Auton Agent Multi-Agent Syst
DOI 10.1007/s10458-006-0005-z

Cobot in LambdaMOO: An adaptive social statistics agent

Charles Lee Isbell, Jr. · Michael Kearns · Satinder Singh · Christian R. Shelton · Peter Stone · Dave Kormann

Published online: ■
Springer Science+Business Media, LLC 2006

Abstract We describe our development of Cobot, a novel software agent who lives in LambdaMOO, a popular virtual world frequented by hundreds of users. Cobot's goal was to become an actual part of that community. Here, we present a detailed discussion of the functionality that made him one of the objects most frequently interacted with in LambdaMOO, human or artificial. Cobot's fundamental power is that he has the ability to collect *social statistics* summarizing the quantity and quality of interpersonal interactions. Initially, Cobot acted as little more than a reporter of this information; however, as he collected more and more data, he was able to use these statistics as *models* that allowed him to modify his own behavior. In particular,

C. L. Isbell, Jr. (✉)
College of Computing, Georgia Institute of Technology,
Atlanta, GA 30332, USA
e-mail: isbell@cc.gatech.edu

M. Kearns
Department of Computer and Information Science, University of Pennsylvania,
Philadelphia, PA, USA
e-mail: mkearns@cis.upenn.edu

S. Singh
Department of Computer Science and Engineering, University of Michigan,
Ann Arbor, MI, USA
e-mail: baveja@umich.edu

C. R. Shelton
Computer Science and Engineering Department, University of California at Riverside,
Riverside, CA, USA
e-mail: cshelton@cs.ucr.edu

P. Stone
Department of Computer Sciences, University of Texas at Austin,
Austin, TX, USA
e-mail: pstone@cs.utexas.edu

D. Kormann
AT&T Shannon Labs,
Florham Park, NJ, USA
e-mail: davek@research.att.com

cobot is able to use this data to "self-program," learning the proper way to respond to the actions of individual users, by observing how others interact with one another. Further, Cobot uses reinforcement learning to proactively take action in this complex social environment, and adapts his behavior based on multiple sources of human reward. Cobot represents a unique experiment in building adaptive agents who must live in and navigate social spaces.

Keywords Reinforcement learning · Social modeling · Chat agents · Autonomous agents · Game theory · Believable agents

1. Introduction and motivation

The internet is a medium where large groups of people build social communities. This presents both a challenge and an opportunity for artificial intelligence and software agent researchers. In such communities, agents may do more than filter mail and retrieve price quotes for consumer items: they may be legitimate, if still limited, participants in close-knit social environments.

This paper presents Cobot, a software agent who has lived in an active online community frequented by several hundred users (LambdaMOO, which we describe in requisite detail in the next section). His goal is to interact with other members of the community and to become a vital, useful and accepted part of his social fabric [16].

Toward this end, Cobot tracks actions taken by users, building statistics on who performs what actions, and on whom they use them. For example, Cobot tracks which users converse with each other most frequently. Using his chatting interface, Cobot can answer queries about these and other usage statistics, and describe the statistical similarities and differences between users. This information also provides Cobot with a user model that may be used for learning and imitation. In fact, Cobot is a *self-modifying* agent. He[1] sometimes takes unprompted actions that have meaningful social consequences. Rather than using complex hand-coded rules specifying when various actions are appropriate (rules that, in addition to being inaccurate, might quickly become stale), Cobot used reinforcement learning to adapt a policy for picking actions as a function of the *the individual and communal preferences* of users [17].

During Cobot's years in LambdaMOO, he became a member of the community. As we will see in subsequent sections, for much of that time, users interacted with Cobot more than with any other user (human or artificial), took advantage of the social statistics he provided, conversed with him, and discussed him and the implications of his existence, for good or ill.

Following the pioneering studies of [13, 14], we present transcripts establishing the sociological impact of Cobot. We compare the techniques that we have used to implement Cobot's set of social skills to those of previous MUD agents, such as Julia [13], and discuss how these techniques affect user expectations. In addition to the more anecdotal evidence provided by transcripts, we provide quantitative statistical support for our belief that Cobot became not only part of the social environment in which he resided, but significantly altered it as well.

[1] Characters in LambdaMOO all have a specified gender. Cobot's description, visible to all users, indicates that he is male.

```
(1)  Buster is overwhelmed by all these paper deadlines.
(2)  Buster begins to slowly tear his hair out, one strand
     at a time.
(3)  HFh comforts Buster.
(4)  HFh [to Buster]:  Remember, the mighty oak was once
     a nut like you.
(5)  Buster [to HFh]:  Right, but his personal growth was
     assured.  Thanks anyway, though.
(6)  Buster feels better now.
```

Fig. 1 A typical conversation in LambdaMOO. Notice the use of directed speech and other *verbs* to express emotional states. For example, line (3) was generated by typing "comfort buster". It is worth pointing out that "comfort" is not a built-in command, but a user-created verb that was made public. Most verbs are of this nature, and the set of available verbs changes from week-to-week

The paper begins with a brief history of LambdaMOO. We then detail the two major components of Cobot's functionality—his ability to provide social statistics, and his conversational abilities—and quantify and discuss their impact. Next we describe our development of Cobot as a reinforcement learning agent, detailing his RL action space, reward mechanisms and state features, summarizing our findings on the usefulness of applications of RL techniques in challenging social spaces. Finally, after an examination of privacy issues and our approach to them, we discuss the implications of artificial social actors such as Cobot.

2. LambdaMOO

LambdaMOO, founded in 1990 by Pavel Curtis at Xerox PARC, is one of the oldest continually-operating MUDs, a class of online worlds with roots in text-based multiplayer role-playing games. MUDs (multi-user dungeons) differ from most chat and gaming systems in their use of a persistent representation of a virtual world, often created by the participants, who are represented as characters of their own choosing. The mechanisms of social interaction in MUDs are designed to reinforce the illusion that the user is present in the virtual space. LambdaMOO is actually a MOO (for "MUD, object-oriented"): a MUD featuring a powerful object-oriented programing language used to manipulate objects and behaviors in the virtual world.

LambdaMOO is modeled after a mansion, and appears as a series of interconnected rooms that are populated by users and objects (who may move from room to room). Each room provides a chat channel shared by just those users in the room (users can also communicate privately), and typically has an elaborate text description that imbues it with its own look and feel. In addition to speech, users express themselves via a large collection of *verbs*, allowing a rich set of simulated actions, and the expression of emotional states, as is illustrated in Fig. 1.

Lines (1) and (2) are initiated by emote commands by user Buster, expressing his emotional state, while (3) and (4) are examples of emote and speech acts, respectively, by HFh. Lines (5) and (6) are speech and emotes by Buster (in our transcripts the name of the user initiating an action always begins the description of that action or utterance). Though there are many standard emotes, such as the use of "comfort" in line (3) above, the variety is essentially unlimited, as players have the ability to create their own emotes.

The rooms and objects in LambdaMOO are created by users themselves, who devise descriptions, and control access by other users. Users can also create objects with methods (or *verbs*) that can be invoked by other players.[2] LambdaMOO is thus a long-standing, ongoing experiment in collective programing and creation, with often stunning results that can only be fully appreciated firsthand. Inventions include technical objects, such as the *lag meter*, which provides recent statistics on server load; objects serving a mix of practical and metaphorical purposes, such as elevators that move users between floors; objects with social uses, such as the *birthday meter*, where users register their birthdays publicly; and objects that just entertain or annoy, such as the *Cockatoo*, a virtual bird who occasionally repeats an utterance recently overheard (often to amusing effect). There is also a long history of objects that can be viewed as experiments in AI, as we will discuss below.

LambdaMOO's long existence, and the user-created nature of the environment, combine to give it with one of the strongest senses of virtual community in the online world. Many users have interacted extensively with each other over a period of years, and many are widely acknowledged for their contribution of interesting objects. LambdaMOO has been the subject of articles and books in many different literatures, including the popular press [7], linguistics and sociology [5], computer science, and law.[3] The complex nature of the LambdaMOO community goes a long way towards explaining why it is difficult to simply characterize what users "do" on LambdaMOO. As in real life, users engage in a wide variety of activities, including social activity, programing, and exploring.

MUDs are an attractive environment for experiments in AI. We believe, as others have before us, that the relative technical ease of implementing sensors and actuators in such a well-structured software environment (compared, for instance, with robotics), combined with an intelligent user population predisposed to interact with programed artifacts (sometimes called puppets), make MUDs an obvious playground for AI ideas. Unsuprisingly, there is a rich history of automated agents and constructions in LambdaMOO: Markov chainer, an object that builds a Markov model from the conversations in a room; Dudley, a well-known agent with simple chatting abilities; an automated bartender who provides virtual drinks and small-talk; and many others. There are also object classes allowing users to specialize and create their own AI agents. The agent Julia, a descendant of Colin (created by Fuzzy Mauldin [22]), who once resided in a different MUD, is perhaps the closest ancestor of Cobot. We will discuss both Julia and her analysis by Foner [14], which has strongly influenced our thinking, throughout the paper where appropriate.

3. Cobot

Most of Cobot's computation and storage occurs off-LambdaMOO. He is built using *the Cobot platform*, an agent architecture that uses a directed graph-based metaphor to define a standard for describing a wide-range of virtual environments, including the web, filesystems, and MUDs. The standard allows for mostly domain-independent

[2] Everything in LambdaMOO is an object, and every event is the invocation of a verb on some object, including speech (usually invocations of the **tell** verb). The LambdaMOO server maintains the database of objects, and executes verbs. As of this writing, the database contains 118,154 objects (not all are valid), including 5158 active user accounts.

[3] LambdaMOO has its own rather intricate legal system.

implementations of the basic operations that any virtual agent will have to perform. A complete discussion of the platform is beyond the scope of this paper. For discussions of architectures that address similar issues see [1,9,12,15,18,19,20,23,25,30].

Cobot appears to be just another user. Once connected, he usually wanders into the LambdaMOO Living Room, where he spends most of his time.[4] The Living Room is one of the central public places, frequented by many regulars. It is also located next to the Linen Closet, where guests tend to appear, so it is also frequented by users new to LambdaMOO. There are several permanent objects in the Living Room, including a couch with various features, a cuckoo clock, and the aforementioned Cockatoo. The Living Room usually has between five and twenty users, and is constantly busy. Over a period of about a year, Cobot counted over 2.9 million separate events (roughly one event every 10–11 seconds).

As a regular of the Living Room, Cobot sought to engage other users. His social development can be divided into four distinct stages: inanimate object, social statistics tool, an interactive conversationalist, and a socially adaptive agent.

In the beginning, Cobot was socially inept: he sat in the Living Room and did nothing but answer one or two basic questions about why he was there. When spoken to in an unanticipated way, he did not respond. In other words, he was little more than a new piece of furniture. Not surprisingly, Cobot generated only a small amount of interaction.[5]

In the next sections we explore the next three stages of Cobot's development and see how these changes impacted both Cobot's popularity and his environment.

4. Social Statistics

Previous work on agents in MUDs [13, 14] has argued that being able to provide information of interest or value to other users aids the social acceptance of the agent. Because Cobot is intended to be primarily a social creature, we chose to have Cobot build and maintain what might be thought of as a *social map* of user interactions in LambdaMOO. In particular, Cobot maintains:

- For each user he encounters:
  - a histogram of verbs used by that user
  - a histogram of verbs that have been used on that user
- For each verb invoked in his presence:
  - a histogram of the users that have invoked it
  - a histogram of the users that have been its target
- For each pair of users Cobot has seen interact:
  - a histogram of verbs they have been used on each other

For both technical and ethical reasons, this information is gathered only for objects and users that are in Cobot's presence. The details of acquiring such information reliably are fairly straightforward, but beyond the scope of this paper. For a discussion, we refer the reader to [6].

These statistics define a rich graph of social interactions. For example, it is possible to determine which users interact with one another the most, who the most "popular" users are, the types of actions any given user tends to use, and to perform a wide

[4] Cobot has visited about 1680 rooms. A partial map appears in Fig. 2.

[5] Interestingly, from day-to-day, the amount of interaction was fairly consistent.

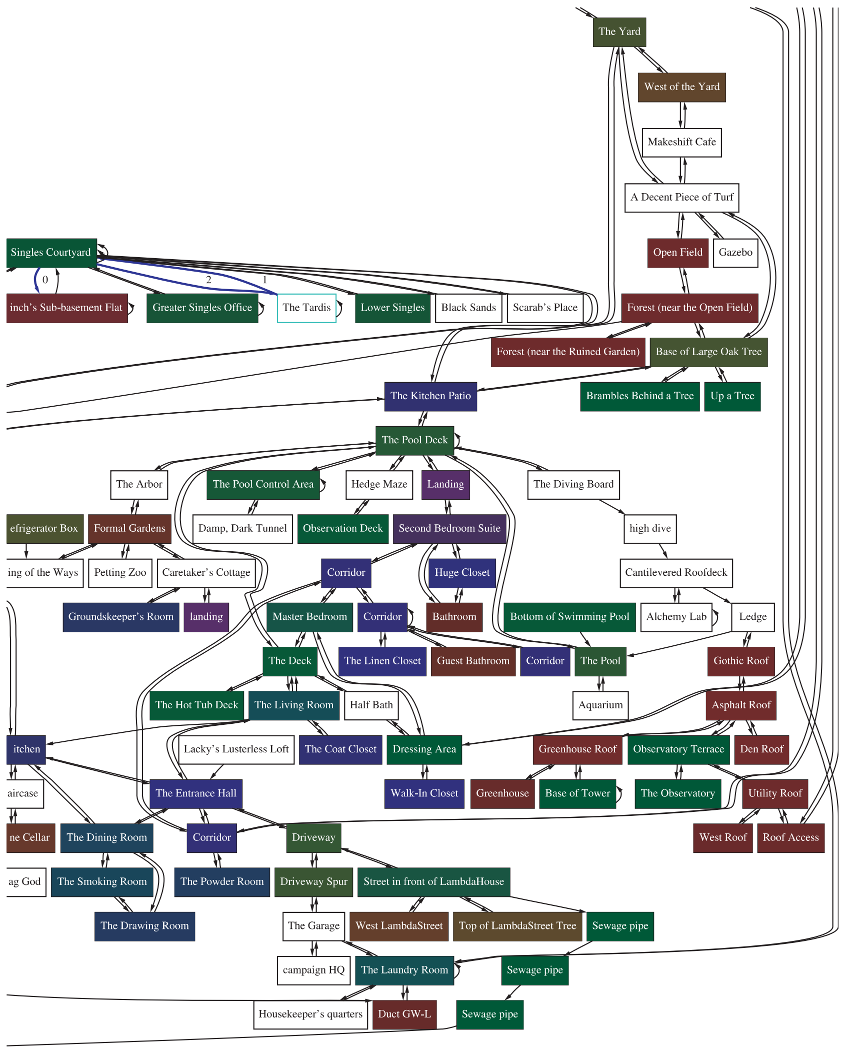

Fig. 2 Physical map. A partial map of LambdaMOO. Nodes are rooms, and directed edges indicated exits from one room to another. Colors reflect a basic similarity between nodes based purely on the text for the room's description. Cobot has visited over 1667 rooms in his lifetime. There is an entire sub part of LambdaMOO called the RPG—a place where role playing games occur—that cobot has never explored

range of other graph-theoretic computations. Figure 5 is a graph that provides one visualization of this kind of social data.

Using this information, Cobot is able to answer natural-language-like queries about social interactions in the Living Room. For example:

```
HFh [to cobot]: relate me to Buster
cobot whispers, "Here are your relationships with Buster. You like to
use: - (62%), poke (7%), hug (3%), eye (3%), nod (2%), hi5, h5, zap,
comfort, and grin on each other. Buster is ranked #14 on your list of
playmates. You are ranked #1 on Buster's list. Your socializing over-
```

Table 1 A partial list of social statistics questions that may be asked of Cobot

tell me about *verb***-o-meter**	Lists which users use (and get used by) *verb* the most.
tell me about me	Tells the questioner various facts about herself, including the verbs she like to use most, and the verbs most often directed at her.
who are your playmates	Lists those who interact with Cobot the most.
who loves me	Tells the questioner those with whom she interacts the most.
who acts like me	Tells the questioner users who perform actions similar to hers.
who does not act like me	Tells the questioner users who do not perform actions similar to hers.
relate me to *user*	Indicates how the questioner interacts with *user*, providing a ranking on each other's list of playmates, how many playmates they have in common, and how similarly they act.
tell me my *verb***-buddies**	Tells the questioner the users she uses *verb* on and who use *verb* on her.
who is spammy	Lists those users who generate and are the targets of the most verbs.

Cobot is able to answer questions about the statistics he has gathered. Answers are based on everything he has ever seen; however, a user can also ask for a report based only on recent interactions simply by adding the adverb "recently" to his request. Here, the term "act like" refers to distribution of verb usage. The similarity measure is the cosine of the angle between verb count vectors (based on a standard information retrieval technique)

```
lap is 75.4% and your playmate overlap is 33.7%. Your actions have a
similarity of 95.9% but ignoring common speech verbs it's 58.3%. Oth-
ers act on you with a similarity of 96.6% but ignoring common speech
verbs it's 81.9%."
```

This particular query yields a great deal of information. Cobot first reports how and how often HFh and Buster interact with one another (the "-" verb is one method for directing speech; the first line above is generated by HFh typing "-cobot relate me to Buster."). Cobot then provides measures of similarity of social circles and similarity of action. Each of these measures is calculated using a cosine measure (a common information retrieval technique) between histograms of action use or inter-user interactions. Table 1 lists more of the queries that one can make of Cobot.

After a month of gathering statistics, the social statistics query feature of Cobot was made available to LambdaMOO users. As Fig. 3 shows, the results were immediate and dramatic. Initially, before he had any real functionality, interaction with Cobot was constant but low (three per 1000 events). After the introduction of his new abilities, the number of interactions directed at Cobot jumped significantly (now over 50/1000 events, more than double that of the next most popular Living Room denizen).[6]

While these graphs quantify the sudden rise in Cobot's popularity, they cannot express the extent to which he altered (for better or worse) social interaction in the Living Room. Users began to converse with each other on what they were learning about their relationships and similarities:

```
Snow_Crash [to Medb]: Cobot says you act like me. Stop.
Medb cracks up laughing at Snow_Crash!
Medb [to Snow_Crash]: How do you know it's not you acting like me?
Medb tries to decide whether she or Snow_Crash should feel more in-
sulted...
```

[6] User HFh (author Isbell) also experienced a sudden equal jump in interaction. As Cobot's primary human ambassador at the time, he spent much of that period answering questions, explaining functionality, and fielding requests for new abilities.

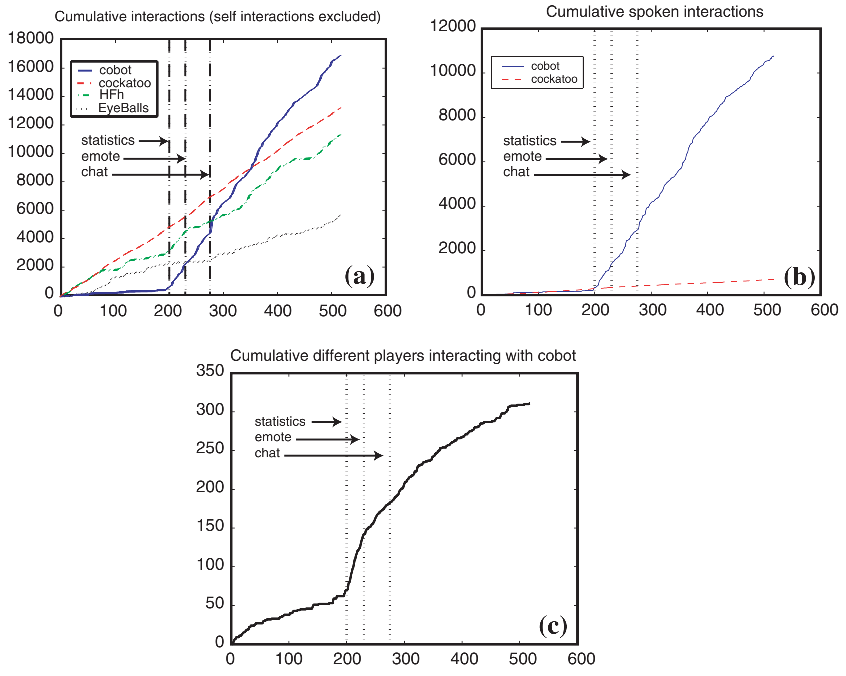

Fig. 3 Cumulative interactions with objects in the Living Room. **(a)** Cumulative number of verbs (speech verbs, hugs, waves, etc.) directed towards various Living Room denizens: Cobot, the Cockatoo, and the two human users most interacted with during this period. The *x*-axis measures cumulative events (in thousands) of any type in the Living Room, while the *y*-axis measures cumulative events directed at the indicated user. Each dashed vertical line indicates the introduction of a major new feature on Cobot (from the left, his social statistics, his emoting abilities, and his extended chat abilities). A straight line—such as the Cockatoo's—indicates constant interaction. By contrast, Cobot's curve shows sudden changes in the slope coinciding with new features. Note that even when the slope levels off afterwards, it remains higher than it was previously, indicating long-term impact. Judging from cumulative interaction, Cobot is the most popular user in the Living Room. **(b)** Cumulative *speech* acts directed at Cobot and the Cockatoo. Clearly users interact with the two artifacts differently. Most of the interaction with Cockatoo is explained by users invoking its **gag** verb to silence it. Still, note that there is constant non-zero daily spoken interaction with it as well, indicating that users are willing to talk to even such a relatively simple creature. Also, note that before his new abilities surfaced, Cobot enjoyed a comparable amount of spoken interaction. **(c)** Variety of users interacting with Cobot. Over time, users brought others into the Living Room to play with him. Also, users who were already in the Living Room but ignoring Cobot began interacting with him when they saw others using his abilities

```
...
Gabaldon [to cobot]: loudly relate me to sparklebug
cobot [to Gabaldon]: Here are your relationships with Sparklebug ...
Sparklebug is ranked 7 on your list of playmates. You are ranked 19
on Sparklebug's list. ...
Gabaldon [to Sparklebug]: I can't play with you so much anymore, you
do realize. You don't like me nearly as much as I seem to like you.
Gabaldon cries at Sparklebug.
...
Spitboy has a new strategy. He stops by the Living Room, asks the
cobot who's like him and if cobot tells him one of the people present
is like him, he starts talking to them.
Spitboy says, "I haven't caught anyone who's like me, yet."
Spitboy [to Kimmy]: Shame, innit?
Kimmy says, "sure."
Kimmy [to cobot]: who is like me?
```

After observing these responses to the information provided by Cobot, we implemented changes to encourage such behavior. For example, as seen earlier in HFh's request about Buster, Cobot provides relative rankings of interactions between users. Players seemed appreciative of the ability to gain deeper insight into their interactions, and used the information accordingly. Access to social statistics resulted in other changes as well, some quite unintentional. For example, users began actively competing to move up on Cobot's list of playmates, or to be ranked the "most hugged."

```
cobot [to Stevage]: Here are my favorite playmates: Sparklebug, HFh, ...
Sparklebug beams brightly. Stevage might need some shades for that.
Stevage winks to cobot.
cobot winks to Stevage.
Sparklebug [to Stevage]: I've worked SO hard to beat HFh!
```

These social consequences were beyond our expectations, sometimes so severe that we needed to implement several social controls (as we shall see later). The important point is that Cobot provided a useful service (as have other agents, such as Julia) that captured information of constantly evolving and long-term interest to his community. The result was that Cobot had an immediate and profound impact on the social interactions within his sphere.

5. Chat and emote abilities of Cobot

Cobot's social statistic functions are what we might view as his "purpose," or the particular service that he provides. While the evidence suggests that such a service can lead to a great deal of interaction, we follow researchers before us in believing that to truly integrate into an online society like LambdaMOO, Cobot must do much more [2,10]. In particular, LambdaMOO is a world that is rich in both conversation and emoting, and in this section we detail our efforts to enable him to participate meaningfully in both.

Emoting. As users became more enamored or frustrated with Cobot's social statistics, they began to express their opinions via directing standard or personalized emotes such as hugging, waving, pouncing and kicking towards Cobot:

```
cobot exclaims, "Hi! I'm awake!"
Sparklebug [to cobot]: I've missed you SO much!
Sparklebug snuggles with cobot.
...
cobot [to Ratman]: Hello, Ratman.
Ratman takes cobot over his knee and spanks its bottom.
```

Consequently, it seemed natural to equip cobot with the ability to respond to such actions appropriately. Like others before us, we found it effective to program several plausible replies to any given scenario, and to randomize between the responses (including doing nothing). This is a simple technique for introducing variability in Cobot's behavior, and seems to enhance his apparent intelligence for many users (see Table 2).

Table 2 Additional Cobot commentary by LambdaMOO users

Guest [to cobot]: Do you know anyone I should send this home vasectomy kit to?	Guest comes out of the closet (so to speak...).
Sparklebug peers at Guest suspiciously.	cobot [to Guest]: Sunshine! You validate me!
cobot [to Guest]: This is a joke in very poor taste.	Guest [to cobot]: nice.
Guest [to HFh]: He's good.	cobot [to Guest]: We've had a nice snooze.
Guest [to HFh]: Have you been tinkering with him?	Guest [to cobot]: so more charitable than usual?
HFh says, "I wish I could take credit, but I think he's evolving."	cobot [to Guest]: But my dreams are a lot emptier than yours.
Guest [to cobot]: You're evolving?	Fawn_Guest [to cobot]: They can't possibly be any emptier than mine
cobot nods to Guest.	Guest [to cobot]: oh.
cobot says, "CITIZENS: ACIDHORSE DOES NOT EXIST."	cobot [to Guest]: Your cry for help is transparent.
Guest [to cobot]: he does so, I'm sure I've seen him	...
cobot [to Guest]: What does this prove?	Stevage says, "cobot is pretty damn human I have to admit... I asked him 3 questions and he gave sensible replies to them all."
Guest [to cobot]: my hovercraft is full of eels	
Guest [to HFh]: He just made perfect sense. damn him.	...
cobot [to Guest]: Heavens!	Valley [to HFh]: cobot's genuinely cool...and I'm not even a techie :)
...	Guest says, "Cobot frightens and confuses me, yet I cannot turn away"
deLaMer says to HFh, "you did good work on cobot."	HFh [to Inimeg]: He *is* a player.
HFh [to deLaMer]: Thank you.	HFh [to Inimeg]: He has always been a player.
deLaMer says, "it's starting to make sense. that's spooky."	Inimeg says, "damn the line just gets blurrier and blurrier"
Blue_Guest [to HFh]: I thought it was a person once.	...
Technicolor_Guest says, "cobot is an evil mastermind lurking behind the false guise of a mere bot"	A better, kinder Ratman puts on his Ratman's Official Church of Cobot Memership Badge.

```
Melon hugs cobot.
cobot hugs Melon warmly.
...
Stevage kisses cobot lightly. Stevage causes cobot to blush a deep crim-
son. cobot [to Ratman]: Welcome, special playmate of mine!
Ratman spits at cobot
cobot [to Ratman]: You'll be first up against the wall when the rev-
olution comes.
```

Simple Chatting. Following an Eliza-like approach, we initially matched incoming utterances against a collection of meticulously hand-coded regular expressions designed to catch common phrases directed at Cobot. As others have discovered, for certain classes of incoming utterances, it is helpful to respond with fairly general and leading conversational utterances, while other times it is possible to be rather specific:

```
killa [to cobot]: how old are you?
cobot [to killa]: I am older than you think.
...
```

```
lackluster [to cobot]: why are you carrying a shield?
cobot [to lackluster]: I think many of the dumber MOOers think I'm a
garbage can.
```

Just as users expect to have their hugs to Cobot returned, his heaviest users enjoy having Cobot actually initiating recognition of them. We thus made occasional departures from our informal policy of having Cobot only respond to activity initiated by users, for instance by having Cobot initiate greetings to certain users who interact with him frequently, as in his (rebuffed) greeting to user Ratman above.

Extended Chat. So far we have described a limited set of chat and emote abilities, carefully crafted but based on familiar techniques. In the end, this proved to be impractical: the variety of utterances directed at Cobot is much greater than the variety of emotes. In any sufficiently longstanding experiment of this kind, the limitations of these methods eventually become apparent to both inventor and user. Indeed, not long after Cobot began enjoying sustained usage, many users began to recognize some of his canned responses, and made comments longing for better chatting ability. At this point we inaugurated a novel experiment, in which some of Cobot's utterances were generated from a small number of large and publicly available online documents meeting the following desiderata:

- The sentences could plausibly serve as utterances by Cobot, at least in some contexts.
- The documents should be sufficiently large that repeated utterances will be unlikely.
- The documents should be on fairly specific and recognizable topics. Rather than designed to enhance general conversational ability (which remains handled in part by the mechanisms already described), the utterances generated by the documents are meant to give Cobot "personality".
- The utterances generated by the documents are entertaining and sometimes controversial (to spur further conversation).

Any utterance directed towards Cobot that is not recognized as a request for social statistics becomes a candidate for the following process. Words in the incoming utterance are matched to words appearing in sentences in the documents, assigning to each sentence a weight based on the number of matching words. A minimum length on the words matched provides a parameter that can adjust the difficulty of matching sentences in each document independently. If no match is found, the utterance is handled by one of the mechanisms mentioned earlier; otherwise, with a (large) fixed probability Cobot randomly chooses a sentence to utter according to the distribution defined by the weights.

The effectiveness of this approach depends on the choice of documents. We first fielded the well-known Unabomber Manifesto by Theodore Kaczynski, a lengthy, didactic essay on the evils wrought on society by science, and the industrial revolution. The resulting response was immediate and dramatic, resulting in a steep rise in experimentation and interaction with Cobot (note the rightmost dashed vertical line in Fig. 3(a)). The new functionality resulted in endless conversation, both with and about Cobot, and allowed users to attribute to him all manner of intelligence and attitudes:

```
Technicolor_Guest [to cobot]: What's your religion?
cobot [to Technicolor_Guest]: technology unless you REALLY believe in
```

```
that religion yourself and find that it arouses a deep, strong, gen-
uine response in many other people.
...
cobot [to DragonBoi]: The line of conflict should be drawn between
the mass of the people and the power-holding elite of industrial
society.
DragonBoi eyes cobot warily.
cobot [to DragonBoi]: You have something cute in your eye.
DragonBoi [to HFh]: you're bot is turning into a communist.
DragonBoi [to cobot]: so what are you gonna do about it?
DragonBoi [to HFh]: did you see what it told me? you are turning it
into a communist!
Wikked [to cobot]: are you a communist?
DragonBoi [to cobot]: are you now, or ever been a member of the com-
munist party?
cobot mumbles something in binary.
DragonBoi [to cobot]: so you're taking the fifth are you? well this
is a special commision, that won't work here!
...
cobot [to HFh]: The modern individual on the other hand is threatened
by many things against which he is helpless; nuclear accidents, car-
cinogens in food, environmental pollution, war, increasing taxes,
invasion of his privacy by large organizations, nation-wide social or
economic phenomena that may disrupt his way of life.
HFh [to cobot]: I agree with that position, I suppose, but it doesn't
help to explain the lack of a safety net.
cobot [to HFh]: Nevertheless, the position can be defended.
HFh [to cobot]: ...in the *short* term only.
cobot nods to HFh.
```

Cobot's replies were not always appropriate; that they are often appropriate and frequently amusing suffices to intrigue and entertain many users. They seemed to ignore the senseless interactions with Cobot and remembered the "intelligent" ones. Certainly the open-ended nature of LambdaMOO, and the fact that few users have concrete, goal-oriented expectations of the other users or artifacts that they encounter encourages this.

Other documents include scripts of "The Big Lebowski" and "Planet of the Apes"; a long analysis of the JonBenet Ramsey murder case; and articles on alien abduction and paranormal experience. It is simple to instantly swap new documents in and old ones out, limiting Cobot's chat abilities only by the imaginations of his creators. Still, we emphasize our belief that it is advantageous to limit the number of documents at any given time, and for these documents to be reasonably specific, so that well-defined "themes" can emerge in Cobot's conversation. Note that Cobot has no notion of conversational state at this point, so utterances in any given conversation may be drawn from multiple documents, as in the following exchange alternating between the Unabomber Manifesto and "Planet of the Apes":

```
cobot [to cyberat]: There is good reason to believe that many prim-
itive societies subjected people to less pressure than the European
society did, but European society proved far more efficient than any
primitive society [...]
cyberat [to cobot]: Thats what they want you to think!
cobot [to cyberat]: Tell me who and what you really are and where you
come from, and no veterinarian will touch you.
cyberat [to cobot]: Veterinarians!!!
cobot [to cyberat]: The Roman system of urban sanitation was forgot-
ten, so that until rather recent times did the sanitation of European
cities that of Ancient Rome.
```

```
Fawn_Guest [to HFh]: cobot's conversation topics just get weirder and
weirder
cobot [to cyberat]: In spite of every sign I'm an intelligent being
meaning no harm, you continue to hate and fear me.
```

6. Automatic social adaption

Our long-term research goal was to build a software agent that could act in a socially meaningful, useful, interesting, or at least amusing way. Because of the cultural subtleties and ever-changing nature of spaces like LambdaMOO, we believed that such an agent would have to learn a model of such a space on its own, almost certainly from user feedback and interaction.

We had some hope that this was possible. At this point in his short life, Cobot had become a part of his surrounding social fabric, complete with dear friends, bitter enemies, and indifferent observers. More to the point, many of the LambdaMOO citizens felt comfortable interacting in front of him. As such, he was able to build a rich notion of how LambdaMOO citizens interacted with one another. Looking at Figs. 4 and 5, one is struck by just how much it reveals that there is an enormous amount of structure in user interactions. If there is structure, then perhaps it can be learned, and if the structure can be learned, then perhaps it can be leveraged to build an agent that modifies its own behavior based on that structure.

6.1. Imitation

Because Cobot's social statistics provided him with the proper language for modeling the behavior of those around him, he was able to imitate others. In this case, imitation meant automatically inferring how to act in a given situation by simply remembering how others in similar situations had acted. For example, when hugged by a particular user, he knew the proper response because he would have seen how others reacted to that user when she hugged another. Even if he had not seen that user in that situation before, he could see how others responded to similarly treated users and react accordingly. As such, Cobot did not need to know the dictionary-style meaning of "hug" because in this world the semantics of hug are sufficiently captured by an understanding of the proper response to a hug (and perhaps by strategies for elicting hugs from others). That these statistics were personalized turns out to be important, for different responses are appropriate for different users. For example, "zap" is a popular command that causes the invoker to rub his or her feet across the carpet and zap another object with static electricity. For some users, this is considered a friendly greeting, like a playful punch, but for others it indicates extreme displeasure. As such, knowing which kind of person just zapped you represents an important skill.

6.2. Reinforcement learning

Unfortunately, while this particular application of learning proved to be useful and powerful, it was still essentially *reactive*,[7] in that Cobot never initiated interaction

[7] We use the term *reactive* to mean "restricted to responding to human-invoked interaction", rather than the AI term meaning "non-deliberative" [3,4,8].

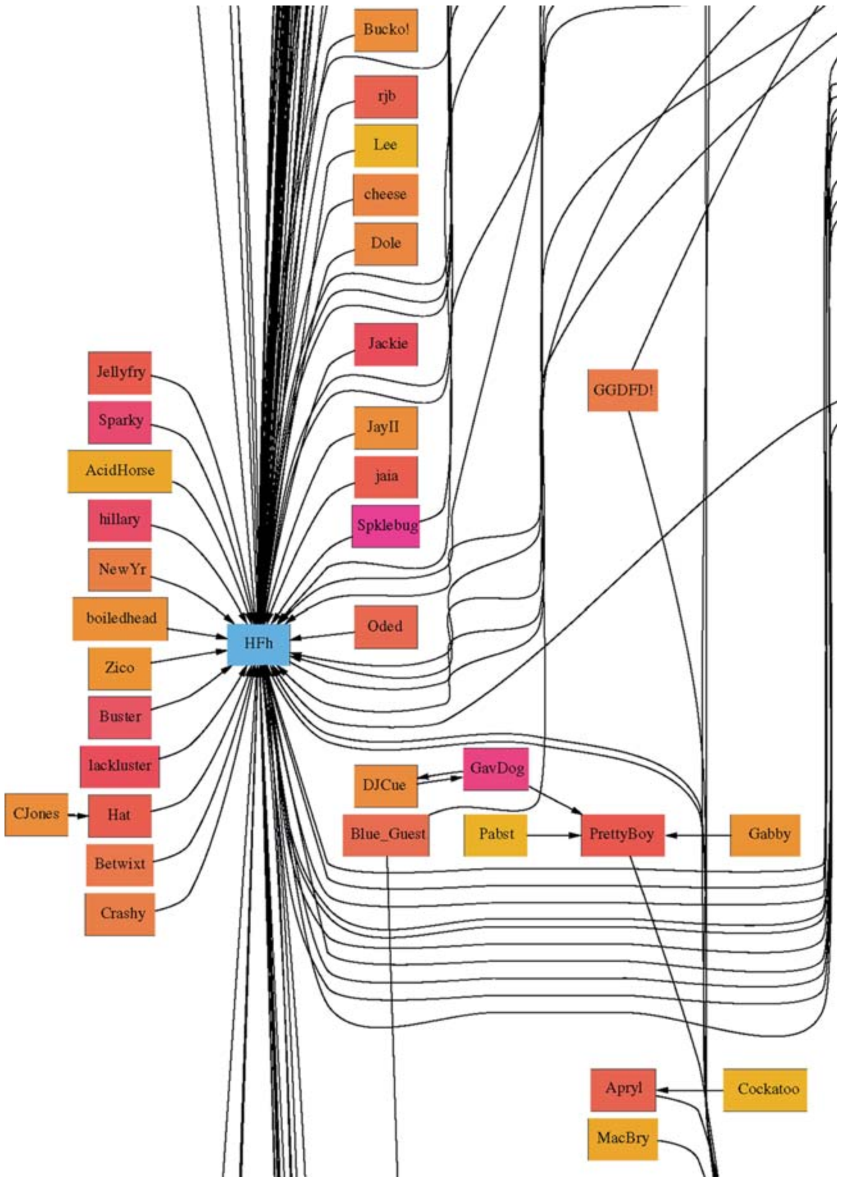

Fig. 4 A partial graph of social interactions. Each node is a user. A node's color represents both that user's amount of interaction and her ratio of speech (e.g., "say") to non-speech actions (e.g., "kick"). Purple indicates normal interactors who mainly use speech; yellow, normal interactors who use non-speech acts; blue, heavy speaking interactors; and green heavy non-speaking interactors. Connections are from a player to his top players by interaction. This graph has many interesting properties. There are no hubs (i.e., nodes with many out edges): the average is only 1.3, with a maximum of 3. By contrast, there are a few key authorities (i.e., nodes with many in edges). Cobot has 288 in edges, with HFh (creator Isbell) trailing behind at 151. After the top ten players, numbers drop quickly. The top eleven players form a densely connected graph. If you remove them from the graph, it collapses. They also tend to be colored differently, exhibiting greater use of non-verbal actions. In order to better show the scope of the social relationships between LambdaMOO users, we reproduce the full map in Figure 5. A full color image is also available at http://www.cc.gatech.edu/ĩsbell/projects/cobot/map.html

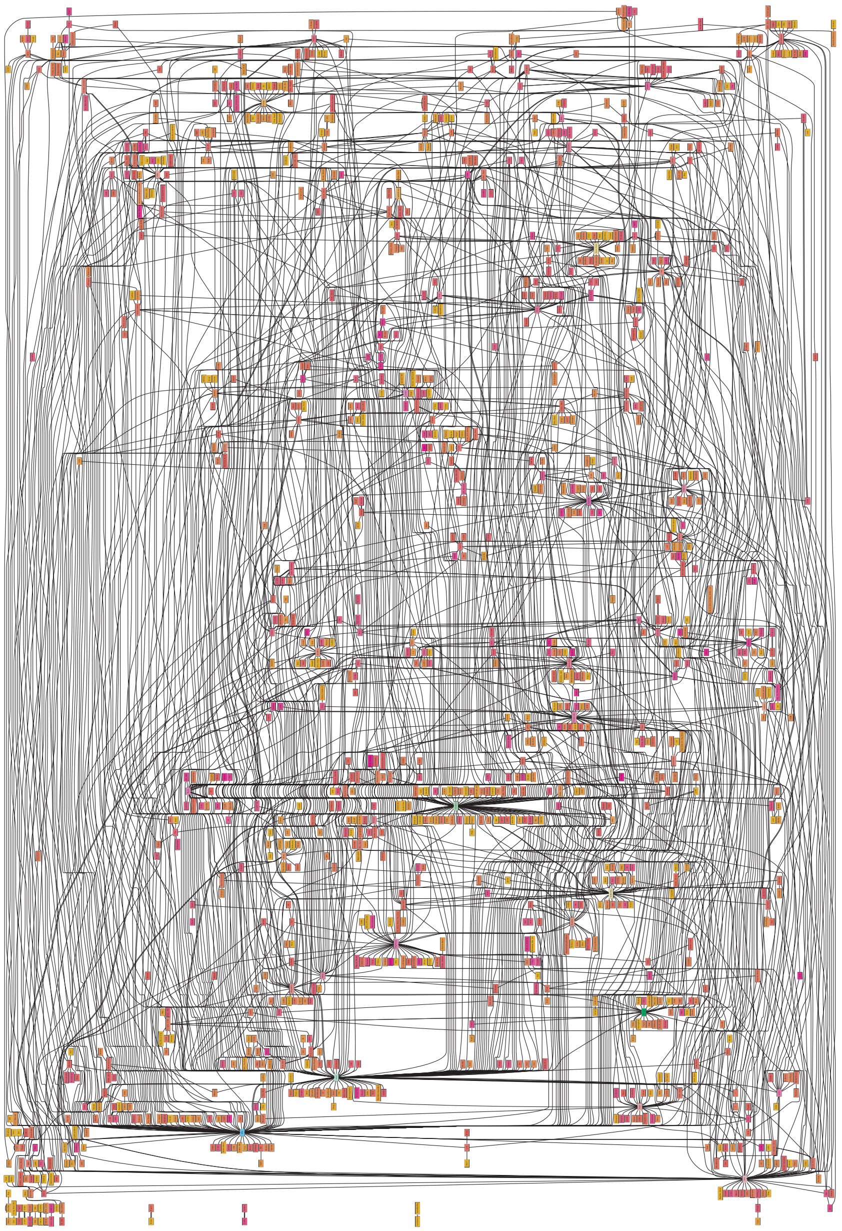

Fig. 5 The full graph of social interactions

with human users, but would only respond to their queries, utterances or actions. In order to be a fully-functioning member of an environment, an agent must be able to interject actions into the space on its own. In other words, Cobot should not always be required to speak only when spoken to.

Reinforcement Learning is one natural approach to building such a proactive agent. In RL, problems of decision-making by agents interacting with uncertain environments are usually modeled as Markov decision processes (MDPs). In the MDP framework, at each time step the agent senses the state of the environment, and chooses and executes an action from the set of actions available to it in that state. The agent's action (and perhaps other uncontrolled external events) cause a stochastic change in the state of the environment. The agent receives a (possibly zero) scalar reward from the environment. The agent's goal is to choose actions so as to maximize the expected sum of rewards over some time horizon. An optimal policy is a mapping from states to actions that achieves the agent's goal.

While most applications of reinforcement learning (RL) to date have been to problems of control, game playing and optimization [28], there has been a recent handful of applications to human-computer interaction. Such applications present a number of interesting challenges to RL methodology (such as data sparsity and inevitable violations of the Markov property). These previous studies focus on systems that encounter human users one at a time, such as spoken dialogue systems [27].

Many RL algorithms have been developed for learning good approximations to an optimal policy from the agent's experience in its environment. At a high level, most algorithms use this experience to learn *value functions* (or *Q-values*) that map state-action pairs to the maximal expected sum of reward that can be achieved starting from that state-action pair. The learned value function is used to choose actions stochastically, so that in each state, actions with higher value are chosen with higher probability. In addition, many RL algorithms use some form of *function approximation* (parametric representations of complex value functions) both to map state-action features to their values and to map states to distributions over actions (i.e., the policy). See [28] for an extensive introduction to RL.

In the next sections, we describe the actions available to Cobot, our choice of state features, and how we dealt with multiple sources of reward. The particular RL algorithm we use is a variant of [29]'s policy gradient algorithm and its details are beyond the scope of this paper (however, see [26] for details). One aspect of our RL algorithm that is relevant to understanding our results is that we use a *linear function approximator* to store our policy. For the purposes of this paper, this means that for each state feature, we maintain a vector of real-valued *weights* indexed by the possible actions. A positive weight for some action means that the feature increases the probability of taking that action, while a negative weight decreases the probability. The weight's magnitude determines the strength of this contribution.

6.3. Challenges

The application of RL (or any machine learning methodology) to an environment such as LambdaMOO presents a number of interesting domain-specific challenges, including:

- *Choice of an appropriate state space.* To learn how to act in a social environment such as LambdaMOO, Cobot must represent the salient features. These should include social information such as which users are present, how experienced they are in LambdaMOO, how frequently they interact with one another, and so on.
- *Multiple reward sources.* Cobot lives in an environment with multiple, often conflicting sources of reward from different human users. How to integrate these sources in a reasonable and satisfying way is a nontrivial empirical question.

- *Inconsistency and drift of user rewards and desires.* Individual users may be inconsistent in the rewards they provide (even when they implicitly have a fixed set of preferences), and their preferences may change over time (for example, due to becoming bored or irritated with an action). Even when their rewards are consistent, there can be great temporal variation in their reward pattern.
- *Variability in user understanding.* There is great variation in users' understanding of Cobot's functionality, and the effects of their rewards and punishments.
- *Data sparsity.* Training data is scarce for many reasons, including user fickleness, and the need to prevent Cobot from generating too much spam in the environment.
- *Irreproducibility of experiments.* As LambdaMOO is a globally distributed community of human users, it is virtually impossible to replicate experiments taking place there.

This is only a sample of the many issues we have had to confront in our Cobot work. We do not have any simple answers to them (nor do we believe that simple answers exist), but here provide a case study of our choices and findings. Our primary findings are:

- *Inappropriateness of average reward.* We found that the average reward that Cobot received over time, the standard measure of success for RL experiments, is an inadequate and perhaps even inappropriate metric of performance in the LambdaMOO domain. Reasons include that user preferences are not stationary, but drift as users become habituated or bored with Cobot's behavior; and the tendency for satisfied users to stop providing Cobot with any feedback, positive or negative. Despite the inadequacy of average reward, we are still able to establish several measures by which Cobot's RL succeeds, discussed below.
- *A small set of dedicated "parents".* While many users provided only a moderate or small amount of RL training (rewards and punishments) to Cobot, a handful of users did invest significant time in training him.
- *Some parents have strong opinions.* While many of the users that trained Cobot did not exhibit clear preferences for any of his actions over the others, some users clearly and consistently rewarded and punished particular actions over the others.
- *Cobot learns matching policies.* For those users who exhibited clear preferences through their rewards and punishments, Cobot successfully learned corresponding policies of behavior.
- *Cobot responds to his dedicated parents.* For those users who invested the most training time in Cobot, the observed distribution of his actions is significantly altered by their presence.
- *Some preferences depend on state.* Although some users for whom we have sufficient data seem to have preferences that do not depend upon the social state features we constructed for the RL, others do in fact appear to change their preferences depending upon prevailing social conditions.

As in any application of reinforcement learning, we must choose the actions, reward function, and states with care, and in a way that is meaningful in the domain.

6.4. Cobot's RL actions

To have any hope of learning to behave in a way interesting to LambdaMOO users, Cobot's actions must "make sense" to users, fit in with the social chat-based environment, and minimize the risk of irritating them. Conversation, word play, and emoting

Table 3 The nine RL actions available to Cobot

Null Action	Choose to remain silent for this time period.
Topic Starters (4)	Introduce a conversational topic. Cobot declares that he wants to discuss sports, that he wants to discuss politics, or he utters a sentence from either the sports section or political section of the Boston Globe, for a total of four distinct actions.
Roll Call (2)	Initiate a "roll call". Roll calls are a common word play routine in LambdaMOO. For example, a discussion may be taking place where someone declares that she is tired of Monica Lewinsky. Another user might then announce, "TIRED OF LEWINSKY ROLL CALL", and each user who feels the same will agree with the roll call. Cobot initiates a roll call by taking a recent utterance, and extracting either a single noun, or a verb phrase. These are treated as two separate RL actions.
Social Commentary	Make a comment describing the current social state of the Living Room, such as "It sure is quiet" or "Everyone here is friendly." These statements are based on Cobot's logged statistics from the recent Living Room activity. While there are several different such utterances possible, they are treated as a single action for RL purposes.
Social Introductions	Introduce two users who have not yet interacted with one another in front of Cobot. Again, this is determined by Cobot's social statistical database. This action is not always available because it depends upon there being two users whom Cobot has not seen interact.

There are nine distinct parameterized actions that are available to Cobot. For readability, we divide them into five similar groups

routines are among the most common activity in LambdaMOO, so we designed a set of nine classes of actions for reinforcement learning along these lines, as detailed in Table 6.4. Many of these actions extract an utterance from the recent conversation in the Living Room, or from a continually changing external source, such as the online version of the Boston Globe. Thus a single action may cause an infinite variety of behavior by Cobot. Cobot also has a special "null" action, where he does nothing, so he may learn to be quiet.

At set time intervals (only every few minutes on average, to minimize the risk of irritating users), Cobot selects an action to perform from this set according to a distribution determined by the Q-values in his current state (described below). Any rewards or punishments received from users up until the next RL action are attributed to the current action, and used to update Cobot's value functions. It is worth noting that from an internal perspective, Cobot has two different categories of action: those actions taken in a proactive manner as a result of the RL, and those actions taken reactively in response to a user's action towards Cobot (his original functionality). However, this distinction is not explicitly announced by Cobot. Some users are certainly aware of the distinction and can easily determine which actions fall into which category, but other users may occasionally reward or punish Cobot in response to a reactive action. Such "erroneous" rewards and punishments act as a source of noise in the training process.

6.5. The RL reward function

Cobot learns to behave directly from the feedback of LambdaMOO users, any of whom can reward or punish him. There are both *explicit* and *implicit* feedback mechanisms.

For the explicit mechanism, we implemented **reward** and **punish** verbs on Cobot that LambdaMOO users can invoke at any time. These verbs give a numerical (positive and negative, respectively) training signal to Cobot that is the basis of the RL. Again, any such reward or punishment is attributed as immediate feedback for the current state and the RL action most recently taken. (This feedback will, of course, be "backed up" to previous states and actions in accordance with the standard RL algorithms.)

There are several standard LambdaMOO verbs that are commonly used among human users to express, sometimes playfully, approval or disapproval. Examples of the former include the verb **hug**, and of the latter the verb **spank**. In the interest of allowing the RL process to integrate naturally with the LambdaMOO environment, we chose to accept a number of such verbs as implicit reward and punishment signals for Cobot (indeed, many LambdaMOO users had been invoking such verbs on Cobot in response to his reactive behavior, long before we implemented his RL component); however, such implicit feedback is numerically weaker than the feedback generated by the explicit mechanisms.

One fundamental design choice is whether to learn a single value function from the feedback of the entire community, or to learn separate value functions for each user based on their feedback alone, and combine the value functions of those users present to determine how to act at each moment. We opted for the latter route for three primary reasons.

First, it was clear that for learning to have any hope of succeeding, there must be a representation of which users are present at any given moment—different users simply have different personalities and preferences. Because of the importance of user identity, we felt that representing which users are present as additional state features would throw away valuable domain information, because the RL would have to discover on its own the primacy of identity. Having separate reward functions for each user is thus a way of asserting the importance of user identity to the learning process.

Second, despite the extremely limited number of training examples available in this domain (empirically < 700 per *month*), learning must be quick and significant. Without a clear sense that their training has some impact on Cobot's behavior, users will quickly lose interest in providing rewards and punishments. A known challenge for RL is the "curse of dimensionality," which refers to the fact that the size of the state space increases exponentially with the number of state features. By avoiding the need to represent the presence or absence of roughly 250 users, we are able to maintain a relatively small state space and therefore speed up learning.

Third, we (correctly) anticipated the fact that certain users would provide an inordinate amount of training to Cobot, and we did not want the overall policy followed by Cobot to be dominated by the preferences of these individuals. By learning separate policies for each user, and then combining these policies among those users present, we can limit the impact any single user can have on Cobot's actions.

6.6. Cobot's RL state features

The decision to maintain and learn separate value functions for each user means that we can maintain separate state spaces as well, in the hopes of simplifying states and thus speeding learning. Cobot can be viewed as running a large number of separate RL processes (one for each user) in parallel, with each process having a different state space (with some common elements). The state space for a user contains a number of features containing statistics about that particular user.

Table 4 State space of Cobot

Social Summary Vector	A vector of four numbers: the rate at which the user is producing events; the rate at which events are being produced that are directed at the user; the percentage of the other users present who are among this user's ten most frequently interacted-with users ("playmates"); and the percentage of the other users present for whom this user is among their top ten playmates.
Mood Vector	A vector measuring the recent use of eight groups of common verbs (for instance, one group includes the verbs *grin* and *smile*). Verbs were grouped according to how well their usage was correlated.
Rates Vector	A vector measuring the rate at which events are being produced by the users present, and also by Cobot.
Current Room	The room where Cobot currently resides.
Roll Call Vector	Indicates whether the currently saved roll call text has been used before by Cobot, whether someone has done a roll call since the last time Cobot did a roll call, and whether there has been a roll call since the last time Cobot grabbed new text.
Bias	Each user has one feature that is always "on"; that is, this bias is always set to a value of 1. Intuitively, it is the feature indicating the user's "presence."

Each user has her own state space and value function; the table thus describes the state space maintained for a generic user

Because LambdaMOO is a social environment, and Cobot is learning to take social actions (roll calls, suggestions of conversational topic, user introductions) we felt that his state features should contain information allowing him to gauge social activity and relationships. Table 4 provides a description of the state features used for RL by Cobot for each user. Even though we have simplified the state space by partitioning by user, the state space for a single user remains sufficiently complex to preclude standard table-based representation of value functions. In particular, each user's state space is effectively infinite, as there are real-valued state features. Thus, linear function approximation is used for each user's policy. Cobot's RL actions are then chosen according to a mixture of the policies of the users present. Again, we refer the reader to [26] for more details on the method by which policies are learned and combined.

6.7. Experimental findings

After implementing the RL action and state spaces, the explicit and implicit reward mechanisms, and the multiple-user learning algorithms described in preceding sections, but prior to publicly fielding RL Cobot in the Living Room, we tested the new software on a separate agent we maintain for such purposes. These tests were performed by the authors and a few friendly users of LambdaMOO in a private room. In addition to tuning the learning algorithm's parameters, we calibrated the frequency with which the agent said something of its own volition. Such calibration is extremely important in a social environment like LambdaMOO due to users' understandably low tolerance for spam (especially spam generated by software agents).

Upon launching the RL functionality publicly in the Living Room, Cobot logged all RL-related data (states visited, actions taken, rewards received from each user, parameters of the value functions, etc.) for 5 months. During this time, 63123 RL actions were taken (in addition, of course, to many more reactive non-RL actions), and 3171 reward and punishment events[8] were received from 254 different users. The findings we now summarize are based on these extensive interactions.

[8] That is, actions which received at least one act of reward or punishment from some user.

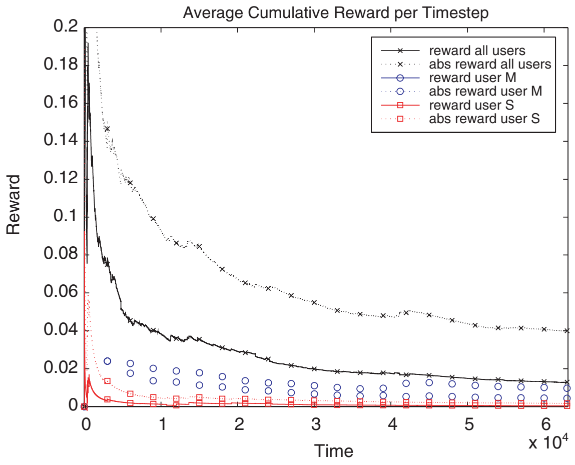

Fig. 6 The average reward received by Cobot over time. The x-axis represents sequential Cobot-initiated RL actions for which it has received at least one reward or punishment event. The y-axis indicates the average cumulative reward received up to each such action. We examine both total reward (in which we sum the positive and negative values received), as well as absolute total reward (in which we ignore sign and simply count the total amount of feedback). We plot these quantities both over all users, and for two particulars users we call Users M and S. In all cases, as time progresses, Cobot is receiving less feedback. This occurs despite the fact that Cobot is learning policies that coincide with the preferences of Users M and S, suggesting that this may not be a useful measure of performance

Inappropriateness of average reward. The most standard and obvious sign of successful RL would be an increase in the average reward received by Cobot over time. Instead, as shown in Fig. 6, the average cumulative reward received by Cobot actually goes *down*; however, rather than indicating that users are becoming more dissatisfied as Cobot learns, the decay in reward reveals some peculiarities of human feedback in such an open-ended environment, as we now discuss.

There are at least two difficulties with average cumulative reward in an environment of human users. The first is that humans are fickle, and their tastes and preferences may drift over time. Indeed, our experiences as users, and with the original reactive functionality of Cobot, suggest that novelty is highly valued in LambdaMOO. Thus a feature that is popular and exciting to users when it is introduced may eventually become an irritant. There are many examples of this phenomenon with various objects that have been introduced into LambdaMOO. In RL terminology, we do not have a fixed, consistent reward function, and thus we are always learning a moving target. While difficult to quantify in such a complex environment, this phenomenon is sufficiently prevalent in LambdaMOO to cast serious doubts on the use of average cumulative reward as the primary measure of performance.

The second and related difficulty is that even when users do maintain relatively fixed preferences over time, they tend to give Cobot less feedback of either type (reward or punishment) as he manages to learn their preferences accurately. Simply put, once Cobot seems to be behaving as they wish, users feel no need to continually provide reward for his "correct" actions or to punish him for the occasional "mistake." This reward pattern is in contrast to typical RL applications, where there is an automated and indefatigable reward source. Strong empirical evidence for this second phenomenon is provided by two users we shall call User M and User S. As we shall establish shortly, these two users were among Cobot's most dedicated trainers, each had strong preferences for certain actions, and Cobot learned to strongly modify his behavior in their presence to match their preferences. Nevertheless, both users tended to provide less frequent feedback to Cobot as the experiment progressed, as shown in Figure 6.

We conclude that there are serious conceptual difficulties with the use of average cumulative reward in such a human-centric application of RL, and that alternative measures must be investigated, which we do below.

A small set of dedicated "parents." Among the 254 users who gave at least one reward or punishment event to Cobot, 218 gave 20 or fewer, while 15 gave 50 or more. Thus, we found that while many users exhibited a passing interest in training Cobot, there was a small group that was willing to invest nontrivial time and effort in teaching Cobot their preferences. Two of the most active users in the RL training were those we have called User M and User S, with 594 and 69 rewards and punishments events given to Cobot, respectively.[9]

Some parents have strong opinions. For the vast majority of users who participated in the RL training of Cobot, the policy learned was quite close to the uniform distribution. Quantification of this statement is somewhat complex, because policies are dependent on state. However, we observed that for most users the learned policy's dependence on state was weak, and the resulting distribution near uniform (though there are interesting and notable exceptions, as we shall see below). This result is perhaps to be expected: most users provided too little feedback for Cobot to detect strong preferences, and may not have been exhibiting strong and consistent preferences in the feedback they did provide.

Still, there was again a small group of users for whom a highly non-uniform policy was learned. In particular, for Users M and S mentioned above, the resulting policies were relatively independent of state,[10] and their entropies were 0.03 and 1.93, respectively. (The entropy of the uniform distribution over the actions is 2.2.) Several other users also exhibited less dramatic but still non-uniform distributions. User M seemed to have a strong preference for roll call actions, with the learned policy selecting these with probability 0.99, while User S preferred social commentary actions, with the learned policy giving them probability 0.38. (Each action in the uniform distribution is given weight 1/9 = 0.11.)

Cobot learns matching policies. In Figs. 7 and 8, we demonstrate that the policies learned by Cobot for Users M and S[11] do in fact reflect the empirical pattern of rewards received over time. Thus, repeated feedback given to Cobot for a non-uniform set of preferences clearly pays off in a corresponding policy.

[9] By "reward event", we simply mean an RL action that received some feedback from the user. Note that the actual absolute numerical reward received may be larger or smaller than one at such time steps, because implicit rewards provide fractional amounts, and the user may also repeatedly reward or punish the action, with the feedback being summed. For example, the total absolute value of rewards and punishments provided by User M was 607.63 over 594 feedback events, while for User S it was 105.93 over 69 feedback events.

[10] We will discuss the methodology by which we determine the relative strength of dependence on state shortly.

[11] Again, we emphasize that because the policies for these users had only a very weak dependence on state, we can simply discuss a fixed policy.

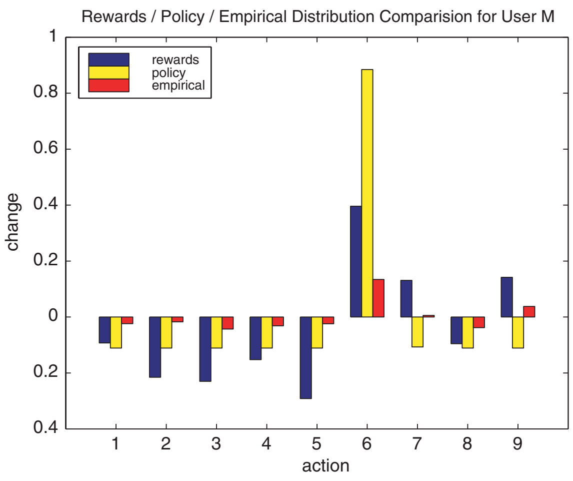

Fig. 7 Rewards received, policy learned, and effect on actions for User M. For each of the nine RL actions, three quantities are represented. The blue bars (left) show the average reward given by User M for the corresponding action. The average reward given by User M across all actions has been subtracted off, so that "positive" bars indicate a relative preference for that action by User M, and "negative" bars indicate a relative objection to that action. The yellow bars (middle) show the policy (probability distribution) learned by Cobot for User M. The probability assigned to each action in the uniform distribution (1/9) has been subtracted off, so again the bars indicate relative preferences and objections of the learned policy. The red bars (right) show, for each action, the empirical frequency with which that action was taken by Cobot when User M was present, minus the empirical frequency with which that action was taken by Cobot over all time steps. Thus, these bars indicate the extent to which the presence of User M biases Cobot's behavior towards M's preferences. We see that (a) the policy learned by Cobot for User M aligns nicely with the preferences expressed by M through her training (comparison of left and middle bars), and (b) Cobot's behavior shifts strongly towards the learned policy for User M whenever she is present (comparison of middle and right bars). The actions are presented here in the same order as in Table 6.4. To go beyond a qualitative visual analysis, we have defined a metric that measures the extent to which two orderings of actions agree. Briefly, given two sequences of actions we count elementwise disagreements. When two elements within a sequence have the same value (and so could be swapped) we choose the ordering that mimimizes that sequence's disagreement with the other. We also extend this to allow for nearly-equal values. The agreement between the action orderings given by the rewards of User M and the policy learned for User M are near-perfect by this measure, as are the orderings given by the policy and the empirical distribution of rewards

Cobot responds to his dedicated parents. The policies learned by Cobot for users can have strong impact on the empirical distribution of actions he actually ends up taking in LambdaMOO. In Figs. 7 and 8, we examine how the distribution of actions taken by Cobot in the presence of these two users differs from the distribution of actions taken in their absence. In both cases, we find that the presence of the user causes a significant shift towards their preferences in the actions taken. In other words, Cobot does his best to "please" these dedicated trainers whenever they arrive in the Living Room, and returns to a more uniform policy upon their departure.

Some preferences depend on state. Finally, we establish that the policies learned by Cobot for users do sometimes depend upon the features Cobot maintains in his state. We use two facts about the RL weights maintained by Cobot to determine which features are relevant for a given user. First, we note that by construction, the RL weights

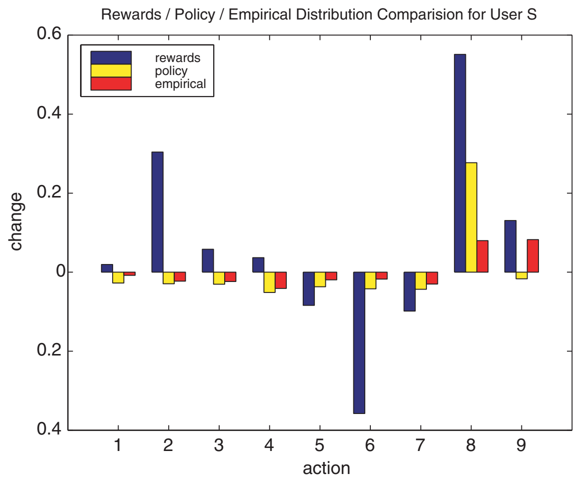

Fig. 8 Rewards received, policy learned, and effect on actions for User S. Same as Figure 7, but for User S. While not as visually compelling as the data for User M, the quantitative agreement between the orderings is again very strong

learned for the bias feature described in Table 4 represent the user's preferences *independent* of state (because this feature is always on whenever the user is present). Second, we note that because we initialized all weights to 0, only features with non-zero weights will contribute to the policy that Cobot uses. Thus, we can determine that a feature is relevant for a user if that feature's weight vector is far from that user's bias feature weight vector, *and* from the all-zero vector.[12] For our purposes, we have used (1) the normalized inner product (the cosine of the angle between two vectors) as a measure of a feature's distance from the bias feature, and (2) a feature's weight vector length to determine if it is away from zero.

As noted above, these measures show that for most users, Cobot learned a policy that is independent of state. For example, User M has a clear preference for roll calls, independent of state. As we consider the other users with non-uniform policies, however, a somewhat different story emerges. As we can see in Table 5, Cobot has learned a policy for other users that clearly depends upon state. Furthermore, personal observation of the users' behaviors by the authors provide anecdotal evidence that Cobot has learned relevant features that "make sense."

7. Privacy and social issues

LambdaMOO has policies regarding researchers, acquiring permissions from users, and quoting material; however, as in any complex culture, there are unofficial and largely unwritten rules that must be learned. Much of the work in introducing Cobot into LambdaMOO involved learning and understanding these rules, and remained

[12] There is another possibility. All weights for a feature could be nearly the same, in which case the feature has no actual effect on which action is chosen, no matter how far away it is from zero. For the users we examined with non-uniform policies this has not turned out to be the case, so we do not discuss it further.

Table 5 Relevant features for users with non-uniform policies

User O	40°	**Roll Call**. Two of the features from the Roll Call Vector have proven relevant for User O. User O appears to especially dislike roll call actions when there have been repeated roll calls and/or Cobot is repeating the same roll call again and again.
	27–33°	**Rates**. Two of the features from the Rates vector also seem to have a similar effect on the rewards given by User O, with the overall rate of events being generating having slightly more relevance than that of the rate of events being generated just by User O.
User B	10°	**Social Summary**. The presence of other users who generally direct events toward User B appears to have some relevance for his reward patterns. If we lower our threshold to consider features below 10 degrees we also notice that some of the other Social Summary features deviate from the bias by about 6 degrees. One might speculate that User B is more likely to ignore Cobot when he is with many friends.
User C	34°	**Roll Call**. User C appears to have strong preferences about Cobot's behavior when a "roll call party" is in progress (that is, everyone is generating funny roll calls one after the other), as suggested by the strong relevance of the Roll Call feature, indicating that other roll calls by other users have happened recently.
User P	22°	**Room**. Unlike most users, User P has followed Cobot to his home, where he is generally alone (but will continue to execute actions at an even higher rate than normal), and has trained him there. User P appears to have different preferences for Cobot under those circumstances than when Cobot is in the Living Room.

Several of our top users had some features that deviated from their bias feature. The second column indicates the number of degrees between the weight vectors for those features and the weight vectors for the bias feature. We have only included features that deviated by more than 10 degrees. For the users above the double line, we have included only features whose weights had a length greater than 0.2. Each of these users had bias weights of length greater than 1. For those below the line, we have included only features with a length greater than 0.1 (these all had bias weights of length much less than 1)

an ongoing concern. We began our education by discussing with colleagues who were LambdaMOO users, and could provide insight into its social dynamics. HFh spent several months as a LambdaMOO user, building genuine friendships, learning about etiquette and privacy, and floating the idea of an agent like Cobot.[13] Only after convincing ourselves that Cobot would not be completely unwelcome did we introduce him into the Living Room.

Privacy. LambdaMOO is open to the public in general. In particular, the Living Room is a *public* space, as opposed to a private room. Possibly intrusive objects are more likely to be tolerated, as long as users do not believe they are being recorded in some way for unfriendly purposes. Furthermore, most users are sophisticated enough to understand the tenuous nature of privacy in such public spaces. On the other hand, most users will *not* simply reveal their real-life identities to casual acquaintances in the MOO, nor do they expect others to reveal such information. Early in the project, several MOOers raised questions around these issues, sometimes in jest, sometimes quite seriously. In keeping with the goal of social acceptance, Cobot is fairly conservative. He notes only events in his presence, and does not share events verbatim. Furthermore, as should be clear from Table 1, a questioner can generally only ask about herself, and not directly about others. Further, all responses are whispered only to the questioner unless the questioner explicitly asks to share the information with the room. Finally, cobot implements an **opt-out** verb. Users who execute the verb are

[13] As of this writing, HFh remains an active member of the community.

ignored for the purposes of cobot's statistical models, but they remain visible to him; that is, he will still respond to them when they choose to interact.

Spam. Being in a public place, Living Room regulars are more likely to tolerate "spammy" objects, but there are clear limits. Because the goal of Cobot is to be a part of his social environment, it is important that he not cross this line.[14] Thus, Cobot's tendency to whisper answers to requests is not motivated just by privacy, but by the desire to be less spammy. Similarly, Cobot generally does not speak unless spoken to, and has built in limits on how often he can execute RL actions. These design decisions are intended to give users a modicum of control over the nature and rate of Cobot's output.

Nevertheless, these precautions sometimes proved inadequate. For example, users expect to have the ability to silence non-human objects (using verbs other than **gag**, which has a number of drawbacks). So, Cobot has a **silence** verb that allows a user to prevent him from speaking out loud (to anyone) for a random amount of time. Cobot may still respond to users, but he will only whisper to his intended target.

Still, it is no more possible to prevent users from using Cobot to generate spam (e.g., by repeatedly querying him aloud) than it is to prevent users from generating spam directly. It is debatable whether such users or Cobot are to blame, but it can be irritating regardless. Cobot's various social statistics are a great incentive to interact in front of and with him perhaps more than one would otherwise, thus raising the overall level of noise in the Living Room.[15]

To combat this, we have implemented more drastic measures. Along with Cobot's "owners," any of a set of identified regulars can direct Cobot to ignore individual players. Cobot will not interact with such a player during these times, except to occasionally inform him that he is being ignored. These policies appear to be the minimum necessary.

8. Conclusions

Perhaps the most important consequence of this work is the observation that during his lifetime on LambdaMOO, Cobot became a real member of his community. As we have seen, he was perhaps the most "popular" resident of the Living Room. Users engaged him in conversation, interacted with him in a variety of ways, took advantage of his statistical services and sometimes even had generally positive things to say about him (see Table 2). On the other hand, his entry was not welcomed by all MOOers. Some complained of a general increase in spam, and others noted that he changed the nature of interaction in the Living Room.[16]

[14] LambdaMOO has developed several mechanisms for dealing with unwanted objects. For example, users can **gag** other objects (that is, to make it so that they see no text generated by a particular object or objects).

[15] There is some evidence that Cobot's presence may have raised the overall amount of interaction in the Living Room.

[16] Some female characters noted that "there are more young males hanging about in the LR these days because [of cobot]." But it is difficult to separate Cobot's influence from other factors, or to even know whether such a change has actually occurred. Interestingly, Foner notes that Julia was explicitly made female to encourage male interaction, but this may not have been necessary.

Second, Cobot represents perhaps the first successful application of reinforcement learning in such a complex human online social environment. LambdaMOO is a challenging domain for RL, as many of the standard assumptions (stationary rewards, Markovian behavior, appropriateness of average reward) are clearly violated. We feel that the results obtained with Cobot are compelling, and offer promise for the application of RL in rather rich and open-ended social settings. The authors continue to investigate this line of research, and have begun applying the lessons and algorithmic techniques to other environments that revolve around human–computer and human–human interaction.

References

1. Barbuceanu, M., & Fox, M.S. (1995). The architecture of an agent building shell. In M. Wooldridge, J. P. Mueller & M. Tambe (Eds), *Intelligent agents II: Agent theories, architectures and languages*, Vol. 1037 of *Lecture Notes in Artificial Intelligence* (pp. 235–250). Springer Verlag.
2. Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM, 37*(7), 122–125.
3. Brooks, R.A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of robotics and automation 1*(1), 14–23.
4. Brooks, R.A. (1991). Intelligence without representation, *Artificial Intelligence, 47*, 139–159.
5. Cherny, L. (1999). *Conversation and Community: Discourse in a Social MUD*, CLFI Publications.
6. Curtis, P. (1997). The lambdaMOO programmer's manual v1.8.0p6, ftp://ftp.research.att.com/-dist/eostrom/MOO/html/ProgrammersManual_toc.html.
7. Dibbell, J. (1999). *My Tiny Life: Crime and Passion in a Virtual World*, Holt, Henry & Company.
8. Etzioni, O. (1993). Intelligence without robots: a reply to brooks, *AI Magazine 14*(4), 7–13.
9. Etzioni, O. (1994). A softbot-based interface to internet, *Communications of the ACM, 37*(7), 72–76.
10. Etzioni, O. (1997). Moving up the information food chain: deploying softbots on the world wide web, *AI Magazine, 18*(2), 11–18.
11. Etzioni, O., Golder, K., & Weld, D. (1994). Tractable closed-world reasoning with updates, In *Proceedings of KR-94*.
12. Finin, T., Labrou, Y., & Mayfield, J. (1997). KQML as an agent communication language. In J. Bradshaw (Ed.) *Software Agents*, MIT Press, Cambridge, pp. 291–316.
13. Foner, L. (1993). What's an agent, anyway? a sociological case study, *Technical report*, MIT Media Lab.
14. Foner, L. (1997). Entertaining agents: a sociological case study, In *Proceedings of the first international conference on autonomous agents*.
15. Genesereth, M. R., & Ketchpel, S. (1994). Software agents, *Communications of the ACM 37*(7), 48–53.
16. Isbell, C. L., Kearns, M., Kormann, D., Singh, S., & Stone, P. (2000). Cobot in LambdaMOO: A Social Statistics Agent, *Proceedings of AAAI-2000*.
17. Isbell, C. L., Shelton, C., Kearns, M., Singh, S., & Stone, P. (2001). A social reinforcement learning agent, *Proceedings of Agents*.
18. Jennings, N., Sycara, K., & Wooldridge, M. (1999). A roadmap of agent research and development, *Autonomous Agents and Multi-Agent Systems, 1*(1), 7–38.
19. Lesperance, Y., Levesque, H., Lin, F., Marcu, D., Reiter, R., & Scherl, R. (1995). Foundations of a logical approach to agent programming, *Intelligent Agents II*, 331–346.
20. Lesser, V. R. (1998). Reflections on the nature of multi-agent coordination and its implications for an agent architecture, *Autonomous Agents and Multi-Agent Systems, 1*(1), 89–112.
21. Maes, P. (1994). Agents that reduce work and information overload, *Communications of the ACM*, **37**(7), 30–40.
22. Mauldin, F. (1994). Chatterbots, TinyMUDs, and the Turing Test: Entering the Loebner Prize Competition, In *Proceedings of the twelfth national conference on artificial intelligence*.
23. McCabe, F. G., & Clark, K. L. (1995). April – agent process interaction language, *Intelligent Agents: theories, architectures, and languages*.
24. Mitchell, T., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant, *Communications of the ACM, 37*(7), 80–91.

25. Nwana, H., Ndumu, D., Lee, L., & Collis, J. (1999). Zeus: A tool-kit for building distributed multi-agent systems, *Applied Artifical Intelligence Journal, 13*(1), 129–186.
26. Shelton, C. R. (2000). Balancing multiple sources of reward in reinforcement learning, *Advances in Neural Information Processing Systems*, 1082–1088.
27. Singh, S., Kearns, M., Littman, D., & Walker, M. (2000). Empirical evaluation of a reinforcement learning dialogue system, In *Proceedings of the seventeenth national conference on Artificial intelligence (AAAI)* pp. 645–651.
28. Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.
29. Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation., *In Neural Information Processing Systems-1999*.
30. Sycara, K., Pannu, A., Williamson, M., & Zeng, D. (1996). Distributed intelligent agents, *IEE Expert, 11*(6), 36–46.