A Framework for Learning Agents and its Application to Market-Based MAS

José M. Vidal
Artificial Intelligence Laboratory, University of Michigan
1101 Beal Avenue, Ann Arbor, MI 48109-2110
jmvidal@umich.edu

Abstract

This document describes work in progress that aims towards the implementation of a framework for describing and understanding the behavior of systems with learning agents (that learn models of other agents), and the application and use of this framework within the market-based MAS that is the University of Michigan Digital Library.

Introduction

This document describes work in progress that aims towards the implementation of a framework for describing and understanding the behavior of systems with learning agents (that learn models of other agents), and the application and use of this framework within the market-based MAS that is the University of Michigan Digital Library (UMDL). Since the research has not been completed we have, as of now, no experimental results to report.

Our framework consists of a set of difference equations used to describe the evolution of an agent's error over time as it learns (e.g. builds models of other agents) in a MAS that is, perhaps, composed of other similar learning agents. Our application domain implements these learning agents and gives them protocols and some rudimentary meta-level decision capabilities that allows them to decide when to learn deeper models of the other agents and when to take actions that eliminate their need to keep deeper models (and, thereby, lower their computational costs without decreasing the expected value they get from participating in the market).

The World

We define a MAS as consisting of a finite number of agents, actions, and world states. We let N denote the set of agents in the system, W denote the set of of world states (i.e. distinct states of the world that the agents can perceive), and A_i, where $|A_i| \geq 2$, denotes the set of actions agent $i \in N$ can take. We assume discrete time, indexed in the various functions by the superscript t, where t is an integer greater than 0.

Each agent $i \in N$ has a **decision function**, given by $\delta_i^t : W \rightarrow A_i$. This function maps each state $w \in W$ to the action $a_i \in A_i$ that i will take in that state. The action that agent i *should* take, assuming knowledge of all other agents' decision functions and i's payoffs, is given by the **target function** $\Delta_i^t : W \rightarrow A_i$. The agent's learning task is to change its decision function so that it matches the target function. The target function is assumed to be "myopic"; that is, it does not take into account the possibility of future encounters, but instead simply finds the action that maximizes the immediate payoff given the current situation.

Modeling a Learning Algorithm

The agents in the MAS are engaged in a discrete action/learn loop that works as follows: At time t the agents perceive a world $w^t \in W$ which is drawn from a fixed distribution $\mathcal{D}(w)$. They then each take the action dictated by their δ_i^t functions, and all of these actions are assumed to be taken in parallel. Lastly, they receive some sort of lesson or reward which their learning algorithm uses to change the δ_i^t so as to (hopefully) better match Δ_i^t . By time $t+1$, the agents have new δ_i^{t+1} functions and are ready to perceive the world again and repeat the loop. Notice that, at time t the Δ_i^t takes into account the δ_j^t of all other agents $j \in N_{-i}$.

The agent's learning algorithm is responsible for changing δ_i^t into δ_i^{t+1} so that it better matches Δ_i^t. Different machine learning algorithms will achieve this with different degrees of success. We have found a set of parameters that can be used to model the effects of a wide range of learning algorithms. We will also assume that each agent uses only one learning algorithm during its lifetime. This means that when we talk about an agent's learning capabilities we are really talking about the capabilities of the agent's learning algorithm.

After agent i takes an action and receives some reward/lesson, it will try to change its δ_i^t to match Δ_i^t. We can expect that for some w's it was true that $\delta_i^t(w) = \Delta_i^t(w)$, while for some other w's this was not the case. That is, some of the $w \rightarrow a_i$ mappings given by $\delta_i^t(w)$ might have been incorrect. In general, a learning algorithm might affect both the correct and

incorrect mappings. We will treat these two cases separately.

We start by considering the incorrect mappings and define the **change rate** of the agent as the probability that the agent will change one of its incorrect mappings. Formally, we define the change rate c_i for agent i as

$$\forall_w \; c_i = \mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)] \quad (1)$$

This tells us how likely the agent is to change an incorrect mapping into something else. This "something else" might be the correct action. The probability that the agent changes an incorrect mapping to the correct action is called the **learning rate** of the agent, which is defined as l_i where

$$\forall_w \; l_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)] \quad (2)$$

The value of l_i must take into account the fact that the worlds seen at each time step are taken from $\mathcal{D}(w)$. There are two constraints that must always be satisfied by these two rates. Since changing to the correct mapping implies that a change was made, the value of l_i must be less than or equal to c_i, i.e. $l_i \leq c_i$ must always be true. Also, if $|A_i| = 2$ then $c_i = l_i$ since there are only two actions available, so the one that is not wrong must be right. The complimentary value of $1 - l_i$ gives us the probability that an incorrect mapping does not get fixed. As an example, $l_i = .5$ means that, if agent i initially has all mappings wrong, it will get half of them right after the first iteration.

We now consider the agent's correct mappings and define the **retention rate** as the probability that a correct mapping will stay correct in the next iteration. The retention rate is given by r_i where

$$\forall_w \; r_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) = \Delta_i^t(w)] \quad (3)$$

We propose that the behavior of a wide variety of learning algorithms can be captured (or at least approximated) using appropriate values for c_i, l_i and r_i.

Finally, we define **volatility** to mean the probability that the oracle function will change. Formally, volatility is given by v_i where

$$\forall_w \; v_i = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \quad (4)$$

We can determine the value of v_i in terms of the other agents' changes in their decision functions. That is, in terms of $\mathbf{Pr}[\delta_j^{t+1} \neq \delta_j^t]$, for all other agents j.

In order to do this we first need to define the **impact** I_{ji} that agent j's changes in its decision function have on i.

$$\forall_{w \in W} \; I_{ji} = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) \mid \delta_j^{t+1}(w) \neq \delta_j^t(w)] \quad (5)$$

We can now start by determining that, for two agents i and j

$$\begin{aligned}\forall_{w \in W} \; v_i^t &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \\ &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_j^{t+1}(w) \neq \delta_j^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)] \\ &\quad + \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_j^{t+1}(w) = \delta_j^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_j^{t+1}(w) = \delta_j^t(w)]\end{aligned} \quad (6)$$

The first thing to notice is that volatility is no longer constant, it varies with time (as recorded by the superscript). The first conditional probability in (6) is just I_{ji}, the second one we will set to 0 since we are specifically interested in MASs where the volatility arises *only* as a side-effect of the other agents' learning. That is, we assume that agent i's target function changes only when j's decision function changes. We ignore any other outside influences that might make the agent's target function change.

We can then simplify this equation and generalize it to N agents, under the assumption that the other agents' changes in their decision functions will not cancel each other out, making δ_i^t stay the same as a consequence. This is an approximation that makes the equations simpler. v_i^t then becomes

$$\begin{aligned}\forall_{w \in W} \; v_i^t &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \\ &= 1 - \prod_{j \in N_{-i}} (1 - I_{ji} \mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)])\end{aligned} \quad (7)$$

We now need to determine the expected value of $\mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)]$ for any agent. Using i instead of j we have

$$\begin{aligned}\forall_{w \in W} \; &\mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w)] \\ &= \mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_i^t(w) \neq \Delta_i^t(w)] \\ &\quad + \mathbf{Pr}[\delta_i^t(w) = \Delta_i^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_i^t(w) = \Delta_i^t(w)]\end{aligned} \quad (8)$$

where the expected value is:

$$E[\mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w)]] = c_i e(\delta_i^t) + (1 - r_i) \cdot (1 - e(\delta_i^t)) \quad (9)$$

We can then plug (9) into (7) in order to get the expected volatility

$$E[v_i^t] = 1 - \prod_{j \in N_{-i}} 1 - I_{ji}(c_j e(\delta_j^t) + (1 - r_j) \cdot (1 - e(\delta_j^t))) \quad (10)$$

The c_i, l_i, r_i and I_{ij} constitute what we call the CLRI model of an agent's learning abilities. These parameters allow us to capture the aspects of an agent's

incorrect mappings. We will treat these two cases separately.

We start by considering the incorrect mappings and define the **change rate** of the agent as the probability that the agent will change one of its incorrect mappings. Formally, we define the change rate c_i for agent i as

$$\forall_w \, c_i = \mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)] \quad (1)$$

This tells us how likely the agent is to change an incorrect mapping into something else. This "something else" might be the correct action. The probability that the agent changes an incorrect mapping to the correct action is called the **learning rate** of the agent, which is defined as l_i where

$$\forall_w \, l_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)] \quad (2)$$

The value of l_i must take into account the fact that the worlds seen at each time step are taken from $\mathcal{D}(w)$. There are two constraints that must always be satisfied by these two rates. Since changing to the correct mapping implies that a change was made, the value of l_i must be less than or equal to c_i, i.e. $l_i \leq c_i$ must always be true. Also, if $|A_i| = 2$ then $c_i = l_i$ since there are only two actions available, so the one that is not wrong must be right. The complimentary value of $1 - l_i$ gives us the probability that an incorrect mapping does not get fixed. As an example, $l_i = .5$ means that, if agent i initially has all mappings wrong, it will get half of them right after the first iteration.

We now consider the agent's correct mappings and define the **retention rate** as the probability that a correct mapping will stay correct in the next iteration. The retention rate is given by r_i where

$$\forall_w \, r_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) = \Delta_i^t(w)] \quad (3)$$

We propose that the behavior of a wide variety of learning algorithms can be captured (or at least approximated) using appropriate values for c_i, l_i and r_i.

Finally, we define **volatility** to mean the probability that the oracle function will change. Formally, volatility is given by v_i where

$$\forall_w \, v_i = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \quad (4)$$

We can determine the value of v_i in terms of the other agents' changes in their decision functions. That is, in terms of $\mathbf{Pr}[\delta_j^{t+1} \neq \delta_j^t]$, for all other agents j.

In order to do this we first need to define the **impact** I_{ji} that agent j's changes in its decision function have on i.

$$\forall_{w \in W} \, I_{ji} = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) \mid \delta_j^{t+1}(w) \neq \delta_j^t(w)] \quad (5)$$

We can now start by determining that, for two agents i and j

$$\begin{aligned}\forall_{w \in W} \, v_i^t &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \\ &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_j^{t+1}(w) \neq \delta_j^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)] \\ &\quad + \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_j^{t+1}(w) = \delta_j^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_j^{t+1}(w) = \delta_j^t(w)]\end{aligned} \quad (6)$$

The first thing to notice is that volatility is no longer constant, it varies with time (as recorded by the superscript). The first conditional probability in (6) is just I_{ji}, the second one we will set to 0 since we are specifically interested in MASs where the volatility arises *only* as a side-effect of the other agents' learning. That is, we assume that agent i's target function changes only when j's decision function changes. We ignore any other outside influences that might make the agent's target function change.

We can then simplify this equation and generalize it to N agents, under the assumption that the other agents' changes in their decision functions will not cancel each other out, making δ_i^t stay the same as a consequence. This is an approximation that makes the equations simpler. v_i^t then becomes

$$\begin{aligned}\forall_{w \in W} \, v_i^t &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \\ &= 1 - \prod_{j \in N_{-i}} (1 - I_{ji} \mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)])\end{aligned} \quad (7)$$

We now need to determine the expected value of $\mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)]$ for any agent. Using i instead of j we have

$$\begin{aligned}\forall_{w \in W} \, &\mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w)] \\ &= \mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_i^t(w) \neq \Delta_i^t(w)] \\ &\quad + \mathbf{Pr}[\delta_i^t(w) = \Delta_i^t(w)] \\ &\quad \cdot \mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^t(w) | \delta_i^t(w) = \Delta_i^t(w)]\end{aligned} \quad (8)$$

where the expected value is:

$$E[\mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w)]] = c_i e(\delta_i^t) + (1 - r_i) \cdot (1 - e(\delta_i^t)) \quad (9)$$

We can then plug (9) into (7) in order to get the expected volatility

$$E[v_i^t] = 1 - \prod_{j \in N_{-i}} 1 - I_{ji}(c_j e(\delta_j^t) + (1 - r_j) \cdot (1 - e(\delta_j^t))) \quad (10)$$

The c_i, l_i, r_i and I_{ij} constitute what we call the CLRI model of an agent's learning abilities. These parameters allow us to capture the aspects of an agent's

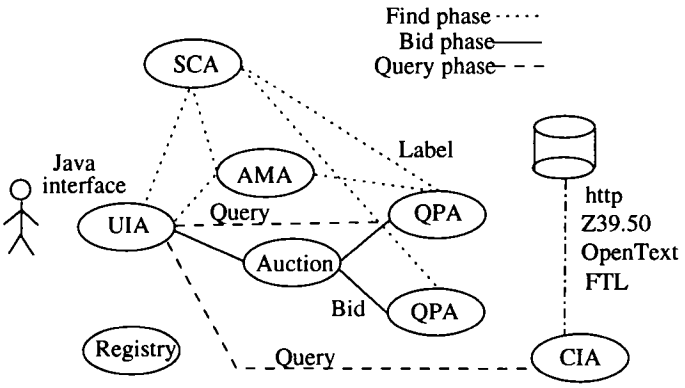

Figure 1: The UMDL market economy.

learning dynamics that are important in determining how the agent will behave, and how it will impact other agents, in a multi-agent system composed of similar agents.

Calculating the Agent's Error

In order to more formally talk about an agent's behavior, we define the error of agent i's decision function δ_i^t at time t as:

$$e(\delta_i^t) = \sum_{w \in W} \mathcal{D}(w)\mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^t(w)] \qquad (11)$$

$e(\delta_i^t)$ gives us the probability that i will take a wrong action given that the worlds are taken from a fixed probability density function $\mathcal{D}(w)$.

The agent's expected error at time $t+1$ can be calculated by considering the four possibilities of whether $\Delta_i^t(w)$ changes or not, and whether $\delta_i^t(w)$ was correct or not. This approach leads to the derivation of a difference equation that tells us how the agent's error is expected to progress with the passage of time. This difference equation, along with its derivation, simplifications, and experimental applications, can be found in (Vidal & Durfee).

The Application

In the UMDL we have given the agents the ability to learn such that the UMDL becomes the type of system detailed in Section . We have User Interface Agents (UIA) that act as buyers, Query Planning Agents (QPAs) that act as sellers, and Auction agents. A picture of the system can be seen in Figure 1.

Each QPA has a specialty service it provides (e.g. QPA1 could sell `<recommend, science, professional, fast, precise>`), but it can sell at auctions that sell a service that subsumes the one that the QPA sells (e.g. QPA1 could participate in an auction that sells `<recommend,*,*,*,*>`, where * is a wildcard). The Auction Manager Agent (AMA) tells the QPA all the auctions that it can participate in. The AMA does *not* create a new auction specifically for each QPA, instead it tells the QPAs to go to an auction(s) that subsume it service description.

The UIAs, on the other hand, do not know which specific services they prefer. This reflects the fact that the users will not know what type of service they prefer until they have had some experience with the different kinds of services. UIAs ask the AMA for a general auction to buy from and start buying from this auction. While they are buying they also remember the the value of each good received. Value is a function of both price and quality. UIAs will generally prefer goods of high quality and low price, the combination of these two implies a high value.

If the UIAs notice a variance in the value that they are receive they then start to remember the specific quality delivered by each QPA. After a while they decide that they would prefer an auction where only some of the QPAs participate. If this subset of QPAs can be captured by a more specific service label that excludes all the other QPAs (e.g. `<recommend,science,*,fast,*>`, then a new auction is created for this specific service and all the appropriate agents are notified.

Summary of Current Results

Preliminary experimental results show that the system does behave as expected. New auctions are created at the request of buyers and the buyer's need to keep specific models of the sellers disappears. However, this need disappears only temporarily because the buyer must always be wary of new, even more specific agents appearing. The buyer must also keep track of the price because if the price rises too much then it might be better for the buyer to go back to a previous auction. In fact, in the extreme case where the new auction is populated by only one seller we notice that the buyer quickly comes back to the previous auction. This is due to the fact that in this case the seller has a monopoly on that auction and, therefore, raises prices as high as he can.

In summary, we find that while our system does take some steps in facilitating the use of satisficing models, it still leaves a lot of the burden on the agent. Specifically, the agent must be able to determine when the models it keeps are too burdensome. After identifying this fact, however, the agent can set our protocol in motion and (if everything goes as expected) eliminate his need for the more detailed models. Also, the agent must always continue to monitor his value/profit for the kind of volatility that would indicate that more detailed models should be used.

We are also applying our CLRI model to this problem in an effort to predict how the individual agents perform over time. There are, however, some obstacles that we have had to surpass. Firstly, we noticed that the creation of new auctions does, in fact, correspond to a simplification of the agent's learning prob-

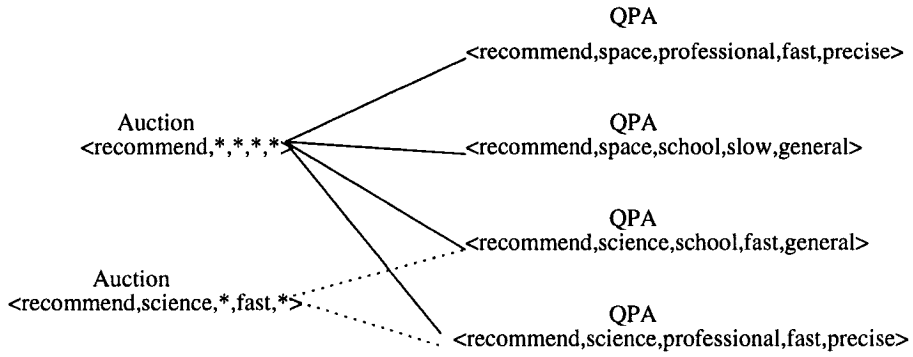

Figure 2: Creation of a more specific auction.

lem. That is, by creating a new auction the agent no longer needs to keep models of the seller's and their prices so its learning task is simplified. We capture this effect by increasing the learning rate l_i and retention rate r_i of the agent i.

In fact, we found it best to consider both cases separately. That is, the world before the auction is created and the world after the auction is created are two different scenarios in our CLRI model. They each have (possibly) different values for the parameters. Given these parameters we can then apply our difference equation to find the agent's error before and after the new auction is created.

Another obstacle we found was the definition of $e(\delta_i^t)$ within this system. The problem lies specifically in the definition of the target function. There does not seem to be one clear choice for a target function. What the agent *should* do seems to be open to a few equally valid propositions. For example, a buyer could bid on of the prices (there might be a tie) that give it the highest expected value, or it could bid a price that assumes it will get the service from one particular seller it favors (which would get it the highest total value, if that seller is picked), or it could pick some risk-averse strategy and try to lose as little money as possible, etc. However, once a particular target function definition is chosen, our CLRI model can predict the agents' error within this definition.

Our ongoing work involves formalizing the mapping between the CLRI model of the agents and the real-world situation as given by our UMDL economic system. This will allow us to make quantitative predictions on the specificity of the models that an agent should build (i.e. how detailed should they be, and under which circumstances).

References

Durfee, E. H.; Kiskis, D. L.; and Birmingham, W. P. 1997. The agent architecture of the University of Michigan Digital Library. *IEE Proceedings on Software Engineering* 144(1):61–71.

Hu, J., and Wellman, M. P. 1996. Self-fulfilling bias in multiagent learning. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 118–125.

Mullen, T., and Wellman, M. P. 1996. Market-based negotiation for digital library services. In *Second USENIX Workshop on Electronic Commerce.*

Vidal, J. M., and Durfee, E. H. The moving target function problem in multi-agent learning. Submitted to ICMAS 98.

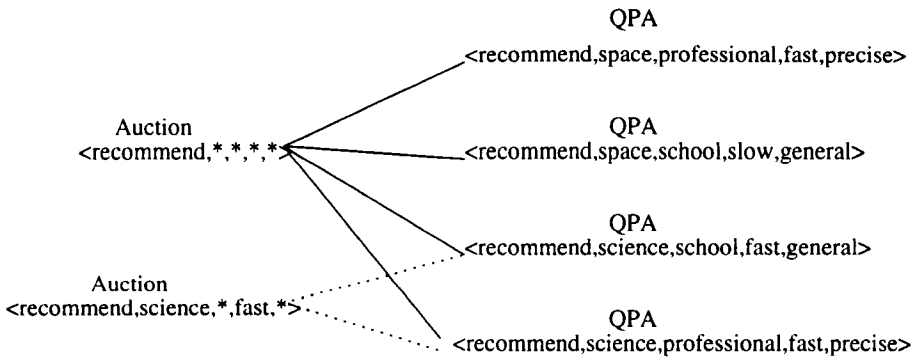

Figure 2: Creation of a more specific auction.

lem. That is, by creating a new auction the agent no longer needs to keep models of the seller's and their prices so its learning task is simplified. We capture this effect by increasing the learning rate l_i and retention rate r_i of the agent i.

In fact, we found it best to consider both cases separately. That is, the world before the auction is created and the world after the auction is created are two different scenarios in our CLRI model. They each have (possibly) different values for the parameters. Given these parameters we can then apply our difference equation to find the agent's error before and after the new auction is created.

Another obstacle we found was the definition of $e(\delta_i^t)$ within this system. The problem lies specifically in the definition of the target function. There does not seem to be one clear choice for a target function. What the agent *should* do seems to be open to a few equally valid propositions. For example, a buyer could bid on of the prices (there might be a tie) that give it the highest expected value, or it could bid a price that assumes it will get the service from one particular seller it favors (which would get it the highest total value, if that seller is picked), or it could pick some risk-averse strategy and try to lose as little money as possible, etc. However, once a particular target function definition is chosen, our CLRI model can predict the agents' error within this definition.

Our ongoing work involves formalizing the mapping between the CLRI model of the agents and the real-world situation as given by our UMDL economic system. This will allow us to make quantitative predictions on the specificity of the models that an agent should build (i.e. how detailed should they be, and under which circumstances).

References

Durfee, E. H.; Kiskis, D. L.; and Birmingham, W. P. 1997. The agent architecture of the University of Michigan Digital Library. *IEE Proceedings on Software Engineering* 144(1):61–71.

Hu, J., and Wellman, M. P. 1996. Self-fulfilling bias in multiagent learning. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 118–125.

Mullen, T., and Wellman, M. P. 1996. Market-based negotiation for digital library services. In *Second USENIX Workshop on Electronic Commerce.*

Vidal, J. M., and Durfee, E. H. The moving target function problem in multi-agent learning. Submitted to ICMAS 98.