Decentralized Language Learning Through Acting

Claudia V. Goldman   Martin Allen   Shlomo Zilberstein
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
{clag,mwallen,shlomo}@cs.umass.edu

Abstract

This paper presents an algorithm for learning the meaning of messages communicated between agents that interact while acting optimally towards a cooperative goal. Our reinforcement-learning method is based on Bayesian filtering and has been adapted for a decentralized control process. Empirical results shed light on the complexity of the learning problem, and on factors affecting the speed of convergence. Designing intelligent agents able to adapt their mutual interpretation of messages exchanged, in order to improve overall task-oriented performance, introduces an essential cognitive capability that can upgrade the current state of the art in multi-agent and human-machine systems to the next level. Learning to communicate while acting will add to the robustness and flexibility of these systems and hence to a more efficient and productive performance.

1. Introduction

The ability to communicate is essential if intelligent agents are to interact efficiently with other agents or human beings. In turn, communicative agents must be able both to exchange information, and to understand what others communicate. We study how agents can combine interpretation with action, learning to understand messages to improve performance towards a cooperative goal.

Our focus is on cooperative systems, composed either of autonomous agents alone or of a combination of autonomous agents and humans, working to maximize joint global utility. (Our model would also apply in some cases to competitive agents, since even self-interested agents may find it individually beneficial to learn to understand the communications of others.) We assume that the agents are involved in a decentralized control process in which none of them can observe their environment fully [1]. Due to the distributed character of the system, there is no central entity that can oversee the behaviors of all the agents and instruct each how to behave optimally. Even agents who act cooperatively may not necessarily share the same language of communication, and may not simply be able to exchange a translation, mapping discrepancies in the languages, in an off-line manner. Such problems may occur, for instance, when the content of an agent's communication arises from observations available solely to that agent, in the midst of some shared task. Even agents with the same sensing apparatus may still lack the contextual information necessary to correctly interpret each other's messages. Alternatively, the ability to learn new meanings can guard against unintentional design-time errors. Even agents with pre-existing protocols for sharing information may come to recognize that their joint performance is not as expected, and that they need to adjust their interpretation of shared messages in order to rectify the situation. Verification systems [17] aim at identifying inconsistencies between a software specification and its execution code, which can be eventually fixed manually. Our purpose is to automatically learn to correct a misinterpretation in addition to identifying it (on-line or by simulation), in the framework of decentralized control.

In many applications, the misinterpretation of messages can lead to mis-coordination and an eventual decrease in performance. NASA's Climate Orbiter probe, for instance, crashed as a result of an unwitting use of different (metric and imperial) conventions of measure by different design teams, causing the spaceship to follow an incorrect flight plan [13]. In future, automated systems, consisting of interacting agents designed by multiple agencies or nations, will need the ability to re-adjust how various communicated directives are understood, to prevent such misunderstandings if necessary. Such considerations also arise where users of a system may have different levels of competence—as in automated and interactive tutoring—or where it is practically infeasible to specify all necessary communication protocols at design-time. The latter problem arises, for instance, in automated control and diagnosis. In such contexts, the range of ways a particular mechanism may go wrong cannot generally be known in advance, and encountered problems often require novel diagnoses and solutions [2]. Such problems are compounded when various mechanisms are combined as parts of a larger overall process, as is common in manufacturing plants. Without the ability to understand novel communications, automated control systems can be hindered in their work toward the cooperative goal of stabilizing plant performance, since it becomes difficult, if not impossible, to coordinate courses of action in the event of some new combination of mechanical failures.

Each of these problems is an expanded version of general ones to do with coordinated action in the pursuit of common goals. In the study of decentralized Markov Decision Processes for multi-agent systems [1, 15], research has concentrated on cases in which agents share a means of commonly-understood communication (if communication plays any role at all). The previous examples suggest that solving such problems will require not only the ability to learn better policies for joint action, but also the capacity to learn how to interpret exchanged messages correctly. Such problems thus combine the general task of coordinating action towards optimal outcomes with the specific goal of coming to understand novel communications so to better facilitate those outcomes.

This paper frames the language-learning problem as a decentralized control problem (Section 3). After presenting related work (Section 2), we adopt Bayesian filtering methods to the language-learning problem in cooperative contexts. Initial experiments (Section 4) show how languages with different structures can be learned and how the learning process is affected by characteristics of both language and environment. Section 5 discusses our conclusions and points to extensions of the current research.

2. Related Work

Prior studies on coordination in multi-agent systems have generally assumed that agents possess a previously-known and fixed language of communication when appropriate. (See for example [12, 5]); consider also the KQML language [6], a standard that pre-sets all possible inter-agent communication). We believe that robust decentralized systems require that agents adapt their communication language when new situations arise or when mis-coordination occurs possibly due to misunderstandings. Such mis-coordination can either be revealed in practice, or in simulation, and serves as a signal for reinterpretation of messages received.

At the intersection between cognitive science and computational simulations, research has studied the evolution of lexicons and the problem of learning a language from positive examples [20, 7]. Our approach is different: we study how agents can learn to interpret each other's messages to improve performance in the context of a shared task and plan of action. Adaptive language games [18] and the anchoring problem [3] are also relevant areas of study. The latter work raises questions concerning formal models for the study of the language-learning problem. This paper is a step in that direction: we formalize the language-learning problem as a decentralized control problem [9] and study it in the framework of reinforcement learning. Our previous research has focused on the computation of optimal joint policies when a shared language of communication is assumed. Here, the agents need to learn such a language in order to optimize their joint behavior. The system is reinforced so it can improve its global performance. Agents update their interpretations to act better. We do not reinforce the choices of interpretations per se (as in [22]).

Other work has proposed that rational and self-interested agents can negotiate to evolve a shared communication language [8]. In such a context, conflicts between these agents arise because each one prefers a distinct communication language, based on the cost of employing that language; the communication process here is defined in terms of communication acts that increase the speaker's expected utility. We are interested instead in communication that enables efficient coordination of agents towards a mutual goal. Communication serves to increase the overall utility of the system as a whole; the particular language learned will thus be directly related to this system-utility, rather than to the individual cost of using that language. As a result of these choices, agents using our techniques may be able to coordinate action efficiently or optimally even though not all the messages are completely understood. So long as their mutual goals are achieved satisfactorily, the agents will be similarly satisfied with their existing interpretations.

Importantly, our work relies on the idea that agents treat communicated messages as having some sort of *meaningful structure*. Initially, agents presume that others involved in a shared cooperative task are communicating information relevant to that task. This has the effect of reducing the number of possible interpretations an agent has to consider, making the learning process more manageable. As well, treating messages as having meaningful structure speeds learning and allows for generalizations between various environments. Our results show the advantages and complications of this approach. Treating messages as having some semantic structure can allow agents to learn their meanings more quickly; at the same time, the specification of this structure and the learning-updates related to it can become more difficult. The concentration on semantics further distinguishes our approach from such prior work as [21], in which a generalization of the perceptron algorithm was proposed to allow a multi-agent system to collectively learn a single shared concept.

Philosophically, this work stems from thought originating with Quine [16], who argues that interpreting speakers of foreign languages is essentially the same as interpreting speakers of our own. On this view, we are always constructing "translation manuals" between one another's utterances, and we understand others by relating what they say and do to what we ourselves would say and do in the context of our shared environment. Further, translation is always under-determined by available evidence, and there are often multiple competing possible translations of another's language. Davidson [4] and Putnam [14] extend these ideas. Davidson has written on what he calls "radical interpretation," exploring the idea that all understanding of others is a radically indeterminate procedure of making and adjusting predictions about their behavior, based on very basic assumptions about the structure of their beliefs and intentions. Again, this process is essentially the same whether or not we share a common language with the one being interpreted. We update our interpretation of other agents constantly, based upon our success or failure in predicting how they will behave, hoping at best to converge on some generally successful set of expectations regarding that behavior. Putnam examines how changing context can alter the meaning of even apparently well-understood terms in a language. Together, these ideas suggest that designers of communicative agents must allow that even well-defined protocols and languages can lead to cases in which the interpretation of messages become ambiguous or error-laden, and must be adjusted in order to make coordination possible.

3. The Language-learning Model

We frame the language-learning problem through action as a decentralized Markov decision process with direct communication [9]. Agents can exchange information while acting towards some global goal. Interpreting a message consists in deciding upon a course of action, after considering the contents of the messages received. In the particular context of our studies here, agents need to learn a mapping from the speaker's messages to their own observations, and then act based on an existing policy. For example, suppose that agents exchange messages describing some local goal to be pursued. We assume that each agent knows an optimal plan to reach any possible local goal. Then we will say that agent 1 correctly interprets a message σ sent by agent 2, if (*a*) agent 2 uses σ to express some local goal g_σ, and (*b*) agent 1 reaches goal g_σ (or one indifferent from the point of view of overall reward) following an optimal plan of action. The language-learning problem arises when agents act towards a goal as a result of misinterpreting the messages. Since we presume that the agents already possess optimal plans for any such goal, the problem is then learning how to interpret the messages correctly, so that they correctly identify the goal to be pursued.

We employ the model of a decentralized MDP with communication and goal-oriented behavior (GO Dec-MDP-Com) [11]. Formally, a GO Dec-MDP-Com for two agents and finite time-horizon T is given by $\langle S, A_1, A_2, \Sigma, C_\Sigma, P, R, \Omega_1, \Omega_2, O, T \rangle$. The set S is the set of global states, which are partially observed by each agent. Ω_i represents the set of these observations. O is the probability of making a certain observation (for each agent), given the state of the system and the actions performed by the agents (taken from corresponding set A_i). As a result of performing actions, the system transitions to a possibly different state s' with probability $P(s'|s, a_1, a_2)$. Agents exchange information, incurring a cost C_Σ. In this paper, agents do not share a single language of communication, i.e., there are two different languages Σ_1 and Σ_2 such that agent i initially employs Σ_i. We assume that actions incur a cost, and some states (denoted as global goal-states $g \in G \subset S$) yield a reward if reached by time T. That is, the reward function $R(s, a_1, a_2, s')$ is given by $Cost(a_1) + Cost(a_2) + JR(s')$, where $JR(s') \in \Re$ when $s' \in G$. Our model and associated algorithm generalize to the n-agents case, where learning occurs pairwise.

At time 0, each agent i can only send and correctly interpret messages from its own language Σ_i. The cost of communication C_Σ is set to zero. In the future, we will consider cases in which communication is not free, and the kinds of cooperation attainable when the agents cannot afford to communicate enough to allow for complete interpretations.

We apply the approximation scheme [10] based on a mechanism for communication. Such a mechanism decomposes a decentralized problem into temporary single-agent problems, and includes a policy of communication enabling agents to obtain global information when they communicate. In our study, agents share a common goal, which can then be broken down into local goals for each agent. Messages are used to share each agent's observations of relevant features of the environment. Given these observations, each agent can then deduce some local goal, and act towards it; the learning problem thus becomes one of learning how other agents express their observations.

In detail, the mechanism works as follows. Agents exchange messages in languages that are not completely understood by the receiver. Each agent then computes its local goal, and finds the optimal course of action leading to it. (Again, we assume that this solution is known.) The policy of communication instructs the agent to communicate each time they reach their local goal. The reward obtained when the agent reaches a local goal can be either a local reward function or the global reward that the system obtains. In the first case, we can say that agents learn each message correctly so long as they learn to maximize their individual rewards; the "correct" meaning of any message is then just the interpretation that most increases utility. In the second case, agents may still learn to coordinate their actions and maximize joint reward even though they may not have correctly interpreted all the messages exchanged from an individual perspective. For example, both agents may come to believe that they have behaved as expected because the overall reward value was the one expected for the system, even if this is in fact a result of one agent receiving more reward than it usually expects, while the other receives less.

As an example of this mechanism, if the agents' global goal is to meet as soon as possible in some gridworld environment, the mechanism used might be to exchange current locations and then move towards the location at the middle of the Manhattan distance between the agents. Each agent i computes this middle point based on the distance between its own location (known with certainty) and the location of the other agent j, given by the message that j has sent to i. Each agent adopts this point as its local and temporary goal, and applies its own optimal local policy of action to reach it. The agents then attempt to learn a correct mapping between their languages based on the rewards received after reaching the goal-point that each thought correct.

3.1. Bayes Filters

We adopt the method of Bayesian filtering, used for example in robotics to handle localization [19]; the **Bayes-Filter** algorithm for discrete state-spaces appears in Figure 1. In this model, agents possess a set of *beliefs*, in the form of a probability distribution over possible states of the environment; the filtering algorithm updates this distribution over time. Updates occur under two basic circumstances: (1) An agent updates its belief function based on new *observations*. Such observations may determine the state exactly, or may imply a probability distribution over the states depending perhaps on noise factors. (2) Agents make belief-updates *predictively*: before taking any action, the agent updates the probability distribution for each state in which it may end up. Implementation thus requires:

(1) A **sensor model**, giving $P(z|x)$, the probability of observation z in any state x.

(2) An **action model**, giving $P(x|u, x')$, the probability of state x, after action u in state x'.

3.1.1. The Particular Model In the language-learning context, states involve not only the immediate environment, but also a *translation-table*: a probability distribution over possible meanings for each message received. The agent's actions are then determined, in part at least, by the most probable meaning assigned to the most recent messages. The exact form of the table depends upon the presumed structure of the languages being translated. In general, each row in agent i's translation-table corresponds to some atomic component of the agent's own language Σ_i. Columns then correspond to atomic components identified in messages received in the language Σ_j. Entry (σ_i, σ_j) gives the probability that σ_j has the same meaning as σ_i. The overall structure of the table corresponds to bounds on the presumed structure of messages received, based on the assumption that cooperating agents communicate things related to the immediate task. This radically constrains the range of possible interpretations, in order to make communication generally feasible; such constraints correspond to the use of such considerations as relevance and context in real-world communication, in order to make interpretation easier, or even possible.

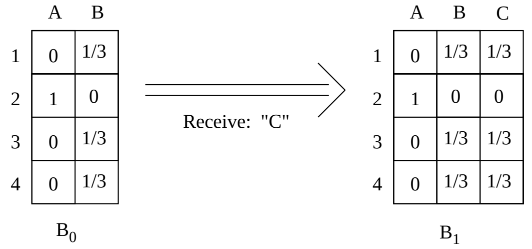
```
Bayes-Filter(Bel(x),d){
NormFactor=0
If d is a perceptual data item z then
    For all x do
        $Bel'(x) = P(z|x)Bel(x)$
        $NormFactor = NormFactor + Bel'(x)$
    For all x do
        $Bel'(x) = NormFactor^{-1}Bel'(x)$
else if d is an action data item u then
    For all x do
        $Bel'(x) = \sum_{x'} P(x|u, x')Bel(x')$
return Bel'(x) }
```

Figure 1. Bayes-Filter Algorithm

Probabilities then play two important roles here. First, the filtering algorithm assigns probabilities to various possible translation-tables, taken as part of the local state of the agent. Second, agents choose actions based upon the probable meaning of recent messages, calculated based on the overall probability that some translation is correct, and the individual probabilities contained in the corresponding table. Language learning interleaves interpretation with action, in a joint process designed to narrow down possible translations, while still acting on current ones, however uncertain they may be.

To cope with particular features of this task, the basic filtering algorithm must be modified in two main ways. First, agents do not enumerate and update *all possible* beliefs: since belief-states themselves contain combinations of continuous probability distributions, there will be infinitely many available at any time. Instead, only those states necessary are generated and updated at each step; for many applications, the procedure is straightforward, and is sound so long as states not generated are properly taken to occur with zero probability. Second, the set of possible belief-states changes over time: since agents do not generally know all the elements of the language to be translated in advance, newly-encountered components must be added to the translation-tables along the way.

As an example, consider a simple gridworld problem: the environment is a 2×2 grid; to identify locations, agents use unambiguous proper names, meaning (1) each grid-square



	A	B
1	0	1/3
2	1	0
3	0	1/3
4	0	1/3

	A	B	C
1	0	1/3	1/3
2	1	0	0
3	0	1/3	1/3
4	0	1/3	1/3

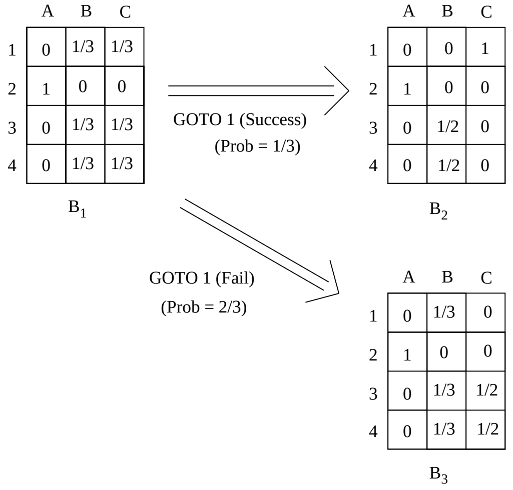
Figure 2. A new message is added.

in the grid has exactly one name, and (2) each name identifies exactly one grid-square. Suppose agent a uses the integers $\{1, 2, 3, 4\}$ to name the four grid-squares; the goal is to find a mapping between these names and those used by another agent. Agents communicate grid locations using their own naming conventions; the other agent receives the message, attempts to interpret it, and proceeds to the most likely square. A reward follows, indicating whether or not the agent has successfully identified the correct location.

Figure 2 shows how agent a adds newly-received messages into its translation-table. We assume that a begins in the sole belief-state B_0 (i.e., a assigns B_0 unit probability). We see that a has previously received two messages, "A" and "B"; further, a has successfully translated the first of these, since $P(A \text{ means } 2) = 1$. Two more features of the table stand out. First, each column sums to 1; a knows the components of its own language, and knows that the distribution for any message must sum properly. Second, rows *do not* sum to 1; before a has seen all of the words in the other language, it cannot know how to fill out each row of its translation-table.

Suppose a now receives a new message, "C"; the belief-state is then updated from B_0 to B_1. A new column is added to the table: since a already knows that the name "2" corresponds to message "A", "C" receives a zero probability for this entry, and all remaining entries are uniformly distributed, reflecting the proper-name model of the language. Since "2" is translated as "A", it will not be translated as any other name, and so the 0 is inserted in the table; further, a has no reason to think "C" any more likely to name one remaining square in the grid than another, and so the remaining probability mass is distributed uniformly.

Having expanded the table, a now chooses an action. Since the probability that "C" means "1", "3", or "4" is the same, a is indifferent. Of course, this need not be true; in general, the choice of an action is computed based on the most likely translation for the current message, weighted by the probability assigned to each possible translation-table. Assume that a chooses "1" as the most likely translation: Figure 3 shows the *action model* update of a's belief-function (the "else" clause of the filtering algorithm in Figure 1) before going to the square named by "1". We assume action outcomes are deterministic: after taking action $Go(1)$, a is sure to end up at the desired square. Furthermore, the reward observed at the end of the action determines precisely whether or not a is correct in its translation (i.e., a receives a particular reward if and only if it has correctly identified the chosen square). Thus, the update process replaces B_1 with two new possible belief-states.



	A	B	C
1	0	1/3	1/3
2	1	0	0
3	0	1/3	1/3
4	0	1/3	1/3

	A	B	C
1	0	0	1
2	1	0	0
3	0	1/2	0
4	0	1/2	0

	A	B	C
1	0	1/3	0
2	1	0	0
3	0	1/3	1/2
4	0	1/3	1/2

Figure 3. The two states possible after $Go(1)$.

Belief-state B_2 reflects the outcome that the translation $(C \text{ means } 1)$ is *correct*. In this case, the entry $(1, C)$ in the table is set to unit probability, and all other entries in the row and column are set to 0. Again, this is a reflection of the unambiguous nature of the name language; more complex update scenarios are possible. Note also that the table in B_2 also updates the probabilities contained in column "B"; since we normalize all columns, the probability mass for entry $(1, B)$ is distributed over the remaining possibilities. Belief-state B_3, for its part, reflects the outcome that the translation $(C \text{ means } 1)$ is *incorrect*. Here, the only column affected is "C"; the entry $(1, C)$ is set to 0, and its probability mass distributed over the remaining entries in that column. Lastly, a assigns predictive probabilities to these two new belief-states before it takes action $Go(1)$, based on what the outcome of that action can tell us about the chosen translation. In this case, $P(B_2|B_1, Go(1))$ is simply the original probability, $1/3$, taken from table B_1, that translation $(C \text{ means } 1)$ is correct. Similarly, the probability of the incorrect translation is $P(B_3|B_1, Go(1)) = (1 - 1/3) = 2/3$.

So long as agents can fully observe their states following any action, the presence of stochastic actions does not fundamentally influence the ability to learn a language. The action-model update step would simply produce more distinct sets of belief states, based on the uncertainty of the outcomes.

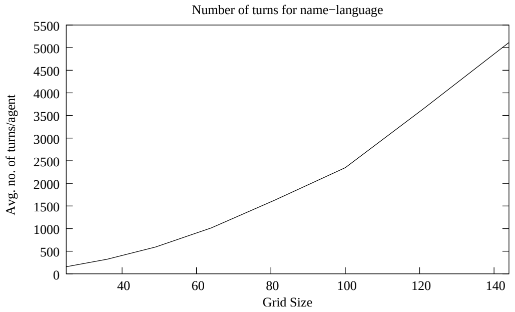


Figure 4. Results for simple-name language.

For this environment, the *sensor model* is also simple: a either receives appropriate reward R for successfully translating "C" or not. If a does receive R, belief-state B_2 is clearly correct and post-observation probability $P(B_2|R)=1$, while $P(B_3|R)=0$ and the latter belief-state can be discarded. If a does not receive R, then the opposite holds. In either case, a is left with a single belief-state on which to base its next actions. Again, we stress that for more complex cases, the relevant updates can be more complicated. In particular, it need not be the case that a is left with but one belief-state at the end of the process of action and observation, since observed rewards may not determine single states, and actions may not have determinate outcomes. Still, the basic principles remain the same.

4. Experiments

Our initial experimental results demonstrate the basic feasibility of the given approach to the language-learning problem, and illustrate how performance is affected by various features, such as the structure of the observations and rewards, or of the language itself. We ran a number of tests involving a basic cooperative task for two agents (a_1 and a_2) in a simple gridworld environment, averaging results over one hundred runs with random starting locations. Work proceeds in turns; on even-numbered rounds:

(1) a_1 is the **actor**, randomly[1] choosing a square and sending a message to a_2 to that effect.

(2) a_2 is the **translator**, choosing a square based on its current translation of a_1's language.

(3) a_2 receives a reward, depending upon its choice, and updates its beliefs accordingly.

On odd numbered rounds, a_1 and a_2 switch roles; the task continues until one agent has successfully translated the other's language. Currently, this involves one agent assigning unit probability to a *complete* translation-table, i.e., every row contains exactly one entry with probability 1 (and the rest 0). Obviously, this could be modified to involve other, perhaps more tolerant, criteria of success.

Language	Turns	Time (s)	Max. Beliefs
Names	156	0.5	1
Coordinates	27	119.3	13812

Table 1. Two languages on a 25×25 grid.

In step (2) of the process, the choice of a square is based on the current message and belief-state of the translator. This belief-state is a probability distribution over translation-tables; for each such table T_i, let P_i be the probability that T_i is the correct translation. Each table T_i maps components of received messages to possible meanings; for any complete received message σ_j (in the actor's language) and meaning σ_k (in the translator's own language), let $T_i(\sigma_k, \sigma_j)$ be the probability computed from table T_i that (σ_j means σ_k). The most likely meaning of the message σ_j is thus calculated as that σ_k satisfying: $\max_k \sum_i P_i \cdot T_i(\sigma_k, \sigma_j)$. The various possible tables T_i are chosen based on the presumed structure of the two given languages; the probabilities P_i are updated using the Bayes-Filter algorithm.

Our first study involves a simple language of *unambiguous names* (as in Section 3.1.1). Each agent attempts to learn the other's mapping from names to grid-squares. Rewards are simple: translators receive a 0 reward for correctly identifying the square chosen, and otherwise the reward is 1; translations are updated in either case. Figure 4 charts the average number of turns as translator an agent must take to arrive at a complete translation for this language. The ratio of grid-size to number of turns is relatively stable: for grid size $G \times G$, the number of turns is roughly $G^2/4$. We can see that agents pursue a decision-theoretically optimal course of action here. Translators initially assign novel messages a uniformly-distributed probability of naming squares for which the name is not already known; further, after visiting any square, the translator knows the exact (0/1) probability that the latest message names it, based on the reward received. Once some message σ_i is known to correspond to the name of square x, the translator always visits x upon receiving σ_i. Conversely, the translator never visits x once it knows that current message σ_j cannot name it. Finally, for messages not yet translated with certainty, the most probable translation is chosen (breaking ties randomly). Since the translation-action strategy is thus optimal with respect to expected reward, it will not in general be possible to do better in terms of average number of attempts before achieving a complete translation.

Such a simple example shows the elementary feasibility of the filtering approach, but we are also interested in testing more complicated languages and reward structures.

[1] Our ongoing research investigates cases in which agents can choose particular messages non-randomly, to guide the learning-process.

Within the same gridworld context, we also investigate messages in a language of (x, y) coordinate-pairs. Due to this structure, translation-updates carry more information than when simple names are concerned. However, the number of possible translations may increase drastically after each update. In the name-language, the reward observed after any action determined a single possible belief-state for the translator: either the translation of some name is known with certainty, or it is absolutely certain that one particular translation is incorrect. In a language of coordinate-pairs, with a basic success/failure reward structure, things are not so simple. If selection of some grid-square is successful, of course, the translation of some message (σ_x, σ_y) is known with certainty, and any other possibilities for that pair are eliminated. If selection is unsuccessful, however, there are three options: either both translations of σ_x and σ_y are incorrect, or only one of them is. In the worst case, then, the essentially uninformative (0/1) reward structure can cause a large increase in the number of belief-states an agent has to entertain, affecting overall performance. Indeed, this increase may be so great that while translation uses a smaller number of *turns*, overall *time* spent is much worse, as agents have to update many more beliefs in any given turn (see Table 1). In general the problem becomes potentially intractable for the combination of the coordinate-language and an essentially uninformative reward.

To improve the situation, we consider other possible reward structures, which give the translator more information about what may have gone wrong. These are: **Reward 0/1/2**: agents receive 0 if translation is correct, 1 if translation fails but one coordinate is translated correctly, and 2 if failure results because both coordinates are translated incorrectly; **Reward 0/1/2/3**: agents receive 0 if translation is correct, 1 if it fails but σ_x is translated correctly, 2 if it fails but σ_y is translated correctly, and 3 if failure results because both are translated incorrectly. The more informative rewards greatly improve performance in terms of all problem-dimensions: number of turns before translation is successful, maximum number of beliefs that an agent must entertain at any time, and overall time of completion.

Figure 5 compares results for the name-language and for the different variations of the coordinate-language, in the context of a 25×25 grid. Scale here is logarithmic, with values normalized to the Reward 0/1/2/3 case; results average over 100 random runs. In general, the extra information provided by the coordinate-language as opposed to the name-language significantly reduces the number of turns taken. Further, with respect to the coordinate-language alone, the more informative reward functions lead to significantly better results in terms of maximum number of belief-states ever updated in one pass of the filtering algorithm, and thus in terms of overall time taken. Such relations are not absolute, however. For instance, although the maximum number of beliefs is somewhat larger for the Reward 0/1/2 case than for the name-language, the increase in information gained by using coordinates still decreases the number of turns enough to lead to a decrease in overall time taken in the former case. In the single case shown, and in general, the performance of the two cases with more structured rewards was essentially the same; while the 0/1/2/3 structure led to slightly improved performance, the differences were not significant. Finally, Figure 6 shows the increase in the number of turns taken as the grid grows in size for both the name-language and for the coordinate-language with 0/1/2 reward structure. We did not test the name-language for very large grids, since the time taken quickly became unmanageable; in any case, the difference is evident for the limited range considered.

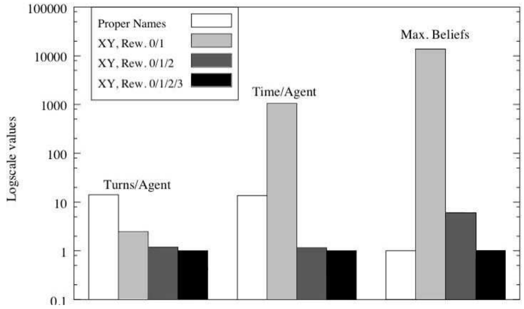


Figure 5. Different languages on a 25×25 grid.

5. Conclusions and Future Work

Automated language learning is clearly a challenging problem. This paper presents a framework to study this problem based on decentralized Markov decision processes and reinforcement learning. We adapt Bayesian filtering techniques to our decentralized environment. Agents are assumed to know optimal local policies to local goals, although they lack one language of communication to exchange their observations. Agents learn to correctly interpret others' messages while acting optimally to some goal, computed based on the current interpretation of the most recent message. Our initial studies show how simple languages can be learned and how characteristics of both these languages and of the rewards and observations available in the local environment can affect this process.

Our results also show the benefit—even necessity—of treating received messages as having meaningful structure. For some elementary cases, it may be possible to duplicate our results using generally naive learning methods. For example, agents might simply extend the state-space of their MDPs by adding received messages to their observations, and use straightforward reinforcement learning approaches over the expanded state-space. However, the improved re-

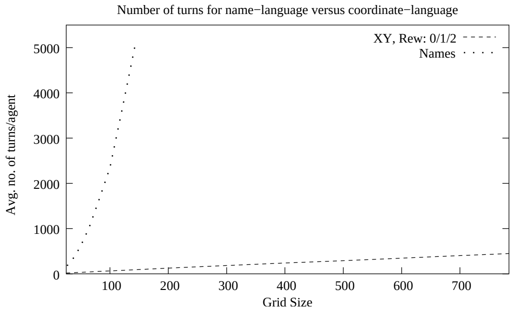


Figure 6. Names vs. coordinates.

sults we achieve using coordinate-languages, as opposed to simple names, demonstrates how paying attention to the structure of messages *as messages* (as opposed to treating them as unanalyzed features of the environment) can speed learning. Language is thus one problem domain for which considerations of structure become important to the ability to learn. Further, by treating messages as having meaningful structure, it becomes possible to learn languages that are portable over different environments and tasks.

We intend to continue studying theoretical aspects of this learning problem as well as the scalability of our approach. The first is related to what classes of languages can be learned efficiently. The second will consider generalizations of our approach, for example by clustering reachable or otherwise similar states, and learning how to communicate in those states. Learning speed can improve when it is possible to generalize over messages. (e.g., when features are functions of other features in the same message). We are also interested in analyzing our approach, when communication may be assigned a cost, and when the process may not be jointly fully-observable.

References

[1] D.S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819—840, 2002.

[2] Andrea Bonarini and Piera Sassaroli. Opportunistic multi-model diagnosis with imperfect models. *Information Sciences*, 103(1-4):161–185, 1997.

[3] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.

[4] D. Davidson. *Inquiries into Truth and Interpretation*. Oxford UP, Oxford, England, 1984.

[5] E.H. Durfee and V.R. Lesser. Using partial global plans to coordinate distributed problem solvers. In A.H. Bond & L. Gasser, eds., *Readings in Distributed Artificial Intelligence*, pp. 285–293. Morgan Kaufmann, San Mateo, CA, 1988.

[6] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, ed., *Software Agents*. MIT Press, 1997.

[7] L. Firoiu, T. Oates and P. Cohen. Learning Regular Languages from Positive Evidence. In *Proc. of the* 20^{th} *Annual Meeting of the Cognitive Science Society*, pp. 350—355, 1998.

[8] P.J. Gmytrasiewicz, M. Summers, and D. Gopal. Toward automated evolution of agent communication languages. In *Proc. of the* 35^{th} *Hawaii International Conf. on System Sciences*, 2002.

[9] C.V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proc. of the* 2^{nd} *International Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 137—144, Melbourne, Australia, 2003.

[10] C.V. Goldman and S. Zilberstein. Mechanism design for communication in cooperative systems. *Proc. of the* 5^{th} *Workshop on Game Theoretic and Decision Theoretic Agents*, Melbourne, Australia, 2003.

[11] C.V. Goldman and S. Zilberstein. Decentralized control of cooperative systems. *University of Massachusetts at Amherst, Computer Science*, Technical Report 03-36, 2003.

[12] B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269—357, 1996.

[13] NASA. Mars Climate Orbiter Failure Board Report. *Available at:* `ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf`, 1999.

[14] H. Putnam. The meaning of 'meaning'. In *Mind, Language and Reality: Philosophical Papers*, Volume 2, pp. 215–271. Cambridge University Press, Cambridge, England, 1975.

[15] D.V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *J. of Artificial Intelligence Research*, 16:389–423, 2002.

[16] W. Van Orman Quine. *Word and Object*. The MIT Press, Cambridge, MA, 1960.

[17] N. Sharygina and D. Peled. A Combined Testing and Verification Approach for Software Reliability. In *FME 2001, LNCS 2021*, pp. 611–628, 2001.

[18] L. Steels and P. Vogt. Grounding adaptive language games in robotic agents. In *Proc. of the* 4^{th} *European Conf. on Artificial Life*, 1997.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141, 2001.

[20] P. Vogt and H. Coumans. Investigating social interaction strategies for bootstrapping lexicon development. *J. of Artificial Societies and Social Simulation*, 6(1), 2003.

[21] J. Wang and L. Gasser. Mutual online concept learning for multiple agents. In *Proc. of the* 1^{st} *International Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 362–369, Bologna, Italy, 2002.

[22] H. A. Yanco. Robot Communication: Issues and Implementation. Master Thesis, MIT, 1994.