Expert Systems With Applications 92 (2018) 192–205

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Automatic discovery of concepts and actions

Ana C. Tenorio-González[*], Eduardo F. Morales

Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro #1, Sta. Ma. Tonantzintla, Puebla 72840, México

ARTICLE INFO

Article history:
Received 29 December 2016
Revised 23 May 2017
Accepted 9 September 2017
Available online 20 September 2017

Keywords:
Relational representation
Intrinsic motivation
Reinforcement learning
Concept formation
Predicate invention
Discovery system

ABSTRACT

A truly autonomous artificial intelligence agent should be able to drive its own learning process. That is, decide what to explore and what to learn, identifying what constitutes potential useful data as examples of concepts or what strategy to follow to solve a new task. Different efforts have been developed in machine learning towards this aim. Approaches that introduce new concepts, like *predicate invention* in Inductive Logic Programming (ILP) techniques, normally require the selection of examples by the user. Techniques that learn behavior policies through exploration like Reinforcement Learning (RL) with *intrinsic motivation*, to guide the agent into interesting areas to discover new goals, assume that all the states and actions are predefined in advance. In this paper, we describe a system, called ADC, that combines techniques from ILP with predicate invention and RL with intrinsic motivation to discover new concepts, states and actions to learn behavior policies. ADC drives its own learning process, collecting its own examples for autonomously learning concepts. These new concepts can be used to describe its environment and define new states and actions used to learn behaviors to solve tasks. We show the effectiveness of our approach in simulated robotics environments.

1. Introduction

Designing autonomous artificial intelligent agents requires to create representations for the current knowledge of the agent, methods to acquire new knowledge and to reason with it, and mechanisms to provide incremental and continuous learning (Russell & Norvig, 2008). Ideally, carrying all these processes with low human intervention. Among the Machine Learning (ML) techniques that can be useful to address these challenges are Inductive Logic Programming (ILP) (Muggleton, 1991) and Reinforcement Learning (RL) (Sutton & Barto, 1998). ILP provides a first-order language and robust inference mechanisms from which knowledge, represented as abstract concepts, can be induced. Some ILP techniques also provide predicate invention to introduce new predicates to improve the background knowledge of the system. On the other hand, RL has been successful for learning (near) optimal behavior policies from interactions with the environment. Although, RL normally uses reward functions defined by users, recently, self-motivation functions (Merrick & Mahler, 2013; Singh, Barto, & Chentanez, 2005) have been introduced to automatically guide the agent to reach *interesting* situations that may be useful to solve tasks. These approaches, however, are far from creating autonomous agents as they depend on a careful selection of examples or predefined descriptions of states and actions.

In this paper, a system called ADC, for Automatic Discovery of Concepts, is described. An agent with ADC automatically drives its own learning process to discover relational concepts and learn behaviors. An agent with ADC incrementally builds a graph-based representation of its environment while exploring its environment using intrinsic motivation. Frequent subgraphs are grouped and transformed into logic clauses from which relational concepts are learned using an ILP algorithm. While exploring its environment, the agent is immersed in an RL framework where policies are learned and where new state descriptions can be incorporated during the learning process using the newly discovered relational concepts. Once a policy is learned for a particular subgoal the sequence of states and actions involved in the policy are stored for future use.

The main contributions of ADC are: (a) a novel concept discovery algorithm where an agent automatically decides what to learn by gathering instances of unknown but potential useful concepts and (b) a novel reinforcement learning algorithm for learning behaviors which uses intrinsic motivation and where the state representation can dynamically changed during the learning process. This paper is mainly focused in the second contribution taking advantage of the first one. In particular, this paper describes how the proposed approach can construct new state definitions, with newly

[*] Corresponding author.
E-mail addresses: catanace17@inaoep.mx (A.C. Tenorio-González), emorales@inaoep.mx (E.F. Morales).
URL: https://www.inaoe.mx (A.C. Tenorio-González)

https://doi.org/10.1016/j.eswa.2017.09.023
0957-4174/© 2017 Elsevier Ltd. All rights reserved.

discovered concepts, and how new actions can be learned while exploring an environment. Also two novel techniques (biased actions and intrinsic motivation function based on an asymmetrical version of the Wundt's curve (Berlyne, 1960)) are introduced to automatically guide the exploration of the environment.

The paper is organized as follows. Section 2 provides an overview of related work in two fields: ILP with demand-driven predicate invention and RL with intrinsic motivation. Section 3 describes the ADC algorithm. Section 4 describes the experiments performed to evaluate ADC and its components in two simulated domains using a humanoid robot. Finally, conclusions and future research directions are given in Section 5.

2. Related work

Regarding concept discovery, a wide range of successful approaches are based on Inductive Logic Programming (ILP) with predicate invention. In particular, ADC is closely related to ILP systems with demand-driven predicate invention. Demand-driven predicate invention is used when the existing vocabulary is not enough to form a concept, so it is necessary to add a new predicate (Stahl, 1996). Some of the most relevant works will be described as follows.

Multiple concept learning with predicate invention in robotics domains has been addressed in Hyper (Kosmerlj, Bratko, & Zabkar, 2011; Leban, Zabkar, & Bratko, 2008). Hyper starts with background knowledge and can perform predicate invention adding placeholder predicates (predicates which are being invented) to learn the target predicates. Positive and negative examples are provided to the system, the positive examples are obtained by the robot while it explores the environment guided by a human (through commands) and the negative examples are synthetically generated.

Meta-Interpretive Learning (MIL implemented by Metagol$_{D/O}$) (Muggleton, Lin, Pahlavi, & Tamaddoni-Nezhad, 2014; Muggleton, Lin, & Tamaddoni-Nezhad, 2015) is based on meta-rules, induction and abduction, to produce higher-order datalog programs which take advantage of predicate invention and recursivity. MIL is based on incremental declarative multi-predicate learning, using meta-rules to conduct the search of the hypothesis from a set of examples. Meta-rules are like templates where the meta-interpreter performs substitutions to introduce new predicates (predicate invention). Metagol$_O$ has shown the advantage of using composite objects and actions (formed by other primitive and/or composite objects and actions) to produce resource efficient strategies from examples. Experiments with composite elements have been performed with a simulated humanoid robot.

An alternative approach based on teleo-reactive programs (TRP) for learning new concepts and skills of different hierarchies from existing knowledge is proposed in Li, Stracuzzi, and Langley (2008). In the manner of a STRIPS planner, each skill consists of a goal, an initial state or precondition, an action or method to reach the goal, and a final state or post-condition. First a bottom-up inference mechanism is performed to identify the current state using the perceptions of the agent and background knowledge, then TRP searches for the first high-level goal that is not satisfied, and tries to form a path in the hierarchy of skills from the current state of the agent to that goal. When a sequence of skills to achieve the goal cannot be found, the algorithm introduces new skills. The new skills form their preconditions and goals using preconditions of concepts and skills closest to the goal. SPI and TRP were tested on databases and card games, respectively.

Also, a related work called *Statistical Predicate Invention* (SPI) (Stanley & Domingos, 2007) proposes a generalization of predicate invention, known as statistical learning for hidden variable discovery. The algorithm, *Multiple Relational Clustering* (MRC), is presented to cluster objects, attributes and their relations using Markov logic (an extension of FOL), where each cluster represents a unit predicate.

ADC introduces new predicates (concepts) automatically, as SPI, TRP and Hyper, but also automatically discovers examples of those potential concepts from the environment. In this sense, ADC does not need databases of examples or using predefined templates for the discovery of new predicates. ADC is able to collect positive examples and create negative examples for the induction process. As TRP and Hyper, ADC performs incremental hierarchical multi-predicate learning because of its graph-based representation. Although, other approaches have been proposed to also learn hierarchical concepts (e.g., Chien, Hu, and Ju (2009); Davis et al. (2011); Fu and Buchanan (1985); Holder, Cook, and Djoko (1994); Rivest and Sloan (1994); Rosca (1997); Tani and Nolfi (1999); Zupan, Bohanec, Bratko, and Demšar (1999)), they are usually designed to work on databases or in controlled environments, as in SPI, TRP and Hyper. Metagol has shown the advantages of learning composite objects and actions, while ADC has been used to discover composite concepts about objects in robotics domains. The use of meta-rules in MIL is a powerful tool for predicate invention, but it depends on the number and design of the meta-rules to produce useful predicates. Also, Metagol, like traditional ILP systems, uses examples provided by the user. In this sense, the main advantage of ADC over other ILP systems, including Metagol, is its ability to automatically discover and collect examples of potential concepts directly from the environment, driving its own learning process.

RL with relational representations of states, actions and specific goals has been addressed in what is called, relational reinforcement learning (RRL) (Driessens & Džeroski, 2004; Džeroski, Raedt, & Driessens, 2001; Martínez, Alenya, & Torras, 2015; Morales, 2004; Nickles & Rettinger, 2014), but contrary to ADC, most of these works have fixed representations that are not created during the learning process. Our approach has some differences compared with recent work as Deep reinforcement learning (DRL) (as learning to play video games (Mnih et al., 2013) or in robotic tasks (Levine, Finn, Darrell, & Abbeel, 2016)). The representation used in ADC is easier to understand than that used in DRL. Unlike DRL, ADC is able to discover its own examples to learn incrementally, even when the learning process is interrupted. Also, in ADC when a piece of knowledge is added to its background knowledge, it can be used in other learning tasks.

Recent research on RL has focused on the definition of a reward function that is independent of the main goal to guide the learning process (Merrick & Mahler, 2013; Singh et al., 2005). RL with intrinsic motivation (IM) (Singh et al., 2005) aims to design a general method of motivation to help an agent to autonomously develop skills, regardless of the traditional extrinsic rewards which may or may not simultaneously exist with the intrinsic motivation mechanism. Several mechanisms and applications of intrinsic motivation (IM) have been addressed from different disciplines, from studies in natural systems to artificial intelligence algorithms for the design of self-motivated agents, such as robots that can acquire knowledge and skills through their own experience (Baldassarre & Mirolli, 2013; Merrick & Mahler, 2013). Some approaches have worked with IM based on novelty measuring the level of agreement between the expected and the obtained observations using incremental hierarchical discriminant regression trees and habituated self-organizing maps (Saunders & Gero, 2001; Zhang & Weng, 2002). However, this type of models have problems with random events which are recognized as highly novel (Merrick & Mahler, 2013).

Some researchers have proposed motivators to avoid giving high rewards to random events (Kaplan & Oudeyer, 2003; Merrick & Mahler, 2013; Schmidhuber, 1990). In Kaplan and Oudeyer (2003) three motivators are used: predictability, familiarity, and stability. Predictability measures the prediction error of an observed state in a situation characterized by a set of observations and actions. Familiarity measures how often a state transition occurs. Stability measures the distance between an observation of a state and its average value over a period of time. These three components are used to calculate the final intrinsic reward, being this larger with maximum stability and with increasing familiarity and predictability. However, when the goal is to perform a task rather than staying in certain states, stability can be a problem (stability can lead to stay in similar/equal states). Merrick and Maher (2017); 2013); Schmidhuber (1990) have used notions of cognitive concepts as interest, boring and curiosity using neural networks (curiosity by predictability), context-free grammars and potential tasks (interest based on occurrences of events). These mechanisms have been tested on sensing and controlling motors of a robot and in computer games.

IM for measuring competence progress from sensorimotor vectors data using learning by imitation and learning inverse models has been used in Nguyen and Oudeyer (2014). In Georgeon, Marshall, and Gay (2012) the authors search for sensorimotor patterns, with self-motivation based on a value function, to analyze behaviors instead of states of the environment. In Bonarini, Lazaric, Restelli, and Vitalli (2006), the authors learn skills based on three stages: babbling (random exploration), motivating (based on the probabilities over states and actions preferring states difficult to reach and states which lead to infrequent states), and skill acquisition (by self-generated reinforcements and goals). Also, different kinds of IM based on novelty and habituation for cumulative learning have been applied to learn control programs applied to robotics (Baldassarre & Mirolli, 2013).

In reinforcement learning, IM has been used to provide agents with a sense of self-motivation to learn tasks. A representation of skills based on *options* with IM based on novelty (prediction of outgoing events) is used to learn skills at different levels (hierarchies) in intrinsically motivated RL (Singh et al., 2005). IM based on differences over the time between value functions has been used to guide the exploration mechanism (Simsek & Barto, 2006). A simplified measure for IM in RL is proposed in Bureau and Sebag (2014), where states are represented as multi-armed bandit problems, the IM function considers the number of visits to states and an entropy measure. Sufficiently visited states are not longer considered as intrinsic goals.

The IM used in ADC is similar to the measures proposed in Bureau and Sebag (2014); Merrick and Mahler (2013) but it is designed to work on a concept discovery framework during the exploration of unknown environments, does not require an *a priori* model of the task that is learning, and it does not depend on the occurrence of specific events in the environment.

Besides the aforementioned approaches of RL with intrinsic motivation, a vast number of approaches has been developed using traditional RL for learning behavior policies. However, there is little work on learning other elements, such as representation of states and actions which may assist an agent to identify and improve the exploration of the environment. ADC is able to create and change its own state representation during learning and can learn useful sequences of actions to solve particular tasks. ADC mechanisms are simple to implement, independent of the RL algorithm, and can work in an environment where states may not be characterized with its current background knowledge.

3. ADC: Automatic discovery of concepts

Our approach ADC for intelligent agents is designed to automatically discover concepts and learn behaviors, self-driving its learning process in unknown environments. The agent has an initial background knowledge stored in first-order logic formed by definitions of elements of the environment and relations between them, as well as primitive actions (Tenorio-González & Morales, 2016). The agent explores the environment performing its actions, gathers information with its sensors and identifies elements and their relations with its background knowledge. For instance, given information from laser readings, and definitions in the background knowledge of how to recognize from that sensor's information, say, a “wall”, with some relations between “walls”, like “corner”, a robot traversing its environment will use them to identify “walls” and “corners” in the environment. The objects and relations that can be recognized from the sensors are incrementally represented in a graph while the robot is exploring its environment. The agent uses this representation to learn concepts about objects and identify situations where behaviors can be learned.

In ADC, given:

- An initial background knowledge *BK* formed by a set of object definition clauses *O*, a set of relation definition clauses *R* (between objects and/or their attributes), and a set of action definition clauses *A*.

Perform a self-guided exploration process, with intrinsic motivation and biased actions; and two simultaneous processes:

- Concept formation by an inductive logic programming process to learn concepts *Cg*.
- Learning of actions *A* by a reinforcement learning process based on a relational representation of states *S* defined by concepts *Cg*.

New knowledge ($Cg, S, A \in BK$) is acquired:

- Extending the initial *BK* ($O, R, A \in BK$) with concept definition clauses ($c_g \in Cg$), state definition clauses ($s \in S$) and additional actions *a* representing behavior policies π ($a \in A$).

The pseudocode of ADC is shown in Algorithm 1 and it is described in more detail in the following sections.

Algorithm 1 ADC algorithm.
```
1: Let: $BK$ background knowledge ($O$ objects, $R$ relationsamong
   objects, $A$ set of actions), $r_e$ reward, $T$set of conditions to iden-
   tify target goals
2: Start
3: Set groups $C = \emptyset$, $S = \emptyset$, graph $G$ = null,$\pi = \emptyset$, $Q_{s,a} = \emptyset$ and $e_{s,a} =$
   $\emptyset$
4: repeat
5:    repeat
6:       $Update\ Action\ Learning(BK, G, S, Q, e, r_e)$ % see Algorithm 3
7:    until current state satisfies $T$ or $size(G) < graph\_size$
8:    $Update\ Concept\ Formation(C, G, \pi, BK)$ % see Algorithm 2
9: until a task is learned or goal is reached
10: $AddNewActions(\pi, BK)$ % see Algorithm 4
11: End
```

3.1. Concept formation

The initial background knowledge *BK* is formed by a set of basic objects *O* (e.g., walls) and relations *R* between objects (e.g., touches, on, near, etc.) as well as some basic actions *A* (e.g., move-forward, turn-right, etc.). During the exploration phase, a graph *G*, representing objects and relations identified with the current *BK* from sensor readings in the environment, is incrementally built, where objects and their attributes are represented as vertexes and relations as edges. When the arity of a relation is greater than two and there is not direct mapping to a simple graph, it is possible to use conceptual graphs (Sowa, 2008) to represent objects and their relations as concepts and conceptual relations (both, as vertexes of a graph). The agent incrementally constructs a graph until it reaches an intrinsic goal (an interesting state) or an extrinsic goal (representing a predefined goal of the current task), or a given maximum graph size.

ADC performs four main steps for learning concepts (see Algorithm 2):

Algorithm 2 ADC: Update Concept Formation

```
1: Let: $BK =$ background knowledge, $C =$ set of groups, $G =$constructed graph.
2: **Start**
3: Find frequent sub-graphs ($SG$) in $G$
4: **for** each sub-graph $sg \in SG$ **do**
5:     **if** $sg$ is similar to elements of an existing group $g \in C$ **then**
6:         $g = g \cup \{sg\}$
7:     **else**
8:         Let $g' = \{sg\}$ and $C = C \cup \{g'\}$
9:     **end if**
10: **end for**
11: **for** all $g \in C$ that was created or updated **do**
12:     Induce a new concept $c_g$
13:     $BK = BK \cup \{c_g\}$
14: **end for**
15: **End**
```

1. In the graph-based representation G constructed during the exploration, ADC identifies potential concepts by searching for frequent sub-graphs sg. ADC assumes that frequent sub-graphs represent potentially relevant concepts. The frequent sub-graphs are discovered using a sub-graph discovery system, Subdue (Holder et al., 1994), that uses inexact matching and the minimum description length (MDL) principle.
2. The frequent substructure discovery process can identify a very large number of sub-graphs many of which are redundant or could be instances of a more general concept. To overcome this, ADC forms groups g of similar sub-graphs with their equivalent clausal form definition (each subgraph is represented as a logic clause too). Two similarity measures are used to group sub-graphs. The first measure clusters instances which share the same objects and relations although their structures and sizes may be different. The second measure considers the structure and size, evaluating the cost of structural changes to make two sub-graphs equal.
3. Each time a sub-graph sg is added to a group g, its clausal form and all the clauses associated to that group are used to reinduce the concept definition c_g of that group using Progol (Muggleton, 1995). The clauses representing the sub-graphs are used as positive examples and negative examples are taken from other groups and/or created using variations of the positive examples (variations of positive examples refers to examples created by removing elements from the definitions of the positive examples). The induced concept is a clause definition where the head is an *n-ary* predicate and the body is a conjunction of predicates, with n being the number of the distinct arguments used in the predicates of the body of each clause.
4. The new induced definition c_g of a concept is used to compress the current graph G by substituting the instances of the whole sub-graph (representing c_g) by simple nodes with their corresponding arcs. Currently, the compression of G using c_g is performed through the algorithm provided by Subdue (Holder et al., 1994). The compressed graph is then used as new input to our algorithm to find new common sub-graphs, from which new, hierarchical concepts can be induced.

A detailed description and evaluation of the concept formation process of ADC can be found in Tenorio-González and Morales (2016). In this paper, we incorporated this concept discovery algorithm in a reinforcement learning framework to create an autonomous agent that could also discover useful behaviors to solve tasks. In this case, the graph, from which the concepts are obtained, is incrementally constructed using an ϵ*-greedy* exploration strategy with intrinsic motivation and biased actions (as explained below). Also the learned concepts incorporated into the background knowledge are used to describe new states for the reinforcement learning algorithm described in the following section. New sequences of actions can also be learned and incorporated as primitive actions as described below.

3.2. Behavior policies learning

In addition to discovering new concepts, ADC can learn how to perform interesting tasks. This is achieved through a reinforcement learning algorithm based on SARSA(λ) (Sutton & Barto, 1998) with several modifications (see Algorithm 4), explained in this section:

Algorithm 3 ADC: Add new actions

```
1: Let $s_i =$ initial state in current task, $A$ set of actions in $BK$, $\pi =$ learned policy
2: **Start**
3: Let $sec =$ sequence of states and actions ($s \in S$, $a \in A$) from $s_i$ to final state $s_f$ according $\pi$
4: Let new action be a clause $na \leftarrow s_i, sec, s_f$
5: Set $A = A \cup \{na\}$
6: **End**
```

(i) The set of states and its representation can be constructed and changed while the agent is learning a new task, (ii) once a policy is learned for a particular task, the sequence of state and actions contained in the policy, from the initial to the final state, are grouped as a new individual action and added into the background knowledge for use in future tasks, and (iii) the exploration strategy is based on a novel intrinsic motivation function and/or biased actions. In this stage the background knowledge BK will be extended adding state definitions S describing situations of the environment, and a behavior policy π as a new action. The pseudocode for behavior learning in ADC is presented in Algorithm 4.

1. State formation. When the agent explores the environment, each time the agent performs an action $a \in BK$, it checks which learned concepts ($c_g \in BK$) can be identified (are *true*) in that particular situation. The conjunction of those concepts is stored as a clause defining the new state s which is added to the existing set of states S, if and only if this state does not already exist in S. The body of the clause is formed by the conjunction of $\{c_g | c_g \in BK,\ c_g = true\}$ and the head by a predicate with an identifier argument. These state definitions can be updated adding new learned concept definitions during the learning process (see lines 5 and 8 of Algorithm 4). In contrast to traditional RL algorithms, ADC can create, increase and/or update its state representation during exploration. In practice, the agent may require more episodes to learn, but also more accurate descriptions of the state space can be achieved. Algorithm 5 (lines 4 and 7 of Algorithm 4) describes the process when the agent senses the environment, updates its representation G and tries to recognize its current situation using its BK. Algorithm 6 (lines 5 and 8 of Algorithm 4) describes how a state definition can be created or updated with definitions of new discovered concepts $c_g \in BK$. Also, the number of visits to the state s is updated, this value is used to calculate the intrinsic motivation as it is described in the next steps.

Algorithm 4 ADC: Update action learning

```
1: Let: $BK =$ background knowledge, $C =$ set of groups, $S =$ setof states, $G =$constructed graph, $Q$ and $e$ = Q and trace values, $r_e$=reward
2: Start
3: Set $r_e = 0$, $r_i = 0$, $maxNumVisits_S = 1$, $newKnowledge = false$
4: if $Sense(s_t, BK, G, newKnowl)$ then
5:     $Update\ state(s, S, BK)$
6:     Let $a \leftarrow Perform\ action(s)$ observe $r_e, s_{t+1}$
7:     if $Sense(s_{t+1}, S, BK, G, newKnowl)$ then
8:         $Update\ state(s_{t+1}, S, BK)$
9:         Let $r_i \leftarrow Calculate\ r\_i$ using Eq. 6
10:        Let $r = r_e + r_i$
11:        if $s_{t+1} \neq s_t$ then
12:            $value_a$ += $inc$
13:        else
14:            if $newKnowl = false$ then $value_a$ −= $dec$ else $value_a$ += $inc$ end if
15:        end if
16:        Let $a' \leftarrow Perform\ action(s')$
17:        Update $Q(s, a)$ and trace values, $e(s, a)$, for each$s, a$ pair to update $\pi$ {SARSA−$\lambda$ Equations 1 - 5}
18:    else
19:        if $newKnowl = false$ then $value_a$ −= $dec$ else $value_a$ += $inc$ end if
20:        Let $a' \leftarrow Perform\ action(s')$
21:    end if
22: else
23:    if $newKnowl = false$ then $value_a$ −= $dec$ else $value_a$ += $inc$ end if
24:    Let $a \leftarrow Perform\ action(s)$
25: end if
26: End
```

Algorithm 5 ADC: Sense

```
1: Let $BK =$ background knowledge, $G =$ graph, $s =$ state $(s \in S)$, $c_g =$ concept definition $(c_g \in BK)$, $newKnowl =$ flag
2: $Sense(s, BK, G, newKnowl)$
3: Start
4: Add objects $O \in BK$ and relations $R \in BK$ identified inthe environment to the graph $G$
5: if $BK$ has been updated then $newKnowl = true$ else $newKnowl = false$ end if
6: if situation is defined by $\{s|s \in BK,\ s = true\}$ or itcan be identified using $\{c_g|c_g \in BK,\ c_g = true\}$ then Return$true$ else Return $false$ end if
7: End
```

2. Behavior learning. In ADC, the learning of behaviors is performed with a SARSA algorithm with eligibility traces (Sutton & Barto, 1998), and with intrinsic motivation and biased actions. When an extrinsic (traditional) goal is given by a user, a behavior policy π to reach that goal is learned using SARSA(λ). In SARSA(λ), $Q(s,\ a)$ refers to a value function that represents the expected total reward that an agent can received being in state s and performing action a, following a particular policy. $e(s,\ a)$ is a function used in reinforcement learning to represent the eligibility trace. π is a policy that given a state returns an action $(\pi(s) \rightarrow a)$. π^* is the optimal policy. In ADC, these values are updated, as in SARSA(λ), following Eqs. (1)–(5) (lines 10 and 17 in Algorithm 4).

$$r = r_i + r_e \tag{1}$$

Algorithm 6 ADC: Update state

```
1: Let $A =$ set of actions, $maxNumVisits_S =$ maximum number ofvisits, $s =$ state $(s \in S)$, $minNumVisits_{a-s} =$ minimum numberof times to perform an $a$ on each $s$, $c_g =$ concept definition $(c_g \in BK)$, $BK =$ background knowledge
2: Start
3: Verify which $c_g \in BK$ are $true$ in the situation
4: Let $s \leftarrow c_{g_1}, c_{g_2}, \ldots$., defining $s$ withthose $\{c_g|c_g \in BK, c_g = true\}$
5: Set $S = S \cup \{s\}$ or update $s$ in $S$
6: if $numVisits_s < (|A| * minNumVisits_{a-s})$ then
7:     $numVisits_s = numVisits_s + 1$
8:     if $numVisits_s > maxNumVisits_S$ then
9:         Let $maxNumVisits_S = numVisits_s$
10:    end if
11: end if
12: Return $numVisits_s$
13: End
```

$$\delta = r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{2}$$

$$e(s_t, a_t) = e(s_t, a_t) + 1 \tag{3}$$

for all s, a

$$Q(s, a) = Q(s, a) + \alpha \delta e(s, a) \tag{4}$$

$$e(s, a) = \gamma \lambda e(s, a) \tag{5}$$

Where r_i,r_e are the rewards, α is the learning rate, γ is the discount parameter, and λ is a discounted factor over the length of a trace.

A policy π to reach a goal is considered as learned if the changes in the accumulated reward after consecutive trials are smaller than a threshold value. Once a policy π is considered as learned, a new (macro) action na defined as a clause, representing π, is added to the background knowledge $A \in BK$. The body of the clause is formed by the conjunction of predicates of the starting state s_i of the task, the sequence of states and actions $(s_i - a_i,\ s_s - a_s,\ s_t - a_t, \ldots,\ s_f)$ suggested by the policy from this state to the final state, and the final state s_f. The head of the clause is formed by a predicate with an identifier argument. Algorithm 3 describes that the BK $(S,\ A \in BK)$ of ADC is updated with a new action na, using as input data the behavior policy π learned by the modified SARSA(λ) algorithm (Algorithm 4), and the current background knowledge BK of ADC.

3. Intrinsic motivation. The intrinsic reward is represented by r_i and it is added as an additional reward to the traditional RL reward function (r_e) as $r = r_i + r_e$ (see lines 9 an 10 in Algorithm 4). The intrinsic reward is used to promote the exploration of interesting states. In our approach, it is used to guide the agent to situations from which new concepts may be learned or which are required to learn new tasks. ADC increases gradually the interest for new states that become frequent and decreases rapidly the interest for *known* states. The agent gets bored fast in *known* states to accelerate the learning process and the intrinsic motivation function also prevents the agent to be dazzled with random events. This function is computed by a modified Wundt's curve (Berlyne, 1960) which is based on psychological concepts. In our case we use an asymmetric Gaussian function to obtain the desired behavior illustrated in Fig. 1.

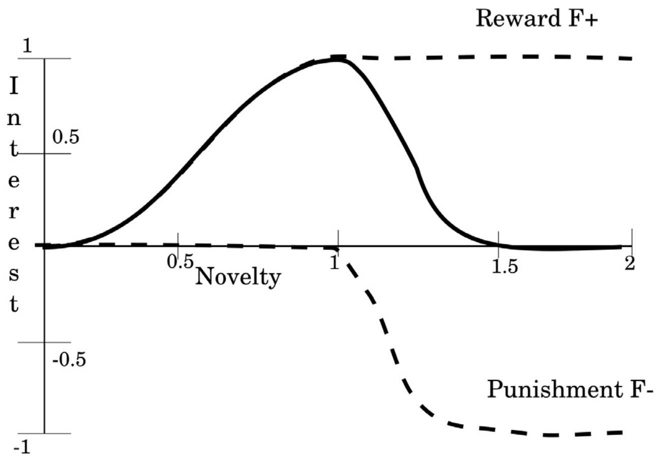

Fig. 1. Sigmoid curves modeling rewards (positive curve) and punishments (negative curve) are added to create an asymmetrical Wundt's curve modeling interest. The asymmetric Gaussian function models the interest describing how an agent is attracted to novel situations cautiously, losing interest in them slightly faster when the situation has been enough seen.

The curve model of r_i in ADC is calculated using Eq. (6) (see line 9 in Algorithm 4).

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

The first term in Eq. (6) corresponds to the $Reward\_Curve_{F+}$ and the second term represents the $Punishment\_Curve_{F-}$ (see Fig. 1). The model requires the definition of the following parameters: *maxReward* and *maxPunish* (upper limits of reward and penalty functions), $minNovelty_R$ and $minNovelty_P$ (lower limits of novelty to reward and punish a state), and C_1 and C_2, where $C_1 < C_2$ defines the asymmetry of the curve. Given a situated agent in the environment, the novelty value is updated as follows:

$$\begin{aligned}
&\text{If } \{s|s \in BK,\ s = true\},\ \text{the situation is known, then,}\\
&\text{If } numActions_s = |A| \text{ or } numVisits_s \geq (|A| * minNumVisits_{a-s})\\
&novelty = 0\\
&\text{Otherwise,}\\
&novelty = \frac{numVisits_s * noveltyMostVisited_s}{maxNumVisits_S}\\
&\text{If the situation is unknown,}\\
&novelty = maxValueNovelty
\end{aligned} \quad (7)$$

Here, $numActions_s$ keeps track of the number of actions performed in a state s. $minNumVisits_{a-s}$ is the minimum number of actions that should be performed in a state s. $numVisits_s$ is the number of visits of the current state during exploration, each state has its own value set to 0 at the beginning of the experiments. $maxNumVisits_S$ is the number of visits of the most viewed state (among all the states) until then, initially, this value is 0 for all the states, and it is equal to the $numVisits_s$ value of the most visited state. $noveltyMostVisited_s$ is the novelty value of the most visited state in the past. In the proposed algorithm, a state stops receiving intrinsic rewards when all its 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.

4. Biased actions. Another option to change the exploration process of the agent is to bias the preference for some actions. In this research this is implemented in the exploratory (random) actions of an ϵ-*greedy* strategy. The RL algorithm follows a modified ϵ-*greedy* strategy. Taking completely random actions with probability ϵ, within a concept discovery algorithm like ADC, can frequently guide the system to states which can not be characterized with the current background knowledge or where it is not possible to learn concepts or tasks. In these cases the exploration process can be largely extended. In ADC, instead of selecting a random action, the action with the highest $value_a$ is chosen. In the proposed approach, each action a in the *BK* has associated a value ($value_a$) which indicates its performance (and preference). When an action a is taken; it is rewarded or punished with a small value ranked from 0 to 1 (*inc* and *dec*, see lines 14, 19 and 23 in Algorithm 4). If the action a guided the agent to a state where new knowledge (concepts) was discovered or to a known state, then, its $value_a$ is increased with a small positive value *inc* (without exceeding 1). When the resulting state after performing an action a cannot be described or identified by the learned concepts or it is the same as the previous state, its $value_a$ is decreased with a small positive value *dec* (without decreasing below 0). The objective of this method is to accelerate the learning process specially in situations where the knowledge of the agent cannot be used to characterize the environment, where only random actions would have been taken. It is assumed that actions which were useful in the past (with greatest $value_a$) may be useful in unexplored states to reach situations where tasks can be learned, or in states that are not possible to characterize with the current background knowledge. The biased actions strategy can be applied during the selection of an exploratory action using $value_a$ as it is shown in Algorithm 7 (lines 16, 20 and 24 of Algorithm 4).

Algorithm 7 ADC: Perform action

```
1: Let $s =$ state ($s \in S$), $a =$ action ($a \in A$), $Q$ valuefunction structure (with indexes $s, a$), $value_a =$ reward value of anaction $a$
2: **Start**
3: **if** $\epsilon - greedy\_selection = \epsilon$ or situation cannot beidentified by $\{s|s \in S\}$ **then**
4:     **if** situation cannot be identified by $\{s|s \in S\}$ and an action$a$ with a highest $value_a$ exists **then**
5:         Choose action $a$
6:     **else**
7:         Choose action $a \in A$ randomly
8:     **end if**
9: **else**
10:    Select action $a$ with the best $Q - Value$ for state $s$
11: **end if**
12: Perform action $a$
13: Return $a$
14: **End**
```

The ADC algorithm ends when a set of conditions, defining the goal state, are satisfied (are true in the current state, $T \leftarrow condition1 \wedge condition2 \wedge condition3 \ldots.$). These conditions depends on the task that is being learned. For example, a goal could be considered as reached (satisfied) if the agent can perceive a green wall.

The most expensive process in ADC is the discovery of concepts, because a frequent subgraph algorithm is used. Although finding sub-graphs is equivalent to sub-graph isomorphism, which is NP-complete (Cook, 1971) (planar subgraph isomorphism can be solved in linear time Eppstein (1999)), we use Subdue, which is designed to find a large number of substructures under computational constraints (in polynomial time). The inexact matching process for a graph 1 with n vertices and a graph 2 with

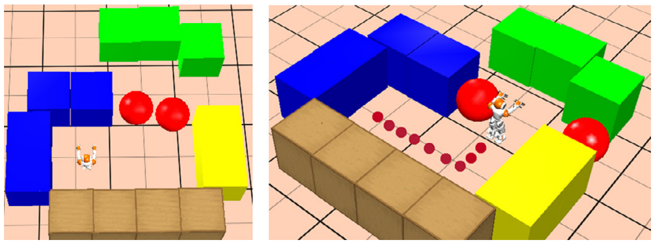
Fig. 2. Environment of *Mobility of objects*: initial state and example of goal state. A simulated humanoid in a room with walls of boxes and movable balls. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

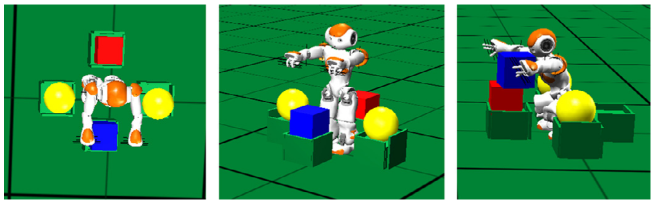
Fig. 3. Environment of the experiments. Examples of initial and goal states. A simulated humanoid surrounded by two yellow balls, one red box and one blue box. Each object is on a fixed green base, all objects could be grabbed by the robot. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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). A more complete analysis of Subdue's run time can be found in Cook, Holder, and Djoko (1996). The website for the ADC system can be found in: https://sites.google.com/site/automaticdiscoveryconcepts/.

4. Experiments

We tested ADC in two robotics domains where the concepts of *stability* and *mobility* of objects were discovered (as in related work Leban et al. (2008)). The concept of *stability* refers to discovering when placing one object over another object is stable or not. The concept of *mobility* refers to discovering when an object is movable or not when pushing it. We performed several experiments to validate the proposed methodology:

1. Compare the proposed asymmetric Wundt's curve against the original Wundt's curve and a measure based on entropy for intrinsic motivation.
2. Compare the performance of the different components proposed in ADC (IM and biased actions) in a RL algorithm with predefined states.
3. Perform the previous experiment but now with ADC at full capacity where new concepts and states can be created during the learning process.

In the environment for *mobility* of objects, the agent was placed inside a room ($3m^2$) formed by four walls, see Fig. 2. The agent explored the environment performing four basic actions (move forward, go left, go right, and push) and states were identified by only four colors (green, blue, yellow and red) and by the relative positions between the objects and the agent (in front of). The agent should learn to *go to the red balls and discover the hidden green wall by pushing the balls*. Four states were defined and provided for the experiments with predefined states (*agent in front of a blue wall, agent in front of a yellow wall, agent in front of a green wall, agent in front of red balls*). The initial background knowledge for ADC is shown in Table 1. In this experiment for learning actions, one extrinsic reward was given and graphs of small size were chosen because of the size of the environment, see Table 3.

In the environment for *stability* of objects, the agent was surrounded by four objects, see Fig. 3. The agent explored the environment performing three basic actions (grab, put and turn right) and states were described by only 3 colors (yellow balls, a blue square box and a red square box) and by the relative positions between objects and the agent (in front of, grabbing and on). When an object is dropped from its base, it is returned to its original stable base position, in this way, the agent could experiment with the four objects continuously. In this domain the agent should learn to *put an object on the top of other object, discovering that this task can only be accomplish when the square object is in the base*. Four states were given by a user for predefined states (*agent in front of an object on its base, agent grabbing an object in front of a square object, agent grabbing an object in front of a round object (or an empty space), and agent with its empty hands in front of an object on top of another one*). The initial background knowledge for ADC is shown in Table 2. Two rewards were used to accelerate the learning of behaviors, a large extrinsic reward for the goal and a small extrinsic reward for an intermediate achievement (in this case when the agent grabs an object), and a small graph size was chosen because of the small environment, which could be completely explored with a small number of actions, see Table 3.

In the first experiment we simulated our own agent and environment, in the rest of experiments our agent was a Nao robot (Aldebaran-Robotics, 2012) in simulated environments through Webots for Nao (Michel, 2004). The robot was equipped with a camera in its head (to perceive colors) and its four main sonar sensors in its chest (to perceive distances).

In the experiments with a relational representation, the situations of the robot and the objects in the environment (positions of objects and robot, presence and color of objects) were determined by the sensor readings of the robot satisfying certain predicates of its *BK*.

For all the experiments, the general parameters of ADC are in Table 3. The values of the parameters for SARSA(λ) were taken from Singh and Sutton (1996). The values used for intrinsic motivation were selected according to Wundt's curve principles. The parameters used for subgraph discovery were experimentally selected guided by Holder et al. (1994) and Cook and Holder (1994). All the experiments were repeated 10 times, and conducted on Debian Linux 7.1 (OS) on an Intel Core i7-3610QM 2.3 GHz (CPU) with 6 Gb of RAM at 1600 MHz.

4.1. Comparison of IM functions in a concept discovery framework

We compared our proposed IM function against other similar IM functions when learning concepts related with the *stability* of objects. We tested two variations of the Wundt's curve and a similar IM strategy but based on entropy: (i) the original curve (Berlyne, 1960; Merrick & Mahler, 2013), (ii) the measure of entropy described in (Simsek & Barto, 2006) and (iii) our asymmetric curve. Table 4 shows the learning time, number of episodes, number of new created states, and number of discovered concepts for each IM function. This experiment was performed with a simulated agent, so times reported are shorter than those presented in the experiments of the next section, where real learning times using a robot in the same task are shown.

From this experiment, it can be appreciated that the proposed asymmetric Wundt's curve achieved the best times, with a smaller number of episodes and a larger set of concepts, all with the lower standard deviations. In the rest of the experiments we used our proposed IM function.

Table 1
Background knowledge provided for *Mobility of objects*.

Background knowledge: Objects, relations and actions
Objects:
object/4: Instance of an object of the world with its arguments: identifier, previous position, current position and color.
robot/3: Agent exploring the environment with its arguments: identifier, previous position and current position.
Relations:
in_front_of_me/2: Relative position (in front of) between an object of the world and the agent.
equal/2: Approximate equality of two positions.
different/2: Approximate inequality of two positions.
Arguments: identifiers of two instances (object or robot).
Actions:
go_right/0, go_right/2: Turn right 90° and move forward while obstacles are not perceived.
go_left/0, go_left/2: Turn left 90° and move forward while obstacles are not perceived.
go_forward/0, go_forward/2: Go forward while obstacles are not perceived.
push/0, push/2: Move the arms to push an object.
Arguments: identifiers of two instances (robot) in two different time stamps.

Table 2
Background knowledge provided for *Stability of objects*.

Background knowledge: Objects, relations and actions.
Objects:
object/3: Instance of an object of the world with its arguments: identifier, shape and color.
robot/1: Agent exploring the environment with its argument: identifier of the robot.
Relations:
in_front_of/2: Relative position (in front of) among an object on its base and the agent
grabbing/2: The agent is grabbing an object
on/2: An object is on top of other object
Arguments: identifiers of two instances (objects or robot).
Actions:
grab/0, grab/2: The agent moves its arms to grab an object
turn_right/0, turn_right/2: The agent turns right 90°
put/0, put/2: The agent moves its arms to place an object
Arguments: identifiers of two instances (robot) in two different time stamps.

Table 3
Values of parameters used for experiments.

Parameter	Value	Parameter	Value
In all experiments			
γ	0.9	λ	0.9
α	0.1	ϵ	0.2
r_e	10		
In experiments with r_i			
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
In experiments with *biased actions*			
inc	0.1	*dec*	0.2
In experiments with ADC in mobility of objects			
graph_size	50		
In experiments in stability of objects			
graph_size (ADC)	20	r_e	0.1

Table 4
Average convergence time, number of episodes and number of concepts for each experiment of *Stability* with different intrinsic motivation functions in ADC.

Experiment	Time (h:min:s) AVG (SD)	Episodes AVG (SD)	States AVG (SD)	Concepts AVG (SD)
ADC with entropy	00:24:09 (00:12:15)	32 (2)	10 (1)	51 (7)
ADC with Wundt	00:18:03 (00:07:51)	33 (3)	11 (5)	45 (10)
ADC with asymmetric Wundt	00:17:30 (00:04:46)	28 (1)	11 (1)	53 (6)

4.2. Evaluation of ADC and its components

We divided these experiments in two parts: (i) Compare the functionality of the different components proposed in ADC against RL but with predefined states (i.e., without concept learning and with a state representation provided by a user). and (ii) Compare ADC and the functionality of the different components proposed in ADC against RL with concept formation (i.e., with concept and state formation).

We labeled those variations of the experiments as:

1. Extrinsic: Traditional RL (Sarsa-λ with eligibility traces and traditional extrinsic reward function).

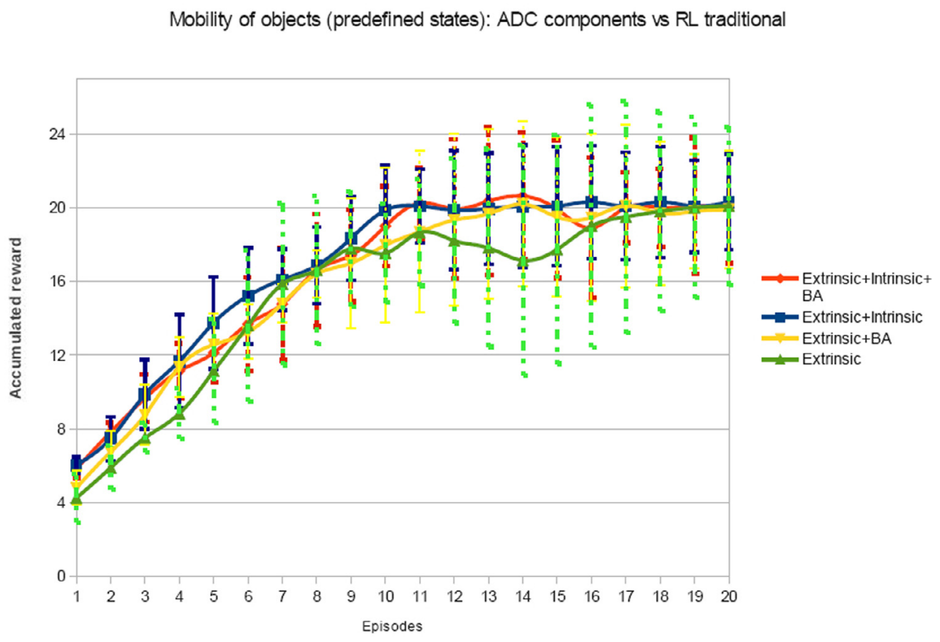

Fig. 4. Learning curves with the different components of the ADC algorithm and traditional RL in *Mobility of objects* with predefined states.

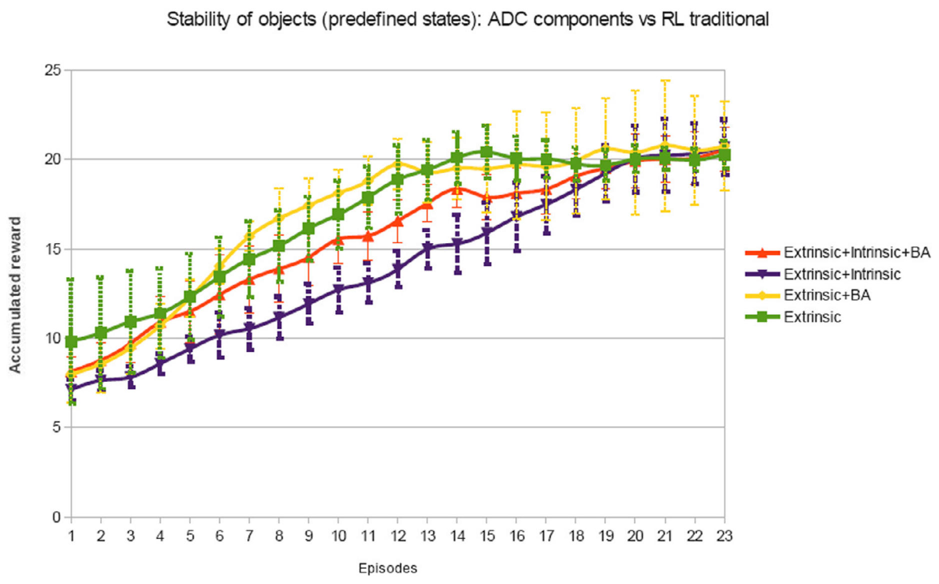

Fig. 5. Learning curves with the different components of the ADC algorithm and traditional RL in *Stability of objects* with predefined states.

2. Extrinsic + Intrinsic: Traditional RL as above but now with the proposed IM function based on the asymmetric Wundt's curve.
3. Extrinsic + Biased actions: Traditional RL but now with the proposed mechanism of biased actions.
4. Extrinsic + Intrinsic + Biased actions: Traditional RL with the proposed IM function and with the proposed mechanism of biased actions.

Learning curves, learning times and number of episodes obtained in both domains with predefined states are shown in Figs. 4 and 5 and Tables 5 and 6.

When the states are predefined in advance, and do not change during the learning process, the best results in both domains are obtained when only the proposed intrinsic reward function 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 up to 2 hours for *mobility* (see Table 5) and half an hour for

Table 5
Average convergence time and number of episodes for each experiment of *Mobility of objects* with the components of ADC and traditional RL with predefined states.

Experiment with predefined states	Time (h:min:s) AVG (SD)	No. episode AVG (SD)
Extrinsic (traditional RL)	03:34:09 (00:47:11)	17 (6)
Extrinsic + Intrinsic	01:21:14 (00:18:01)	10 (2)
Extrinsic + Biased actions	01:29:19 (00:35:03)	14 (4)
Extrinsic + Intrinsic + Biased actions	01:37:48 (00:25:00)	11 (3)

Table 6
Average convergence time and number of episodes for each experiment of *Stability of objects* with the different components of ADC and traditional RL with predefined states.

Experiment with predefined states	Time (h:min:s) AVG (SD)	Episodes AVG (SD)
Extrinsic (traditional RL)	03:00:01 (01:06:28)	20 (3)
Extrinsic + Intrinsic	02:27:09 (00:33:39)	20 (1)
Extrinsic + Biased actions	02:53:57 (00:44:53)	22 (2)
Extrinsic + Intrinsic + Biased actions	03:54:34 (00:10:54)	20 (1)

stability (see Table 6). Adding only the biased actions to a traditional RL algorithm also helps to reduce the time of convergence, reducing the number of actions to explore on each state as the learning process advances, however it learns in more episodes. Combining both strategies, intrinsic motivation and biased actions, reduces the learning time compared with a traditional RL algorithm, however, the learning time is greater than using the guiding strategies separately. It seems that when the intrinsic motivation tries to guide the robot away from often visited states, the biased actions produce additional pressure for using a subset of actions which hinders the convergence of the learning algorithm. In *stability* of objects, the combination of intrinsic motivation and biased actions is more consistent than the other strategies (a lower standard deviation in the number of episodes and time of learning). But, in this case the combination of intrinsic motivation and biased actions has an average learning time almost equal to the worst time when the exploration is guided by the traditional RL algorithm (see Table 6). This could be due to the characteristics of the environment with less unknown situations (in contrast to the *mobility* of objects domain, where it was more likely for the agent to find situations that could not be characterized by its current states). In this first set of experiments in both domains, it is shown that the proposed intrinsic motivation function is working as a good strategy in guiding the learning process.

The second part of the experiments was done using ADC. It incorporates new concepts, new states and new actions during the learning process. The learning curves, learning times, number of episodes, number of states and number of concepts obtained using ADC (incremental representation based on concepts and learning of behaviors) are shown in Table 7 and Fig. 6 for *mobility* of objects, and Table 8 and Fig. 7 for *stability* of objects. The total number of concepts learned with each configuration is shown, as well as the number of general, singleton and hierarchical concepts. General concepts refer to discovered concepts using several instances in each group, singleton concepts refer to discovered concepts involving only one instance, and hierarchical concepts refer to discovered concepts that have in their definitions previously discovered concepts.

Similar behavior policies, states and concepts could be learned by the two exploration strategies (IM and biased actions). The best results were obtained when ADC incorporated the intrinsic motivation function to guide the exploration process (according to the learning time, and consistency in number of concepts, states and episodes). The intrinsic motivation function keeps the agent focused in learning both, concepts and the behavior policy. The intrinsic motivation allows a more stable learning process (a lower standard deviation in number of concepts and learning time) than the traditional strategy of exploration. In this experiment with automatic state formation, the biased actions do not favor the learning process. This fact could be explained since more exploratory actions (biased actions reduce the number of exploratory actions) are needed to discover concepts and create states than when the states are given at the beginning of the learning process (with predefined states). Also, this effect of the biased actions remains even when combined with intrinsic motivation influencing the learning process. This explanation is supported by the good results obtained with the traditional RL algorithm (where only ϵ-*greedy* is used to select the exploratory actions). The final accumulated rewards from all the strategies are similar. The learning times in these experiments were, as expected, larger than those obtained with predefined states. However, here, new concepts were discovered and new states were automatically characterized using those concepts, with an accumulated reward and number of episodes similar to those obtained with predefined states.

From the experiments and results with predefined and relational representations for the *mobility of objects* and initial results with the *stability of objects* it was clear that the best results were going to be obtained with the intrinsic motivation option. In the experiments with relational representations in the first domain, we illustrated how ADC was able to construct and update its own state representation with a background knowledge enough to describe the environment. Also, we showed how ADC can self-guide its exploration, having best results using intrinsic motivation. The strategy with intrinsic motivation provided consistency in the learning process despite the inclusion of new concepts, states and actions. In this point of the experiments, we were more interested in the new concepts and actions discovered by the proposed system and in its ability to solve the task. So, we included only results with intrinsic motivation in the stability domain; as we were expecting similar results in terms of number of concepts, states and actions with the rest of strategies. In the second set of experiments in both domains, ADC showed its abilities to guide its exploration to discover new concepts (predicate invention) and learn a task with them. The concepts learned by ADC were concepts necessary for learning the task, and concepts describing characteristic features and relations of the elements in the environment.

As it was shown in Figs. 4–6 learning curves are very similar. Similar learning curves are expected since all algorithms are learning the same task with similar rewards. However, differences in total accumulated reward can be obtained depending on the exploration, construction of states and influence of the intrinsic motivation mechanism. What should be highlighted in the results is: the mechanisms to self-guide the exploration (intrinsic motivation and biased actions) are consistently guiding the learning process (with smaller standard deviations in most of the measured parameters than those obtained with other strategies) and reducing the learning times, also ADC is learning the same task than a traditional RL algorithm, but constructing and adapting its own state representation on the environment self-guiding its exploration. 95% Confidence intervals for all experiments can be found in Tenorio (2017).

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) automatically learned by ADC using only intrinsic motivation (ADC: Extrinsic + Intrinsic) in *mobility* of objects are shown in Table 9. The robot learned to go to the balls and move them to find the green wall. For reaching this task, four states were identified by the robot using ADC: (i) *the robot has reached a blue object that is in front of it* (*state(1)* using the discovered concept *blue_wall*, (ii) *the robot is in front of a yellow object* (*state(3)* using the concept *yellow_wall*), (iii) *the robot has reached a red object, it is in front of it* (*state(8)* using the concept *red_object*), and (iv) *the robot is in front of a green object* (*state(9)* using the concept *green_wall*).

ADC was able to learn additional useful concepts related with *mobility* in addition to the learned concepts that are used to solve the task, see Table 10. ADC could describe which objects were fixed and which were movable. The blue objects could not be pushed

Table 7
Average convergence time and number of episodes for each experiment of *Mobility of objects* with variations of ADC and traditional RL (with concept learning, state construction and behavior learning) (*G=General, S=Singleton, H=Hierarchical* concepts).

Experiment	Time (h:min:s) AVG (SD)	No. episode AVG (SD)	No. states AVG (SD)	No. concepts {G, S, H} AVG (SD) {AVG (SD), AVG (SD), AVG (SD)}
Extrinsic (traditional RL)	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)}
ADC: Extrinsic + Biased actions	18:29:43 (12:10:00)	18 (8)	20 (4)	140 (52) {34 (15), 107 (22), 113 (39)}
ADC: Extrinsic + Intrinsic + Biased actions	18:33:44 (04:03:00)	18 (4)	24 (3)	129 (16) {27 (6), 110 (25), 110 (24)}

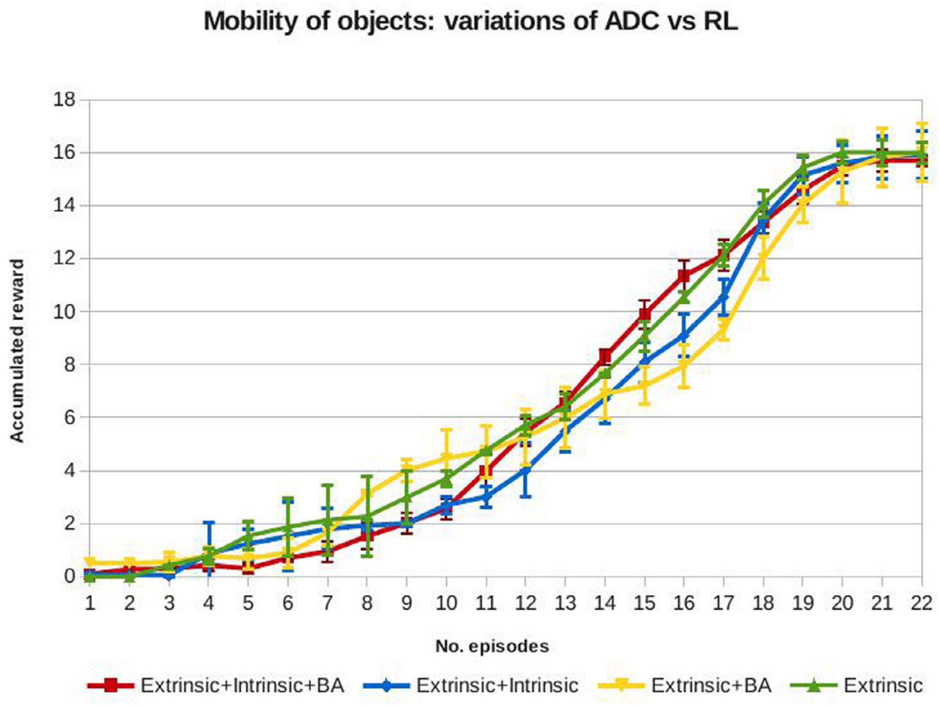

Fig. 6. Learning curves with variations of the ADC algorithm compared with traditional RL in *Mobility of objects* (with concept learning, state construction and behavior learning).

Table 8
Average convergence time and number of episodes for each experiment of *Stability of objects* with ADC (with concept learning, state construction and behavior learning) (*G=General, S=Singleton, H=Hierarchical* concepts).

Experiment	Time (h:min:s) AVG (SD)	No. episode AVG (SD)	No. states AVG (SD)	No. concepts {G, S, H} AVG (SD) {AVG (SD), AVG (SD), AVG (SD)}
ADC: Extrinsic + Intrinsic	04:07:42 (01:30:53)	13 (3)	11 (2)	82 (8) {34 (13), 41 (5), 51 (6)}

by the robot (*fixed, fix_aux*), 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 (*movable, move_aux*).

Examples of a behavior, concepts and states descriptions formed with those concepts learned by ADC in *stability* of objects are presented in Table 11. The robot learned to put objects (yellow spheres and red square objects) on top of square blue objects, discovering that the most stable base to put an object is one with a square shape. The states describe *the robot in front of the different objects* (yellow sphere, red square and blue square; *state(10), state(8), state(4), state(1), state(5)* using the discovered concepts *yellow_sphere, blue_square, red_square, the robot grabbing different objects* (states *state(6), state(3), state(4), state(1), state(5)* using concepts *grabbing_yellow, grabbing_red*, and *an object on the top of another object* (states *state(11), state(12)* using concepts *yellow_ontop_blue, red_ontop_blue*.

As in the mobility domain, ADC learned additional useful concepts and states, in this case related with stability. ADC could describe two types of stable structures (*yellow_ontop_blue, red_ontop_blue*), and a set of concepts and states describing the situations in which an object can be put on the top of another object. In this experiment, not all the possibilities were discovered, e.g., placing objects on top of the red square, as they were not frequent in the graphs constructed during this experiment.

ADC was able to learn concepts about movable and fixed objects, and stable structures, as other ILP systems as Hyper (Leban et al., 2008), but also learned behavior policies during the exploration of an environment. However, some expected concepts were not possible to be learned because of the initial background knowledge of the agent. The experimental results indicate that the learning process is more consistent and faster with intrinsic motivation (in learning time and number of concepts). But as expected, learning concepts and constructing the relational representation of states during exploration can increase the learning process time.

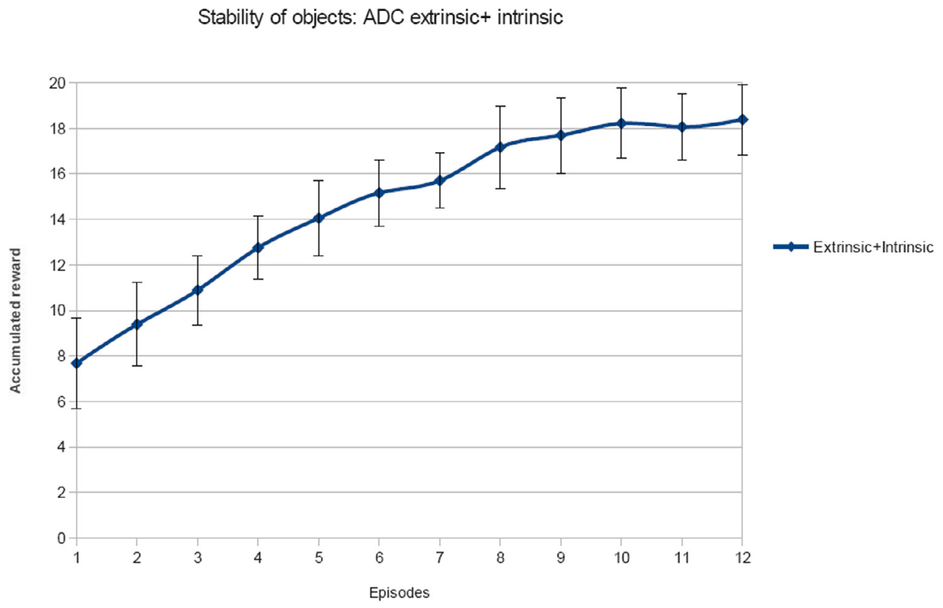

Fig. 7. Learning curves with the ADC algorithm (extrinsic and intrinsic rewards) in *Stability of objects* (with concept learning, state construction and behavior learning).

Table 9
Mobility of objects (Extrinsic + Intrinsic). Learned action, state descriptions where actions were learned and concept definitions which are used to describe the states. Concepts were named manually.

Macro actions:
% The robot learned to go to the right, then go left and push to discover the green wall. The actions go_right*and* go_left*means that the robot makes a 90 degrees turn and then moves forward.*
do_Action(4):- state(1), go_right(), state(3), go_left(), state(8), push(), state(9).
New states:
state(1):- blue_wall(A,B,C,blue,D,E,F).
state(3):- yellow_wall(A,B,C,yellow,D,E,F).
state(8):- red_object(A,B,C,red,D,E,F).
state(9):- green_wall(A,B,C,green,D,E,F).
Discovered concepts:
% Objects in front of the robot:
blue_wall(A,B,C,blue,D,E,F):- object(A,B,C,blue), robot(D,E,F), in_front_of_me(A,D).
yellow_wall(A,B,C,yellow,D,E,F):- object(A,B,C,yellow), robot(D,E,F), in_front_of_me(A,D).
red_object(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).
green_wall(A,B,C,green,D,E,F):- object(A,B,C,green), robot(D,E,F), in_front_of_me(A,D).

Table 10
Mobility of objects (Extrinsic + Intrinsic). Examples of definitions of concepts learned by ADC during the exploration process. For presentation purposes, the number of arguments of previously learned concepts used in the definition of new concepts is reduced to one. Concepts were named manually.

Concepts	Description
movable(St0,St1):- move_aux(St0), push(St0,St1), move_aux(St1).	*The robot pushes a red object, the robot does not change its position, the object changes its position, the object is movable*
move_aux(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 changes its position*
fixed(St0,St1):- fix_aux(St0), push(St0,St1), fix_aux(St1).	*The robot tries to push a blue object, the robot and the object do not change its position, the object is fixed*
fix_aux(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*

5. Conclusions

In this paper the system ADC for concept discovery and learning behaviors was described. Unlike other approaches, ADC is able to learn concepts and behaviors simultaneously, perform predicate invention discovering examples of concepts directly from the environment, and drive its own learning process. In particular, ADC decides what constitutes an example of a potential relevant concept, by identifying frequent structures in an incremental graph-based representation. It decides how to explore its environment using a modified Wundt's curve and biased actions. It discovers interesting concepts using an ILP algorithm over repeated similar descriptions, can incorporate new state descriptions during the learning process, and it learns sequences of actions that solve particular tasks. The algorithm was evaluated in a simulated robotic environment. It was shown that the IM function used in ADC is useful to guide and speed up the learning process. ADC was able to discover useful concepts and characterize with them states to solve tasks given by the user. However, the learning of concepts and intrinsic motivation can also, however, delay the learning process and produce superfluous discoveries.

There are several important differences between traditional Machine Learning approaches and the approach proposed in this paper:

- Efficiency: Although the total learning time of our experiments may seem long, the agent was able to learn how to perform tasks in between 10 and 20 episodes, which makes it an efficient learning framework.
- Discovery: Our approach incrementally discovers new concepts and actions while exploring its environment. This is a different approach to more standard machine learning techniques where the examples and goals are given to the system.
- Incrementality: The concepts are incrementally discovered while exploring the environment. This means that even if we

Table 11
Stability of objects (Extrinsic + Intrinsic). Learned actions, state descriptions where actions were learned and concept definitions which are used to describe the states. Concepts were named manually.

```
Macro actions:
% The robot learned to grab round/square yellow/red objects, turn to the right and
   when it was in front of a blue square object, put the object on the top of the blue
   object.
do_Action(3):- state(10), grab, state(6), turn_right, state(4), put, state(11).
do_Action(4):- state(8), grab, state(3), turn_right, state(1), turn_right,
state(5), put, state(12).
New states:
state(10):- yellow_sphere(A,round,yellow,B).
state(6):- grabbing_yellow(A,round,yellow,B).
state(4):- grabbing_yellow(A,round,yellow,B), blue_square(A,square,blue,B).
state(11):- yellow_ontop_blue(A,round,yellow,B,square,blue).
state(8):- red_square(A,square,red,B).
state(3):- grabbing_red(A,square,red,B).
state(1):- yellow_sphere(A,round,yellow,B), grabbing_red(A,square,red,B).
state(5):- blue_square(A,square,blue,B), grabbing_red(A,square,red,B).
state(12):- red_ontop_blue(A,square,red,B,square,blue).
Discovered concepts:
% Objects in front of the robot:
yellow_sphere(A,round,yellow,B):-
 robot(B), object(A,round,yellow), in_front_of(B,A).
blue_square(A,square,blue,B):-
 robot(B), object(A,square,blue), in_front_of(B,A).
red_square(A,square,red,B):-
 robot(B), object(A,square,red), in_front_of(B,A).
% Robot is grabbing one object:
grabbing_yellow(A,round,yellow,B):-
 robot(B), object(A,round,yellow), grabbing(B,A).
grabbing_red(A,square,red,B):-
 object(A,square,red), robot(B), grabbing(B,A).
% Object on top of another object:
yellow_ontop_blue(A,round,yellow,B,square,blue):-
 object(A,round,yellow), object(B,square,blue), on(A,B).
red_ontop_blue(A,square,red,B,square,blue):-
 object(A,square,red), object(B,square,blue), on(A,B).
```

interrupt the learning process, we can still obtain useful intermediate concepts related to the learning task.

- Re-usability: The incremental approach also allows the system to re-use previously learned knowledge, so once a concept has been learned it can be used in new tasks if necessary.
- Interpretability: The knowledge learned by our system is expressed as clausal definitions, which are easy to understand.

As future work we need to design efficient methods to filter out irrelevant learned concepts. We can provide the system with more robust action descriptions to improve the exploration of the environment (reducing errors and speeding up the learning process). Also, we would like to perform more tests and evaluate the proposed algorithm in other more challenging environments and provide mechanisms to continue learning over longer periods of time.

Acknowledgments

This work was done under partial support from CONACYT, Mexico (Ph.D. scholarship 224491).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.eswa.2017.09.023

References

Aldebaran-Robotics (2012). *Who is NAO?*. https://www.ald.softbankrobotics.com/en/cool-robots/nao.

Baldassarre, G., & Mirolli, M. (2013). *Intrinsically motivated learning in natural and artificial systems*. Berlin Heidelberg: Springer-Verlag. SpringerLink : Bücher https://link.springer.com/book/10.1007%2F978-3-642-32375-1#about

Berlyne, D. (1960). *Conflict, arousal and curiosity*. New York: McGraw-Hill.

Bonarini, A., Lazaric, A., Restelli, M., & Vitali, P. (2006). Self-development framework for reinforcement learning agents. In *Proceedings of the fifth international conference on development and learning (ICDL)*, Indiana, USA.

Bureau, A., & Sebag, M. (2014). Bellmanian bandit network. In *Proceedings of the workshop in autonomously learning robots at nips*, Montreal, Canada.

Chien, B., Hu, C., & Ju, M. (2009). Learning fuzzy concept hierarchy and measurement with node labeling. *Information Systems Frontiers, 11*, 551–559.

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

Cook, D., Holder, L., & Djoko, S. (1996). Scalable discovery of informative structural concepts using domain knowledge. *IEEE Expert, 11*, 59–68.

Cook, S. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third ACM symposium on theory of computing* (pp. 151–158). New York, NY, USA: ACM.

Davis, J., Berg, E., Page, D., Costa, V. S., Peissig, P., & Caldwell, M. (2011). Discovering latent structure in clinical databases. In *Proceedings from NIPS 2011 workshop: From statistical genetics to predictive models in personalized medicine*, Granada, Spain.

Driessens, K., & Džeroski, S. (2004). Integrating guidance into relational reinforcement learning. *Machine Learning, 57*(3), 271–304.

Džeroski, S., Raedt, L. D., & Driessens, K. (2001). Relational reinforcement learning. *Machine Learning, 43*(1), 7–52.

Eppstein, D. (1999). Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications, 3*(3), 1–27.

Fu, L., & Buchanan, B. (1985). Learning intermediate concepts in constructing a hierarchical knowledge base. In *Proceedings of the ninth international joint conference on artificial intelligence: 1* (pp. 659–666). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Georgeon, O. L., Marshall, J., & Gay, S. (2012). Interactional motivation in artificial systems: Between extrinsic and intrinsic motivation. In *Proceedings of the international conference on development and learning (ICDL) and the international conference on epigenetic robotics (EPIROB)* (pp. 1–2). San Diego, USA: IEEE.

Holder, L., Cook, D., & Djoko, S. (1994). Substructure discovery in the subdue system. In *Proceedings of the AAAI workshop on knowledge discovery in databases* (pp. 169–180). Seattle, USA: AAAI Press.

Kaplan, F., & Oudeyer, P. (2003). Motivational principles for visual know-how development. In *Proceedings of the third international workshop on epigenetic robotics : Modeling cognitive development in robotic systems, Boston, USA* (pp. 73–80).

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

Leban, G., Zabkar, J., & Bratko, I. (2008). An experiment in robot discovery with ILP. In F. Zeleznỳ, & N. Lavrac (Eds.), *Inductive logic programming*. In *Lecture Notes in Computer Science: 5194* (pp. 77–90). Berlin, Heidelberg: Springer.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research, 17*(39), 1–40.

Li, N., Stracuzzi, D., & Langley, P. (2008). Learning conceptual predicates for teleoreactive logic programs. In *Proceedings of the late-breaking papers track at the eighteenth international conference on inductive logic programming, Prague, Czech Republic* (pp. 75–80).

Martínez, D., Alenya, G., & Torras, C. (2015). *Relational reinforcement learning with guided demonstrations: 247* (pp. 295–312). Special issue on AI and Robotics, Elsevier Science Publishers Ltd.

Merrick, K., & Maher, M. (2017). Modelling motivation as an intrinsic reward signal for reinforcement learning agents. Available in: www.researchgate.net/publication/265040096_Modelling_Motivation_as_an_Intrinsic_Reward_Signal_for_Reinforcement_Learning_Agents.

Merrick, K., & Mahler, M. (2013). Novelty and beyond: Towards combined motivation models and integrated learning architectures. In *Intrinsically motivated learning in natural and artificial systems* (pp. 235–259). Berlin Heidelberg: Springer-Verlag.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *Proceedings of the NIPS deep learning workshop*.

Morales, E. F. (2004). Relational state abstractions for reinforcement learning. In *Proceedings of the workshop on relational reinforcement learning, Alberta, Canada* (pp. 27–32).

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing, 8*(4), 295–318.

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

Muggleton, S., Lin, D., Pahlavi, N., & Tamaddoni-Nezhad, A. (2014). Meta-interpretive learning: application to grammatical inference. *Machine Learning, 94*(1), 25–49.

Muggleton, S., Lin, D., & Tamaddoni-Nezhad, A. (2015). Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning, 100*(1), 49–73.

Nguyen, S., & Oudeyer, P. (2014). Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots, 36*(3), 273–294.

Nickles, M., & Rettinger, A. (2014). Interactive relational reinforcement learning of concept semantics. *Machine Learning, 94*(2), 169–204.

Rivest, R., & Sloan, R. (1994). A formal model of hierarchical concept learning. *Information and Computation, 114*, 88–114.

Rosca, J. (1997). *Hierarchical learning with procedural abstraction mechanisms*. Rochester, NY, USA: Department of Computer Science, The College of Arts and Sciences, University of Rochester Ph.D. thesis).

Russell, S., & Norvig, P. (2008). *Inteligencia artificial. Un enfoque moderno*. Pearson. Prentice Hall.

Saunders, R., & Gero, J. (2001). Designing for interest and novelty - motivating design agents. In *Proceedings of the ninth international conference on computer aided architectural design futures* (pp. 725–738). Dordrecht, Netherlands: Kluwer.

Schmidhuber, J. (1990). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the first international conference on simulation of adaptive behavior on from animals to animats* (pp. 222–227). MA, USA: MIT Press.

Simsek, O., & Barto, A. G. (2006). An intrinsic reward mechanism for efficient exploration. In *Proceedings of the twenty-third international conference on machine learning* (pp. 833–840). NY, USA: ACM.

Singh, S., Barto, A., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In *Proceedings of advances in neural information processing systems 17 (NIPS)* (pp. 1281–1288). MA, USA: MIT Press.

Singh, S., & Sutton, R. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning, 22*(1), 123–158.

Sowa, J. (2008). Conceptual graphs. In L. Harmelen, & Porter (Eds.), *Handbook of knowledge representation* (pp. 213–237)). Elsevier B. V.

Stahl, I. (1996). Predicate invention in inductive logic programming. In L. D. Raedt (Ed.), *Advances in inductive logic programming* (pp. 34–47). Ohmsha, Amsterdam: IOS Press.

Stanley, K., & Domingos, P. (2007). Statistical predicate invention. In *Proceedings of the twenty-fourth annual international conference on machine learning, New York, NY, USA* (pp. 433–440).

Sutton, R., & Barto, A. (1998). *Introduction to Reinforcement Learning* (1st). Cambridge, MA, USA: MIT Press.

Tani, J., & Nolfi, S. (1999). Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems. *Neural Networks, 12*, 1131–1141.

Tenorio, A. (2017). Automatic discovery of concepts. https://sites.google.com/site/automaticdiscoveryconcepts/.

Tenorio-González, A., & Morales, E. (2016). Automatic discovery of relational concepts by an incremental graph-based representation. *Robotics and Autonomous Systems, 83*(C), 1–14.

Zhang, Y., & Weng, J. (2002). 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, Edinburgh, Scotland* (pp. 47–55).

Zupan, B., Bohanec, M., Bratko, I., & Demšar, J. (1999). Learning by discovering concept hierarchies. *Artificial Intelligence, 109*, 211–242.