Annals of the University of Craiova, Math. Comp. Sci. Ser.
Volume 34, 2007, Pages 124–132
ISSN: 1223-6934

Relational Reinforcement Learning: A Logic Programming based approach

Mircea Cezar Preda

Abstract. A method to combine reinforcement learning with logical descriptions for states and actions is presented. The method uses a distance measure between epistemic logic programs to evaluate the similarities between different state - action pairs and to generalize between them. Due to the use of a more expressive representation language to represent states and actions, reinforcement learning can be applied to a wider range of tasks.

2000 Mathematics Subject Classification. 68T05, 68T27.
Key words and phrases. Reinforcement learning, relational reinforcement learning, epistemic logic programming, distances between logic programs.

1. Introduction

Most representations used in reinforcement learning are not proper to describe problems that require logical representations for states and actions. This generated the relational reinforcement learning domain which study the combination between reinforcement learning and logic programming. The first paper in this field [2] used a relational regression tree to learn a Q-function from sampled traces. This paper presents a method to integrate reinforcement learning and epistemic logic programs by defining a distance measure between epistemic logic programs. The distance framework is highly customizable to the requirements of the particular problems.

2. Reinforcement learning problem

Let M be a Markov decision problem with a set S of states, a set A of actions, a one-step reward function $r : S \times A \to \mathbb{R}$ and a transition function $T : S \times A \times S \to [0, 1]$. $T(s_1, a, s_2)$ represents the probability to transit from state s_1 into state s_2 following action a. The reward for a trajectory in M is the discounted sum of all of its one-step rewards.

$$R = \sum_t \gamma^t r_t$$

where r_t is the reward received at the step t and $0 \leq \gamma \leq 1$ is the *future reward* discount factor. A policy π is a function that maps states to probability distributions over actions. Our aim is to compute an optimal policy, a policy which on average generates trajectories with the highest possible reward. In order to do that, the following notions will be stated. Define $Q^*(s, a)$ to be the highest total expected

Received: December 5, 2007.
The author has been supported by CNCSIS Grant 27GR/11.05.2007 Cod CNCSIS 102/2007.

reward that can be obtained by starting in state s, performing action a, and acting optimally afterwards. Knowledge of Q^* allows to determine an optimal policy

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s, a).$$

Usually, the above presented Markov decision problem is resolved by constructing an approximation to Q^*. If the sets S and A are finite, the values of the function Q^* can be accurately computed by successively updates of a table with an entry for every state - action pair. But the tabular variant does not allow to transfer information from the already known state - action pairs to a new encountered one. By changing the function approximation method, it will be possible to benefit of the generalization capabilities of some advanced approximation methods and to transfer knowledge between closely related state - action pairs. Due to this knowledge transfer, the approximation will converge faster to the real Q^* function and a control component based on this approximation will be efficient from the early stages of the control interaction.

$\overline{Q}(s, a)$ will denote the approximation of $Q^*(s, a)$ for $s \in S$ and $a \in A$. If a symbolic domain is targeted, there is one possible problem: most of function approximation methods were devised to work for functions with real numbers arguments and there are cases when states and actions cannot be immediately converted to arrays of real numbers. Logic programming is a powerful knowledge representation mechanism that is easy to be understood and manipulated both by humans and computers. In particular, logic programs may describe states and transformations between logic programs may describe actions. To ensure the generality, the epistemic logic programs will be used. These are programs with high representation power that subsume weaker forms of representation like general logic programs. In following section it is presented a framework where the approximation $\overline{Q}(s, a)$ is defined to be $\overline{Q}(s, a) = \hat{Q}(g(s, a))$ where $g : S \times A \to \mathbb{R}^n$ transforms state - action pairs represented by epistemic logic programs into arrays of real numbers and $\hat{Q}$ is a linear function of some parameters w. In accordance with [3], the following proposition can be proved:

Proposition 2.1. *If $M =< S, A, T, r >$ satisfies the next properties*
- *S and A are finite sets, all states of S are reachable with a positive probability and all transitions have a finite number of steps;*
- *The approximation $\overline{Q}$ is maintained by using the trajectory based version of SARSA(λ), $\lambda > 0$ reinforcement learning algorithm* [5]*. This version changes the policies only between trajectories by selecting at the begining of each trajectory of the ϵ-greedy policy for its current $\overline{Q}$ function;*
- *g is an injective function*

then the weights vector w will converge with probability 1 to a bounded region.

In order to benefit of the generalization capabilities of the linear approximation $\hat{Q}$, it is required that g to be a distance like function on the state - action pairs space. In the next paragraphs, possibilities to define functions like g will be evaluated.

3. Distances between state-action pairs

Previously, it was stated that the states and actions involved by the reinforcement learning framework will be represented by logic programs. Consequently, if a distance between logic programs can be devised, then this distance can be used to compare how close are two state-action pairs.

3.1. Similarity measure between logic programs. Two logic programs are considered similar if they provide same answers to the most questions. Two logic programs are logically equivalent if they provide identical answers to every question. Usually, the answers provided by logic programs are sets of literals, so a similarity measure for logic programs can be defined specifying similarity measures for literals and sets of literals. A discussion on this topic can be found in [4]. Most of the existent approaches cannot adapted to the particularities of a specific problem. Therefore, they are not effective for a large class of applications that require customization according with their domain's insides. In the next paragraphs, it will be presented an intuitive and highly configurable framework for measuring similarities between sets of literals. The similarities between two literals are given by the similarities between their structures. Directed graphs will be used to represent the elements and relations involved by a structure.

Let us start by presenting the graph representation of a literal. The following assumption that does not limit the generality is made: only ground instantiations of logical programs are used and all literals are considered ground. From now, the attribute ground will be eliminated from descriptions.

Let $\mathcal{F}$ be a finite set of function symbols and $\mathcal{P}$ a finite set of predicate symbols. A term is of the form $f(t_1, ..., t_n)$, $n \geq 0$ with f/n a functor with arity n and t_1, ..., t_n terms. Let Γ_t be the graph attached to the term $t = f(t_1, ..., t_n)$. Γ_t is described by the rules:

(1) If f is a function symbol with arity 0 (a constant) then Γ_t has only one node (the "root") labeled with f.
(2) If f is a function symbol with arity > 0 then Γ_t is composed by a node labeled with $f(t_1, ..., t_n)$ connected by arcs to a node labeled with f and to the "root" nodes of the graphs $\Gamma_{t_1}, ..., \Gamma_{t_n}$. These directed edges are labeled by numbers from 0 to n.
(3) Γ_t cannot include two nodes with identical labels. Each node from Γ_t is uniquely identified by its label.
(4) Γ_t cannot include two arcs with identical sources, destinations and labels. An arc is identified by a triple (s, t, e) where s is the label of the source node, t is the label of the destination node and e is the arc label.

An atom is of the form $p(t_1, ..., t_n)$, $n \geq 0$ with p/n an n-ary predicate symbol and t_1, ..., t_n terms. Let Γ_a be the graph attached to the atom $a = p(t_1, ..., t_n)$. Γ_a is described by the following rules:

(1) If p is a predicate symbol with arity 0 then Γ_a consists from only one node labeled by p.
(2) If p is a predicate symbol with arity > 0 then Γ_a consists from a node labeled with $p(t_1, ..., t_n)$ that is connected by arcs to a node labeled with p and to the "root" nodes of the graphs $\Gamma_{t_1}, ..., \Gamma_{t_n}$. These arcs are labeled with numbers from 0 to n.

Γ_a can be represented as a pair $\Gamma_a = (V_a, E_a)$ where V_a and E_a are the sets of nodes (vertices) and, respective, edges from Γ_a.

A literal l is an atom $l = p(t_1, ..., t_n)$ or is the classical negation of an atom $l = \neg p(t_1, ..., t_n)$. Let $\mathcal{P}'$ be a finite set of predicate symbols, $\mathcal{P} \cap \mathcal{P}' = \emptyset$ and $cn : \mathcal{P} \to \mathcal{P}'$ an injective function. For $\mathcal{P}'$ and cn fixed, Γ_l, the graph attached to a literal l, is defined as:

$$\Gamma_l = \begin{cases} \Gamma_{p(t_1,...,t_n)} & \text{if } l = p(t_1, ..., t_n) \\ \Gamma_{p'(t_1,...,t_n)} & \text{if } l = \neg p(t_1, ..., t_n) \text{ and } p' = cn(p). \end{cases}$$

It can be observed that the classical negation was handled by using the concept of positive form. The positive form is constructed by using the function *cn* every time when the classical negation appears.

A set of literals is reduced to a set of atoms by using *cn*. $\mathcal{A}_g$ describes the family of the all atoms constructed using $\mathcal{F}$ and $\mathcal{P} \cup \mathcal{P}'$. $\mathbb{A}_g = 2^{\mathcal{A}_g}$ is the collection of the all subsets of $\mathcal{A}_g$.

Let $\mathbb{V} = \{V_1, V_2, ..., V_n\}$ be a family of sets of elements that can be found in the graphs associated to the atoms from $\mathcal{A}_g$. For example, a set $V_i, i \in \overline{1,n}$ can be a set of node labels or a set of arc labels. Or, V_i can be a set of subgraphs. $\mathbb{V}$ will act as a collection of criteria devised to study similarities between two sets of graphs. Let us suppose that for every $v \in V_i, i \in \overline{1,n}$, exists a function $f(v) : \mathbb{A}_g \to \mathbb{N}$ that satisfies the following properties:

a) $f(v)(A) \geq 0, \forall A \subseteq \mathcal{A}_g$;
b) $f(v)(\emptyset) = 0$;
c) If $A_{i,i\geq 0} \subseteq \mathbb{A}_g, \forall i, j : A_i \cap A_j = \emptyset$ then $f(v)(\cup_i A_i) = \sum_i f(v)(A_i)$.

$f(v), v \in V_i, i \in \overline{1,n}$ are pseudo measures on the universe $\mathcal{A}_g$. They are not measures because the property $f(v)(A) = 0 \Rightarrow A = \emptyset, \forall A \subseteq \mathcal{A}_g$ is not imposed.

Example: If $v \in V_i, i \in \overline{1,n}$ is a node label then $f(v)$ can be defined as

$$f(v)(A) = |\{V_a | \exists a \in A, \Gamma_a = (V_a, E_a), v \in V_a\}|. \tag{1}$$

$f(v)(A)$ represents the number of occurrences of the node labeled by v in the set of graphs attached to the ground atoms from A. By $|.|$, it is denoted the cardinal function for sets. Similarly, if v is an arc label, $f(v)$ can be defined as

$$f(v)(A) = |\{E_a | \exists a \in A, \Gamma_a = (V_a, E_a), v \in E_a\}|,$$

which is the number of occurrences of the arc label v in the graphs $\Gamma_a, a \in A$.

Let $v \in V_i, i \in \overline{1,n}$. The relation $=_v \subseteq \mathbb{A}_g \times \mathbb{A}_g$ defined by $A =_v B$ if and only if $f(v)(A) = f(v)(B)$ is an equivalence relation (reflexive, symmetrical and transitive).

Let $d : \mathbb{A}_g \times \mathbb{A}_g \to \mathbb{R}_+$. d is a $=_v$-distance defined on $\mathbb{A}_g$ if and only if the following three properties are satisfied:

a) $d(A, B) \geq 0$, $d(A, B) = 0 \Leftrightarrow A =_v B, \forall A, B \subseteq \mathcal{A}_g$,
b) $d(A, B) = d(B, A), \forall A, B \subseteq \mathcal{A}_g$,
c) $d(A, B) \leq d(A, C) + d(C, B), \forall A, B, C \subseteq \mathcal{A}_g$.

d is a $=_{\mathbb{V}}$-distance if and only if d is $=_v$ - distance $\forall v \in V_i, \forall i \in \overline{1,n}$. If d is a $=_{\mathbb{V}}$ - distance then the equivalence $d(A, B) = 0 \Leftrightarrow A =_v B, \forall v \in V_i, \forall i \in \overline{1,n}$ is true.

In these conditions, the following proposition is satisfied:

Proposition 3.1. *Let $\{d(v) : \mathbb{A}_g \times \mathbb{A}_g \to \mathbb{R}_+, =_v$ -distance$|v \in V_i\}$ be n families of $=_v$ - distances defined for the all subsets of ground atoms. If $\mathbb{V}$ includes only finite sets, the functions $d_a^p : \mathbb{A}_g \times \mathbb{A}_g \to \mathbb{R}_+^n$ defined by*

$$d_a^p(A, B) = (d_1^p(A, B), ..., d_n^p(A, B)) \in \mathbb{R}_+^n \tag{2}$$

where

$$d_i^p(A, B) = (\sum_{v \in V_i} d(v)(A, B)^p)^{1/p}, \forall i \in \overline{1,n}, \forall p \in \mathbb{N}^* \tag{3}$$

and

$$d_i^\infty(A, B) = \max_{v \in V_i} \{d(v)(A, B)\}, \forall i \in \overline{1,n}, p = \infty \tag{4}$$

are $=_{\mathbb{V}}$ - distances defined on the family of the all subsets of ground atoms $\forall p \in \mathbb{N}^* \cup \{\infty\}$.

Remark 3.1. *If the set* $\mathbb{V}$ *satisfies the property that* $\forall A, B \subseteq \mathcal{A}_g : A =_v B, \forall v \in V_i, \forall i \in \overline{1,n}$ *infer* $A = B$ *then the functions* d_a^p *are distances (metrics),* $\forall p \in \mathbb{N}^* \cup \{\infty\}$.

Proposition 3.2. *Each of the distances* $d_a^p, p \in \mathbb{N}^* \cup \{\infty\}$ *are similar in sense that for any* $p, q \in \mathbb{N}^* \cup \{\infty\}$, $\exists a, b \in \mathbb{R}$ *such that* $a \cdot d_a^q(A,B) \leq d_a^p(A,B) \leq b \cdot d_a^q(A,B), \forall A, B \subseteq \mathcal{A}_g$.

Example: If $k = \max_{i \in \overline{1,n}}\{|V_i|\}$, then $d_a^\infty(A,B) \leq d_a^2(A,B) \leq \sqrt{k} \cdot d_a^\infty(A,B)$ and $d_a^\infty(A,B) \leq d_a^1(A,B) \leq k \cdot d_a^\infty(A,B), \forall A, B \subseteq \mathcal{A}_g$.

Proposition 3.3. *Fix* $v \in V_i, i \in \overline{1,n}$. *If* $f(v) : \mathbb{A}_g \to \mathbb{N}$ *is a pseudo - measure on the universe* $\mathcal{A}_g$ *then the functions* $d(v) : \mathbb{A}_g \times \mathbb{A}_g \to \mathbb{R}_+$ *defined by*

$$d(v)(A,B) = |f(v)(A) - f(v)(B)|, \tag{5}$$

$$d(v)(A,B) = \frac{|f(v)(A) - f(v)(B)|}{\max\{f(v)(A), f(v)(B)\}}, \tag{6}$$

$$d(v)(A,B) = \frac{|f(v)(A) - f(v)(B)|}{f(v)(A \cup B)}, \tag{7}$$

$$d(v)(A,B) = f(v)((A \setminus B) \cup (B \setminus A)), \tag{8}$$

$$d(v)(A,B) = \frac{f(v)((A \setminus B) \cup (B \setminus A))}{f(v)(A \cup B)}. \tag{9}$$

are $=_v$ *- distances on* $\mathbb{A}_g$.

3.2. Representing epistemical information. Two unary operators K and M are used [1]. Intuitively, if l is a literal then Kl represents l *is known* and Ml, l *may be believed*. The constructions Kl, Ml, $\neg Kl$ and $\neg Ml$, where l is a literal, are called subjective literals. An epistemic logic program is a finite set of rules

$$l_1 \; or ... or \; l_k \leftarrow g_1, ..., g_m, not \; h_1, ..., not \; h_n$$

where $l_1, ..., l_k$ and $h_1, ..., h_n$ are literals, $g_1, ..., g_m$ are subjective literals or literals and $k \geq 1$, $m, n \geq 0$.

Let P be an epistemic program and W a collection of sets of literals, $W \subseteq \mathbb{A}_g$. $\cup W = \bigcup_{L \in W} L$ denotes the set of the literals that may be believed regarding to W and $\cap W = \bigcap_{L \in W} L$ the set of the literals that are known. P^W represents the disjunctive logic program [1] obtained from P by: a) removing all rules containing subjective literals g that are not entailed by W and b) removing all other subjective literals from rules in P. W is named a world view of P if W is the collection of all answer sets of P^W. Two epistemic programs P_1, P_2 are considered to be similar if their world views are similar. The attached world views give the semantic of an epistemic program.

Let $m : 2^{\mathbb{A}_g} \times 2^{\mathbb{A}_g} \to 2^{\mathcal{R}}, \mathcal{R} = \{r | r \subseteq 2^{\mathbb{A}_g} \times 2^{\mathbb{A}_g}\}$ be a function that associates to every pair $W_1, W_2 \subseteq \mathbb{A}_g$ a family of binary relations on the collection of subsets of atoms with the property $\forall r \in m(W_1, W_2), r \subseteq 2^{W_1} \times 2^{W_2}$. The functions $d_w^{m,p} : 2^{\mathbb{A}_g} \times 2^{\mathbb{A}_g} \to \mathbb{R}_+^n$, $d_w^{m,p}(W_1, W_2) = \min_{r \in m(W_1,W_2)}\{(\sum_{(W_1', W_2') \in r}(d_a^p(\cap W_1', \cap W_2') + d_a^p(\cup W_1', \cup W_2')))/|r|\}, p \in \mathbb{N}^* \cup \{\infty\}$ are, in general, pseudo distances because they do not satisfy the triangle inequality for every m. If m is set to be $m^*(W_1, W_2) = \{\{ (W_1, W_2)\}\}$, $d_w^{m^*,p}$ are distances and can be used to compare two world views. These are compared regarding the literals that are known or may be believed.

Let Π be the set of epistemic logic programs that have an unique world view. In these conditions, $d_e^p : \Pi \times \Pi \to \mathbb{R}_+^n$, $d_e^p(P_1, P_2) = d_w^{m^*,p}(W_{P_1}, W_{P_2})$, where W_P is the world view attached to the epistemic program P and $p \in \mathbb{N}^* \cup \{\infty\}$, are $\sim_{\mathbb{V}}$ -

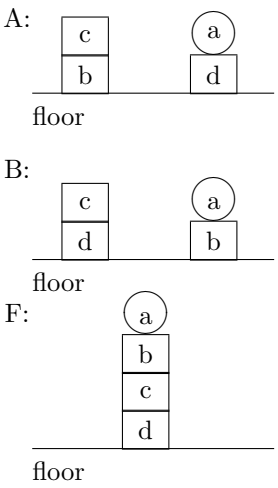

FIGURE 1. Two initial states (A and B) and the terminal one (F) of the experimental blocks world.

distances on Π. $\sim_{\mathbb{V}} \subseteq \Pi \times \Pi$ is the equivalence relation $P_1 \sim_{\mathbb{V}} P_2 \Leftrightarrow \cap W_{P_1} =_{\mathbb{V}} \cap W_{P_2}$ and $\cup W_{P_1} =_{\mathbb{V}} \cup W_{P_2}$.

4. Application

To illustrate the above presented method, let us consider a blocks world with 4 elements: a ball denoted by a and three cubes denoted by b, c, d. An element can be put on top of another element if the last one is clear or can be put directly on the floor, which is denoted by f. There is one restriction, the ball cannot hold any other object. The figure 1 presents three possible configurations of this world. The aim is to find the shortest sequences of movements to transfer the world from an initial state like A or B to the final state F. The optimal paths are depicted in the figures 2 and 3. Each state of the world will be described by a logical program. The following three represent the states A, B, F:

$$P_A : \begin{cases} on(b,f) & \leftarrow & \\ on(d,f) & \leftarrow & \\ on(c,b) & \leftarrow & \\ on(a,d) & \leftarrow & \\ clear(c) & \leftarrow & \\ over(X,Y) & \leftarrow & on(X,Y) \\ over(X,Y) & \leftarrow & on(X,Z), over(Z,Y) \end{cases},$$

$$P_B : \begin{cases} on(b,f) & \leftarrow & \\ on(d,f) & \leftarrow & \\ on(c,d) & \leftarrow & \\ on(a,b) & \leftarrow & \\ clear(c) & \leftarrow & \\ over(X,Y) & \leftarrow & on(X,Y) \\ over(X,Y) & \leftarrow & on(X,Z), over(Z,Y) \end{cases},$$

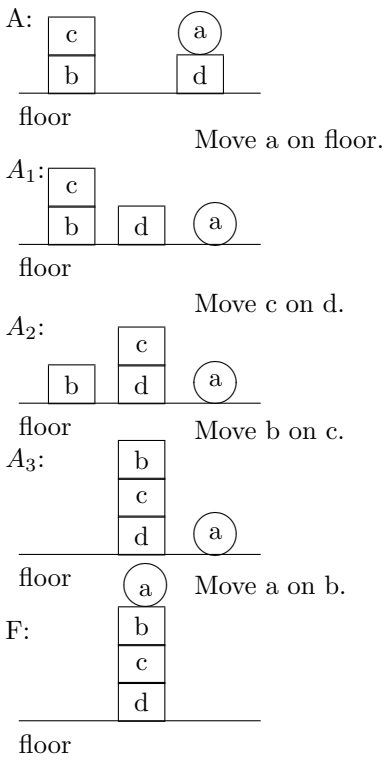

FIGURE 2. The optimal movements path from initial state A to final state F.

$$P_F : \begin{cases} on(d,f) & \leftarrow & \\ on(c,d) & \leftarrow & \\ on(b,c) & \leftarrow & \\ on(a,b) & \leftarrow & \\ over(X,Y) & \leftarrow & on(X,Y) \\ over(X,Y) & \leftarrow & on(X,Z), over(Z,Y) \end{cases}.$$

The semantics of these programs are given by the next three sets of atoms:

$$\begin{aligned} A &= \{on(b,f), on(d,f), on(c,b), on(a,d), \\ &\quad clear(c), over(c,b), over(a,d)\}, \\ B &= \{on(b,f), on(d,f), on(c,d), on(a,b), \\ &\quad clear(c), over(c,d), over(a,b)\}, \\ F &= \{on(d,f), on(c,d), on(b,c), on(a,b), \\ &\quad over(a,b), over(a,c), over(a,d), \\ &\quad over(b,c), over(b,d), over(c,d)\}. \end{aligned}$$

Atoms $over(X,f), X \in \{a,b,c,d\}$ were omitted as obvious. These sets will be used to evaluate the distances between A and F and between B and F.

The sets $\mathbb{V} = \{V_1, V_2, V_3, V_4, V_5\}$ are composed from node labels as results below:

- $V_1 = \{clear\}$ compares two states from point of view of the number of elements that are clear;

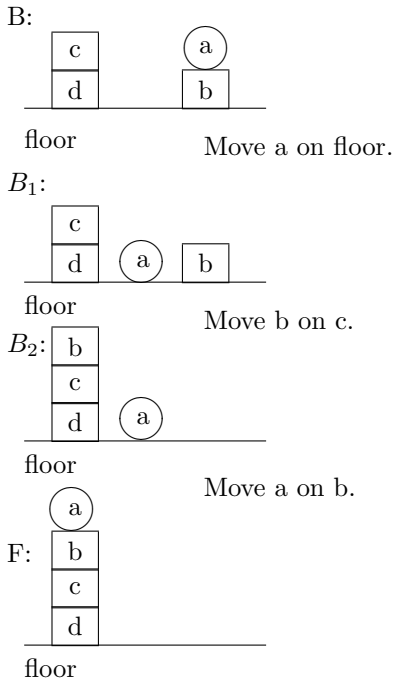

FIGURE 3. The optimal movements path from initial state B to final state F.

- $V_2 = \{clear(b)\}$, $V_3 = \{clear(c)\}$, $V_4 = \{clear(d)\}$ - the states are compared by the precise identification of the elements that are clear;
- $V_5 = \{over(a,b), over(a,c), over(a,d), over(b,c), over(b,d), over(c,d)\}$ compares two states from point of view of the relative positioning of the elements.

Following parameters were used during training episodes: the reward function was considered to be $r(s,a) = -1, \forall a \in A$, $\forall s \in S, s \neq F$ and 0 otherwise; $\overline{Q}$ was defined using tile coding (CMAC) linear function approximator and the discount rate was $\gamma = 1$. The rest of parameters received standard settings. The training was performed first with A as initial state. After 500 episodes, when the optimal path from A to F was found and the approximation $\overline{Q}$ was constructed, the initial state was transformed in B. Due to the generalization capabilities of the CMAC approximation, the system followed almost immediately the optimal path from B to F. .

5. Conclusion

A configurable framework to define distances between epistemic logic programs was presented in order to adapt a distance to the specific of a particular problem. Based on this framework, reinforcement learning could be applied to domains where the states and actions were represented by epistemic logic programs. The generalization capabilities of the reinforcement learning algorithms that use functional approximations were proved in a simple blocks' world domain.

References

[1] C. Baral and M. Gelfond, *Logic programming and knowledge representation*, Journal of Logic Programming, 19, 20, pages 73-148, 1994.
[2] S. Dzeroski and L. De Raedt and H. Blockeel, *Relational Reinforcement Learning*, Proc. ICML' 98, Morgan Kaufmann, 1998.
[3] G. J. Gordon, *Reinforcement learning with function approximation converges to a region*, Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, pages 1040-1046, 2001.
[4] J. Ramon and M. Bruynooghe, *A framework for defining distances between first logic objects*, Technical report CW 236, Katholieke Universiteit Leuven, Belgium, 1998.
[5] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT Press, Cambridge, MA, 1998.

(Mircea Cezar Preda) University of Craiova, Department of Computer Science,
13 Al. I. Cuza street, Craiova, Dolj, RO-200585, Romania
E-mail address: mpreda@central.ucv.ro