Multi-Agent Relational Reinforcement Learning

Karl Tuyls[1], Tom Croonenborghs[2], Jan Ramon[2], Robby Goetschalckx[2], and Maurice Bruynooghe[2]

[1] Theoretical Computer Science Group, Limburgs Universitair Centrum, Belgium
[2] Department of Computer Science, Katholieke Universiteit Leuven, Belgium

Abstract. In this paper we study Relational Reinforcement Learning in a multi-agent setting. There is growing evidence in the Reinforcement Learning research community that a relational representation of the state space has many benefits over a propositional one. Complex tasks as planning or information retrieval on the web can be represented more naturally in relational form. Yet, this relational structure has not been exploited for multi-agent reinforcement learning tasks and has only been studied in a single agent context so far. This paper is a first attempt in bridging the gap between Relation Reinforcement Learning (RRL) and Multi-agent Systems (MAS). More precisely, we will explore how a relational structure of the state space can be used in a Multi-Agent Reinforcement Learning context.

1 Introduction

In recent years, Relational Reinforcement Learning (RRL) has emerged in the machine learning community as a new interesting subfield of Reinforcement Learning [6, 3, 16]. It offers to reinforcement learning a state space representation that is much richer than that used in classical (or propositional) methods. More precisely, states are represented in a relational form, that more directly represents the underlying world and allows to represent complex real world tasks as planning or information retrieval on the web in a more natural manner (see section 2 for an example).

Compared to single agent reinforcement learning, learning in a MAS is a complex and cumbersome task. Typical for a MAS is that the environment is not stationary and the Markov property is not valid. These characteristics make the transition from a one-agent system to a multi-agent system very hard. Furthermore, an agent in a MAS needs to take in account that other agents are also trying to attain the highest utility for their task. A possible solution would be to provide all possible situations an agent can encounter in a MAS and define the best possible behavior in each of these situations beforehand. However, such a solution suffers from combinatorial explosion and is not the most intelligent solution in terms of efficiency and performance.

Yet different approaches have been introduced to solve this multi-agent learning problem ranging from joint action learners [9] to individual local Q-learners. All of these approaches have as well their own merits as disadvantages in learning in a multi-agent context. In the first approach, i.e. the joint action space approach, the state and action space are respectively defined as the Cartesian product of the agent's individual state and action spaces. More precisely, if S is the set of states and $A_1, ..., A_n$ the

action sets of the n different agents, the learning will be performed in the product space $S \times A_1 \times ... \times A_n$, where each agent has a reward function of the form:

$S \times A_1 \times ... \times A_n \rightarrow \mathbb{R}$. This implies that the state information is shared amongst the agents and actions are taken and evaluated synchronously. It is obvious that this approach leads to very big state-action spaces, and assumes instant communication between the agents. Clearly this approach is in contrast with the basic principles of many contemporary multi-agent applications: distributed control, asynchronous actions, incomplete information, cost of communication. The second approach totally neglects the presence of the other agents, and agents are considered to be selfish reinforcement learners. The effects caused by the other agents also acting in that same environment are considered as noise. In between these approaches we can find examples which try to overcome the drawbacks of the joint action approach, examples are [11, 1, 12, 18, 13, 14]. There has also been quite some effort to extend these RL techniques to Partially Observable Markovian decision problems and non-Markovian settings [10].

However, to our knowledge, almost all of these different techniques have been used so far in combination with a state space which is in propositional form, where by no means relations between different features are expressed or exploited. To our belief, multi-agent reinforcement learning in general could greatly benefit from the ideas of relational reinforcement learning, which has proved to be very successful in the single agent case. In this paper we do a first attempt in bridging this gap.

The rest of this document is structured as follows. Section 2 introduces relational reinforcement learning, Section 3 gives an overview of relevant existing work and Section 4 introduces the multi-agent relational reinforcement learning task. Some preliminary experiments are presented in Section 5 and finally Section 6 concludes.

2 Single Agent Relational Reinforcement Learning

Reinforcement learning Reinforcement learning offers a general framework, including several methods, for constructing intelligent agents that optimize their behavior in stochastic environments with minimal supervision. The problem task of reinforcement learning [15] using the discounted sum of rewards is most often formulated as follows: given a set of possible states S, a set of possible actions A, unknown transition probabilities t: $S \times A \times S \rightarrow [0, 1]$ and an unknown real-valued reward function $r : S \times A \rightarrow \mathbb{R}$, find a policy which maximizes the expected discounted sum of rewards $V(s_t) = \mathbb{E}\left(\sum_{i=0}^{\infty} \gamma^i r_t\right)$ for all s_t, where $0 \leq \gamma < 1$.

At every time step t, the learning agent is in one of the possible states s_t of S and selects an action $a_t = \pi(s_t) \in A$ according to his policy π. After executing action a_t in s_t, the agent will be in a new state s_{t+1} (this new state is chosen according to the transition probabilities) and receives a reward $r_t = r(s_t, a_t)$.

A drawback of most work on Reinforcement Learning, using a propositional representation, i.e. a feature vector with an attribute for every possible property of the agent's environment, is the difficulty to represent states that are defined by the objects that are present in this state and the relations between these objects. The real world contains objects. Objects with certain properties, that relate to each other. To

apply reinforcement learning in such complex environments, a structural or relational representation is needed.

To illustrate the need for these structural representations, we will describe the blocks world domain as a reinforcement learning problem. The blocks world consists of a number of blocks, which can be on the floor or onto each other. It is assumed that an infinite number of blocks can be put on the floor and that all blocks are neatly stacked onto each other, e.g. a block can only be on one other block at the same time. The possible actions consist of moving one clear block (e.g. a block with no other block on top of it) onto another clear block or onto the floor.

It is impossible to represent such blocks world states with a propositional representation without an explosion of the number of states. Using First-Order Logic, a blocks world state can be represented as a conjunction of predicates, describing the relations between the blocks, e.g. $\{on(s, b, floor) \wedge on(s, a, b) \wedge clear(s, a) \ldots\}$.

Relational Reinforcement Learning Relational Reinforcement Learning combines the RL setting with relational learning or Inductive Logic Programming (ILP). Because of this structural representation, it is possible to abstract from and generalize over specific goals, states and actions and exploit the results of previous learning phases when addressing new and possibly more complex situations.

Furthermore, because relational learning algorithms are used, there is the possibility to use background knowledge. Background knowledge consists of facts or general rules relevant to the examples or problem domain in the context of reinforcement learning. In the blocks world, a predicate like $above(S, A, B)$ could be specified in the background knowledge to define if block A is above block B in state S. These predicates in the background knowledge can be used in the learning process, i.e., in the representation of a Q-function.

Although, RRL is a relatively new domain, several approaches have been proposed during the last few years, we refer to [16] for an overview.

One of the first methods within RRL, is relational Q-learning [6]. In this work, a Q-learning algorithm is proposed that allows a relational representation for states and actions. The Q-function is represented and learned using an incremental relational regression algorithm. So far, a number of different relational regression learners are developed[3].

Besides relational Q-learning, there has been some work on other methods which is not discussed here. So far, all work on relational reinforcement learning has focused on the single agent case. To our knowledge, there is no existing work on applying relational reinforcement learning in a multi-agent system.

3 The Multi-Agent Learning problem

During the 90's multi-agent systems have become a very popular approach in solving computational problems of distributed nature as for instance load balancing or

[3] A thorough discussion and comparison can be found in [3]

distributed planning systems. They are a conceptually proved solution method for problems of this nature.

However designing a cooperative multi-agent system with both a global high utility for the system and high individual utilities for the different agents is still a difficult problem [17, 8].The joint actions of all agents derive some reward from the outside world. To enable local learning, this reward has to be divided among the individual agents where each agent aims to increase its received reward. However, unless special care is taken as to how reward is assigned, there is a risk that agents in the collective work at cross-purposes. For example, agents can reach sub-optimal solutions in the blocks world example by competing for the same block or goal state, i.e. by inefficient task distribution among the agents as they each might only consider their own goals which can result in a Tragedy of the Commons situation, or policy oscillations [12].

In this setting different researches have already obtained some very nice results within the framework of stochastic dispersion games [7, 17, 8]. Dispersion games are an abstract representation of typical load balancing and niche selection problems.The games are played repeatedly, during which the agents learn to disperse. Still, these type of problems as dispersion games are far more simple than for instance blocks world planning problems, as there is only one state to consider. We would like to extend this type of work to large planning problems, which we will try to solve by an agent-based system, consisting of learning agents in a relational state space.

4 Relational Multi-Agent Reinforcement Learning

Combining multi agent systems and relational reinforcement learning combines two complex domains. We believe that, in order to study the integration of these both settings, one should take care not to make the learning task too complex at once, as a mix up of the many different effects playing a role in both domains could make (especially experimental) results difficult to interpret. Therefore, we will first try to separate a number of effects we want to investigate as much as possible independently. In a second part of this section, we will then propose a number of settings of increasing difficulty in which we plan to conduct experiments.

Complexity factors One could describe the main complexity factors of multi agent systems as uninformedness, communication and interference. First, agents are often assumed to be unaware of parts of the world far away where the other agents are operating. This essentially makes the world only partially observable, and hence agents are less informed than in the Markovian situation. Second, agents are unaware of each other's knowledge and intentions. Though these can not be observed, these can be (partially) revealed by communication. In fact, a lot of work has been published on the study of agent communication. Third, plans and actions of agents can interfere. To act optimally, an agent should take plans and actions of other agents into account (e.g. by knowing them, or by predicting them, or by making his own plan robust).

Relational learning adds extra complexity with increased state and hypothesis spaces, generalization and informedness. First, RRL has been proposed in answer to the need to describe larger state spaces. Though relational languages allow through

their generalization ability to represent state spaces, policies, reward, transition and value functions compactly, they do not take away the fundamental problem that if the task is difficult enough, the optimal policy will be complex and the learning hardness will increase with the state space size. This is illustrated by the fact that in the single agent relational case, up to now no results exist that guarantee convergence to an optimal policy in the case generalization over the state space is performed. Second, while the generalization ability is usually beneficial, it also often has the consequence that due to the generalization the world is only partially observable, i.e. it may be difficult to see the difference between states over which the agent generalizes. Third, while a reason for introducing relational languages was the ability to introduce background knowledge and hence make the agent better informed and put him in a better situation to act intelligently, the background knowledge will only be useful when the algorithm has the ability to exploit it. This often adds an extra level of complexity to the algorithm.

We believe that in a first step, it is worthwhile to see the communication problem of the agents as an issue orthogonal to the relational nature of the language used. Communication problems will arise in the same way, independent from the choice of an internal representation language used by the agent. The same argument holds for the uninformedness of the agents in multi agent systems due to their limited sensor activity. The essential point of multi agent systems is the presence of multi agents and their interference. Therefore, we propose to investigate in a first step the combination of relational reinforcement learning with the interference of agents. Later, the other complexity factors of multi agent systems can be integrated in much the same way as is performed with classical multi agent systems. An additional motivation for this choice is that, due to the fact that relational reinforcement learning was motivated by its ability to let the agent be much more informed, we can expect to gain maximally from this extension in a situation where the agent has access to the most information.

Settings We will introduce first some terminology. A **setting** is a set of properties of the problem and the agent abilities under consideration. We say a setting has the `comm_reward` property iff the agents are trying to maximize the same common reward function (if one agent gets a reward, all other agents get the same reward). We say a setting has the `know_aim` property iff the agents know what function the other agents are trying to maximize. A setting has property `full_obs` iff the agents can observe the full world, including actions performed by other agents (but excluding internals of the other agents). A setting has property `know_abil` iff the agents know the ability of the other agents, i.e. how good they are at their task (e.g. whether an other agent will perform random actions, or whether an other agent always performs the optimal action). A setting has the property `comm_schedule` iff the agents have a way of communication for deciding who is performing actions at which time point. This e.g. would allow to let more experienced or specialized (but maybe also more costly) agents perform actions in certain types of situations. In a setting with the property `talk` the agents have the ability to communicate about their knowledge.

We will now describe a number of settings. Table 1 lists these settings together with their properties. In the column "other's ability" we list the ability of the other

agents in the world. 'teacher' means an agent having a good (but perhaps not optimal) policy. 'perfect' means an agent with an optimal policy. In the text we will refer to specific entries in the Table with numbers between round brackets.

The first two settings, are the standard settings, (0) for the single agent case and (1) for the situation in which n single (R)RL-agents are put together in the same environment.

Probably one of the simplest settings is the case where `comm_reward`, `know_aim` and `full_obs` hold. Still, even this simple situation is not fully studied in the relational case. One empirical work is on "guidance" [4]. One can see guidance as a setting with two agents where one is a teacher, and the other is a learner (2). Both the teacher and the learner perform actions, and hence it is more likely that reward is obtained than in the classical reinforcement learning case. This can be important in difficult planning domains where it would be unlikely to obtain a reward by only exploration. The main advantage is that the teacher directs the exploration towards difficult to reach parts of the state space.

If we also have `know_abil` and the teacher is known to make only optimal actions (3), the learner can directly use the actions performed by the teacher as examples of optimal actions. He could then use a direct learning algorithm that learns actions from states. Another interesting situation (4,5) is the case with `comm_schedule` where the learner may ask the teacher to perform an action (at a certain cost). This is described in [4], while several open questions remain. This can also be seen as a form of active learning.

In the presence of a perfect teacher, the `talk` property together with `know_abil` (6) makes the problem somewhat trivial as the learner can just ask the teacher for the optimal policy (at least if the learner is able to represent that policy). However, the situation gets much more interesting when we have only `comm_reward`, `know_aim`, `full_obs`, `talk` and maybe `know_abil` but no perfect teacher (7). We then have a situation where agents can discuss their learning experiences and even though there is full knowledge, the problem is far from trivial. Indeed, in the relational setting, agents can use very expressive languages to talk. Furthermore, extending the idea of Informed Reinforcement Learning [2], the agents can exchange their learned information. Possible interesting information to share could be subgoals, information about actions like pre- or postconditions but also learned options or macro-actions. No work has been published on which languages are suitable and which questions are most efficient. One could use a language which is a super language of the language to describe states and actions, and can also describe past episodes and statistics. E.g. one could imagine that one agent asks "did you ever see a state with four stacks containing each a red block above a light blue block?" And another agent might answer: "No, but I did see something very similar: I visited 13 states with four stacks containing each a red block above a dark blue one. Is that of interest for you?". Apart from the usual communication issues, one could investigate issues such as the following. What questions are useful in a communication? How to get the desired information at the lowest cost? What generalizations are needed for that? When is it cheaper to explore and find it out oneself?

An other unsolved task (8) occurs in the situation where an agent sees several other agents and knows that some of them are quite good agents. In such a situation it may be interesting to try and find the best agent to learn from. But as the reward is collective, it may not be trivial to detect who is performing the best actions. Or maybe, some agents are experts in certain areas of the state space, and it might be interesting to learn concepts describing the other agent's expertise (the areas where they perform well).

In what precedes, we have listed a number of unsolved tasks which may be assumed to be 'easy' for the learner. Of course, one can make the problems much more difficult by adding a number of supplementary complexity factors, such as partial observability (9) or situations where not all agents try to reach the same goal (10).

Setting	other's ability	comm reward	know abil	know aim	full obs	comm schedule	talk
0. Std. RRL (1 agent)	no			x	x		
1. Std. MAS - RRL (n agents)	any	x					
2. Guidance	teacher	x		x	x		
3. Guided policy learning	perfect	x	y	x	x		
4. Active guidance	teacher	x		x	x	x	
5. Actively guided policy learning	perfect	x	y	x	x	x	
6. Describing the solution	perfect	x	y	x	x	x	x
7. Collaborative RRL	any	x		x	x		x
8. Find the teacher	teacher+any	x		x	x		
9. Partially observable world	any	x		x	x		
10. Different interests	any			x			

Table 1. A number of settings with their properties

5 Preliminary Experiments

In this section, we will present and discuss some results from preliminary experiments. In the presented experiments, we tested three different goals the agent(s) need to achieve: the $on(A, B)$-goal, the *stack*-goal and the *unstack*-goal. In the $on(A, B)$-goal the agent(s) only receive a reward iff block A is directly on top of block B. The objective of the *stack*-goal is to put all blocks in one and the same stack, i.e. there is only one block on the floor. And in the *unstack*-goal the agents are rewarded iff all blocks are on the floor, i.e. there is no block on top of another block.

We used a blocks world with 6 blocks where the testing episodes were ended when a reward is received or when the maximum number of actions is reached. This maximum is the number of actions in the optimal policy (shortest path to the goal), increased with two. Only for the $on(A, B)$-goal (the most difficult goal [3]), a slightly easier setting is used with 5 blocks and a maximum of three extra steps over the optimal path is allowed. The figures show the average over a 10-fold run where each test run consists of 100 episodes and the average reward over this 100 episodes, i.e. the percentage of episodes in which a reward is received, is used as a convergence measure.

Preliminary experiments were performed in the first two settings of Table 1. To analyze the problem of the interference (the agents are trying to move the same blocks) and the incomplete information (since actions are taken simultaneously, there is no full information on the exact state) in the standard setting, we performed the same experiments in an extra setting where one agent tells the other which action he is going to take. This can be seen as a very simple form of communication. According to [19], this corresponds to the setting with one level-0 agent and one level-1 agent.

To overcome the problem of spareness of rewards (since only 1 single agent receives a reward when getting in a goal state), one could reward all agents simultaneously when arriving in a goal state. Another possibility would be to create or define an extra reward or utility function over all agents. The third way, which is somewhat in between this two possibilities, is to give the agents a way to communicate about the rewards they receive. Hence, an agent can act as a sort of teacher to all the other agents when he performed an action for which he received a reward.

We should also remark that the experiments here are performed in a slightly different setting than usual [3]: In most previous work on Relational Reinforcement Learning, it is assumed that it is possible for the learning agent to ask for the valid actions in the current state, but because the current state is unknown to the second agent in our setting, a set of all actions is returned, which makes the learning problem obviously more complex.

The results of these experiments, using the RRL-TG algorithm [5] as Q-regression, for the $on(A, B)$-goal can be found in Figure 1. Figure 2 shows the results for the *stack*-goal and Figure 3 for the *unstack*-goal.

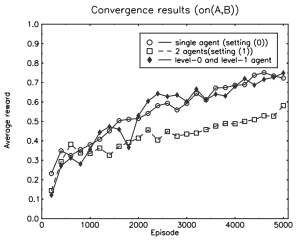

Fig. 1. On(A,B) goal

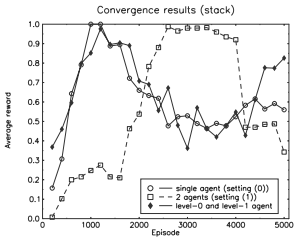

Fig. 2. Stack goal

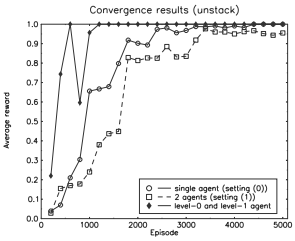

Fig. 3. Unstack goal

Figure 1 clearly shows that a single agent performs more or less the same as the combination of the level-0 and level-1 agent and they both outperform the two level-0 agents. There is no performance gain when moving to multiple agents, this means that no profit is taken for taking two actions instead of one at a single time step. The reason for this is the difficulty of the $on(A, B)$-goal, the optimal policy would be to first move all blocks away above block A and block B and then move A to B. But if the level-1 agent does not learn to work on the other stack, there will be a lot of interference between the two agents.

In case of the *stack*-goal (Figure 2) the results are harder to interpret, because of the variable performance in the single-agent case. The cause for this variance in performance is due to the working of the TG-algorithm. But, is is reasonable to say that in the setting in which one agent tells his action to the other agent, it tends to do a bit better when the learning time increases. The curve for setting (1) has more or less the same shape, but it takes longer to learn.

The best results are obtained for the *unstack*-goal (Figure 3), which is to be expected because it is the goal which is the most easy to distribute over the different agents. Both agents can move blocks to the floor, agent B only needs to learn that he cannot move the same block to the floor as agent A, but the block underneath that one if he moves a block from the same stack. The small difference with the performance of setting (1) also shows that the amount of interference is a lot less for the *unstack*-goal.

From these preliminary experiments, it can be concluded that it takes longer to learn in setting (1) than in setting (0) and that there can be a performance gain if the agents communicate over their actions and the task is not too hard to distribute.

6 Conclusions

In this paper we introduced the novel idea of cross-fertilizing relational reinforcement learning with multi-agent systems for solving distributed dynamic planning tasks as the blocks world example. More precisely, we propose to use a relational representation of the state space in multi-agent reinforcement learning as this has many proved benefits over the propositional one, as for instance handling large state spaces, and as such better reflects real world applications.

We started this paper with a short introduction to relational reinforcement learning and multi-agent learning. Then we proceeded to describe our view on multi-agent relational reinforcement learning. Although this is still work under development, we believe that these ideas can greatly enhance the application of multi-agent distributed planning. We defined different settings in which we believe this research should be carefully conducted, according to six different properties. The different settings are summarized according to their level of complexity in Table 1. The paper is concluded with a discussion of some preliminary experiments.

Acknowledgements

Tom Croonenborghs is supported by the Flemish Institute for the Promotion of Science and Technological Research in Industry (IWT). Jan Ramon is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders.

References

1. C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 15th International Conference on Artificial Intelligence, p.746-752*, 1998.

2. T. Croonenborghs, J. Ramon, and M. Bruynooghe. Towards informed reinforcement learning. In P. Tadepalli, R. Givan, and K. Driessens, editors, *Proceedings of the ICML2004 workshop on relational reinforcement learning*, pages 21–26, Banff, Canada, July 2004.
3. K. Driessens. *Relational Reinforcement Learning*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, 2004.
4. K. Driessens and S. Dzeroski. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57(3):271–304, 2004.
5. K. Driessens, J. Ramon, and H. Blockeel. Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In L. De Raedt and P. Flach, editors, *Proceedings of the 13th European Conference on Machine Learning*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 97–108. Springer-Verlag, 2001.
6. S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.
7. T. Grenager, R. Powers, and Y. Shoham. Dispersion games:general definitions and some specific learning results. In *Eighteenth national conference on Artificial intelligence, Edmonton, Alberta, Canada, Pages: 398 - 403*, 2002.
8. P. Hoen and K. Tuyls. Engineering multi-agent reinforcement learning using evolutionary dynamics. In *proceedings of the 15th European Conference on Machine Learning*, 2004.
9. J. Hu and M. P. Wellman. Experimental results on q-learning for general-sum stochastic games. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 407–414. Morgan Kaufmann Publishers Inc., 2000.
10. L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.
11. M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning, p 157 - 163*, 1994.
12. A. Nowé, J. Parent, and K. Verbeeck. Social agents playing a periodical policy. In *Proceedings of the 12th European Conference on Machine Learning, p 382 - 393, Freiburg*, 2001.
13. S. Sen, S. Airiau, and R. Mukherjee. Towards a Pareto-optimal solution in general-sum games. In *in the Proceedings of the Second Intenational Joint Conference on Autonomous Agents and Multiagent Systems, (pages 153-160), Melbourne, Australia, July 2003*, 2003.
14. P. Stone. Layered learning in multi-agent systems. *Cambridge, MA: MIT Press*, 2000.
15. R. Sutton and A. Barto. *Reinforcement Learning: an introduction*. The MIT Press, Cambridge, MA, 1998.
16. P. Tadepalli, R. Givan, and K. Driessens. Relational reinfocement learning: An overview. In *Proceedings of the ICML'04 Workshop on Relational Reinfocement Learning*, 2004.
17. K. Tumer and D. Wolpert. COllective INtelligence and Braess' Paradox. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence, pages 104-109.*, 2000.
18. K. Tuyls, K. Verbeeck, and T. Lenaerts. A selection-mutation model for Q-learning in Multi-Agent Systems. In *The second International Joint Conference on Autonomous Agents and Multi-Agent Systems. The ACM International Conference Proceedings Series, Melbourne, Australia*, 2003.
19. J. M. Vidal and E. H. Durfee. Agents learning about agents: A framework and analysis. In *Multiagent Learning Workshop*, 1997.