.

Michel, O. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

Morales, E. F. Relational state abstractions for reinforcement learning. In *Proceedings of the ICML'04 Workshop on Relational Reinforcement Learning*, pp. 27–32, 2004.

Mousavi, A., Ahmadabadi, M. Nili, Vosoughpour, H., Araabi, B. N., and Zaare, N. Hierarchical functional concepts for knowledge transfer among reinforcement learning agents. *Iranian Journal of Fuzzy Systems*, 12: 99–116, 2015. ISSN 1735-0654.

Muggleton, S. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991. ISSN 1882-7055.

Muggleton, S. Inverse entailment and progol. *New Generation Computing*, 13(3-4):245–286, 1995. ISSN 0288-3635.

Muggleton, S. and Road, K. Predicate invention and utilisation. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:1–6, 1994.

Munzer, T., Piot, B., Geist, M., Pietquin, O., and Lopes, M. Inverse reinforcement learning in relational domains. In *International Joint Conferences on Artificial Intelligence*, pp. 3735–3741, Buenos Aires, Argentina, 2015.

Nickles, M. and Rettinger, A. Interactive relational reinforcement learning of concept semantics. *Machine Learning*, 94(2):169–204, 2014. ISSN 1573-0565.

Russell, S. and Norvig, P. *Inteligencia Artificial. Un enfoque moderno*. Pearson. Prentice Hall, 2008. ISBN 978-84-205-4003-0.

Singh, S. and Sutton, R. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1):123–158, 1996. ISSN 1573-0565.

Singh, S., Barto, A., and Chentanez, N. Intrinsically motivated reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems 17 (NIPS)*, pp. 1281–1288, MA, USA, 2005. MIT Press.

Stahl, I. Predicate invention in inductive logic programming. In Raedt, Luc De (ed.), *Advances in Inductive Logic Programming*, pp. 34–47. IOS Press, Ohmsha, Amsterdam, 1996.

Stanley, K. and Domingos, P. Statistical predicate invention. In *Proceedings of the 24th annual international conference on machine learning*, pp. 433–440, 2007.

Sutton, R. and Barto, A. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Sutton, R., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. ISSN 0004-3702.

Tadepalli, P., Givan, R., and Driessens, K. Relational reinforcement learning: An overview. In *Proceedings of the ICML'04 Workshop on Relational Reinforcement Learning*, pp. 1–9, 2004.

Wrobel, S. Concept formation during interactive theory revision. *Machine Learning*, 14(2):169–191, 1994.

Zhang, Y. and Weng, J. Novelty and reinforcement learning in the value system of developmental robots. In *Proceedings of the International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, pp. 47–55, Edinburgh, Scotland, 2002.