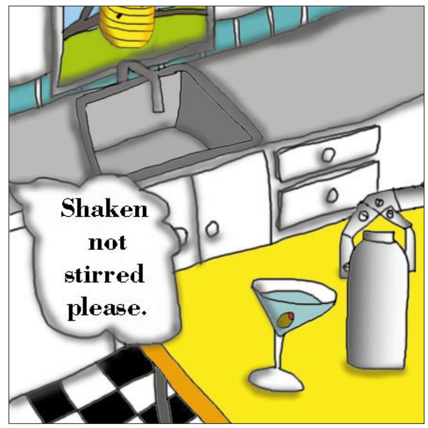
Features Editor: Crystal R. Chweh, cchweh@computer.org

Distributed agents

Agents that move for things that think

Dennis Taylor, dtaylor@computer.org



Want your home bar to mix and serve you a dry martini ("shaken, not stirred, please") as you sit in your recliner, watching "Buffy the Vampire Slayer"? Of course you do. If the researchers at the MIT Media Lab have their way, your wish could be granted sooner than you think. As part of the Things That Think project (www.media.mit.edu/ttt), which attempts to add computing power to everyday objects, they've developed Hive, a distributed-agent platform for controlling appliances.

Hive ecology

Nelson Minar and his colleagues designed Hive (www.hivecell.net) to be an "ecology of distributed agents." That is, applications are formed out of the actions and interactions of agents spread over a network. These agents and other Hive components form a decentralized system. The Media Center researchers believe that such decentralization is necessary to let the system adapt and grow.

Hive is basically a set of Java libraries. Its three main components are *cells*, *shadows*, and *agents*.

Cells are Java programs that act like servers. They are the network nodes, providing a home for agents and shadows. They enable agents to communicate and let agents access devices connected to them (say, a microphone for voice input and a drink mixer for output). Because of memory and computational requirements, cells run on desktop computers. However, they eventually could reside on other devices attached to the network.

Shadows are software interfaces (APIs) to the devices that are attached to a cell. A shadow provides both access and security; agents must access a device through its shadow. Shadows, like cells, are static.

Agents are autonomous, mobile code comprising "a Java object, an execution thread, a remote interface for network communication, and a self-description."[1] They can act as intermediaries for shadows, provide software services, or manage other agents. Although each agent resides in a particular cell, agents can travel to other cells to conduct their business by interacting with other agents. For example, a microphone agent could deliver microphone service to a cell containing an agent that handles natural language processing.

An agent learns about another agent by accessing that agent's *syntactic* and *semantic descriptions*. The syntactic description is that agent's Java type. The semantic description adds information such as the agent's location, owner, and capabilities. This description is based on an ontology that uses the World Wide Web Consortium's Resource Description Framework. (However, according to Raffi Krikorian, who has taken over the Hive project now that Minor has gone off to start his own company, they've been working on a version that is based on XML instead of RDF.) Agents can change their descriptions at runtime, thereby providing flexibility.

One specific type of agent is the graphical user interface (see Figure 1). The GUI lets the user both monitor and control the Hive network. The display uses icons to represent the agents and arrows to represent agent communication paths. Users can change the network and applications by adding or deleting icons and redrawing paths.

If you think Hive sounds like Sun Microsystem's Jini (www.sun.com/jini), you're right. Both are Java-based systems for creating flexible networks of devices. However, Jini does not separate the functionality of agents and shadows. Hive's developers believe that such separation "gives Hive a useful abstraction barrier between local, trusted code and networked, untrusted code."[1] Also, Hive is more location-dependent; Jini does not contain equivalents to cells. Minar and his colleagues think that this makes Hive more scalable. In addition, Jini provides only single-hop mobility; Hive agents have multihop mobility.

Cooking with smarts

Matthew Gray and his colleagues at the Media Lab have used Hive to construct a networked kitchen.[2] (See the sidebar for some other Hive applications, including Counter Intelligence, a different intelligent-kitchen project.) Their goal is a kitchen where the refrigerator and storage areas know their contents and know when something is needed. The appliances would order groceries and supplies and help the cook plan and prepare meals.

To show their project's potential, they constructed a demonstration system and implemented a recipe scheduler for making peanut brittle. The system employed three types of Hive agents. A recipe agent oversaw the recipe's execution and controlled manager agents, which managed the ingredients, the appliances, and user interaction. Device agents managed the appliances, which included tag readers for keeping an inventory of the groceries and supplies, a microwave oven, a scale, a speech output system, and a visual display.

The scheduler determined the best sequence in which to prepare the peanut brittle, given a recipe. The recipe included the steps (recipe steps don't always have to follow a particular order), their interdependencies, and the necessary ingredients and supplies. Because Gray and his colleagues were only demonstrating one recipe, they did not integrate the recipe scheduler with the rest of the demonstration system.

According to Gray, using Hive made their system more flexible.[2] For example, a

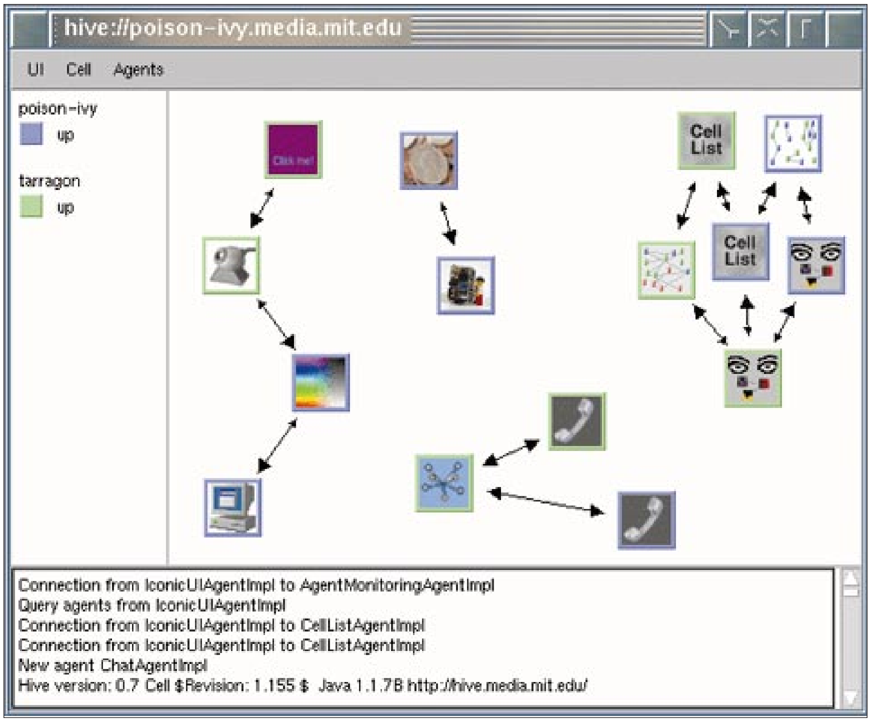


Figure 1. Hive's GUI. Icons represent agents, and arrows represent agent communication paths. Users can change the network and applications by adding or deleting icons and redrawing paths. For a detailed explanation of this image, see www.hivecell.net/screenshot.html. (Image courtesy of MIT Media Lab.)

Some Hive applications

- *Automatic diary*: As a user moves through a building, radio beacons transmit the user's location to his or her wearable computer. Information entered into the computer is automatically "stamped" with its date, time, and location.[1]
- *Counter Intelligence* (www.media.mit.edu/ci): This is a five-year program to investigate future kitchen technologies. It incorporates such projects as Mr. Java, an intelligent coffee machine, Scents Sense, which combines a digital nose with an oven, and Visiphone, which combines graphics with telephone conversations.
- *Honey, I Shrunk the CDs* (www.media.mit.edu/pia/Research/CDs/index.html): Poker chips contain a song title and a radio frequency ID tag. When a user chooses one of the chips and drops it on a table with an RF tag reader, an MP3-based jukebox plays that song.[2,3]
- *Impulse* (agents.www.media.mit.edu/groups/agents/projects/impulse): Agents running on wireless mobile devices conduct negotiations on behalf of buyers and sellers.
- *Personal Ambient Displays* (tangible.media.mit.edu/projects/Personal_Ambient_Disp/Personal_Ambient_Disp.html): These small devices subtly transmit information by getting cold or hot, moving, vibrating, or changing shape. They can be connected through Hive to activate each other or to respond to input from other sources.[2]
- *Personal locator*: A computer keeps track of the location of someone with a wearable computer. If the user so desires, the computer can let other people know his or her location.[1]
- *Personal theme music*: When the user enters a room, his or her wearable sends a signal to a computer in that room, which then plays a song that user has chosen.[1]
- *Remembrance Agents* (rhodes.www.media.mit.edu/people/rhodes/RA): These programs send the user of a wearable computer information relevant to that user's location. A possible application could be an automated tour guide for a museum.[1]

References

1. B. Rhodes, N. Minar, and J. Weaver, "Wearable Computing Meets Ubiquitous Computing: Reaping the Best of Both Worlds," *Proc. ISWC '99: Third Int'l Symp. Wearable Computers*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1999, pp. 141–149; rhodes.www.media.mit.edu/people/rhodes/Papers/wearhive.html (current Mar. 2000).

2. N. Minar et al., "Hive: Distributed Agents for Networking Things," *Proc. ASA/MA '99: First Int'l Symp. Agent Systems and Applications and Third Int'l Symp. Mobile Agents*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1999, pp. 118–129; hive.www.media.mit.edu/projects/hive/hive-asama.html (current Mar. 2000).

3. M.K. Gray, *Infrastructure for an Intelligent Kitchen*, master's thesis, Massachusetts Inst. of Technology, School of Architecture and Planning, Cambridge, Mass., 1999; www.hivecell.net/mkgray-thesis/html (current Mar. 2000).

polyphonic tag reader (which can read several tags simultaneously) that they originally planned to use was unavailable, so they substituted several other tag readers for it without changing any code. He also believes that the system implementation was simpler with Hive than without it, and that Hive's structure allows parts of this system to be used with others.

What's next for Hive?

Krekorian sees Hive as the basis for a new way of creating large computer systems. These systems would go beyond traditional software reuse, exploiting existing resources over a network to create new applications. For example, he hypothesizes two "Hive-ified" devices: a child's interactive stuffed toy and a home security system. Without restarting the system, a user could add another Hive agent that links the two devices, creating an application that would use the toy to tell the child that his mom was home from work when the security system sensed her car going into the garage.

Hive's developers envision an Internet buzzing with Hive agents. However, before that happens, they need to deal with several issues. For example, they need to finish Hive's mobility layer. They also need to increase Hive's stability so that agents can run persistently. In addition, they need to improve security—specifically, protecting hosts and agents from other hosts and agents. Finally, they'd like some help. Toward that end, they've released open-source code for Hive (www.hivecell.net/download.html). Personally, I'm looking forward to the day when my kitchen whips me up a chocolate malt and a homemade pizza while I'm playing pinochle.

References

1. N. Minar et al., "Hive: Distributed Agents for Networking Things," *Proc. ASA/MA '99: First Int'l Symp. Agent Systems and Applications and Third Int'l Symp. Mobile Agents*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1999, pp. 118–129; hive.www.media.mit.edu/projects/hive/hive-asama.html (current Mar. 2000).

2. M.K. Gray, *Infrastructure for an Intelligent Kitchen*, master's thesis, Massachusetts Inst. of Technology, School of Architecture and Planning, Cambridge, Mass., 1999; www.hivecell.net/mkgray-thesis/html (current Mar. 2000).

Honeybees to UAVs: doing the waggle walk

Dick Price, dprice@computer.org

Ongoing research at the Australian National University (ANU) in Canberra shows that honeybees possess a sort of visual odometer to gauge how far they have flown, which helps them direct their hive mates to food sources. Aside from its importance for the study of insect behavior, this finding might well have an important impact on the development of unmanned flying vehicles.

According to Mandyam Srinivasan, the bioengineer leading this investigation, honeybees perform varying dances to communicate with other bees about the distance and direction to food sources. For the type of honeybee used in this work, a round dance indicates that the food is within about 50 meters (see Figure 1). Beyond that distance, the bees switch to a waggle dance, with the dance lasting proportionally longer for more distant food sources and shifting axially according to the sun's position in the sky to communicate direction.

Animal behavioralists once thought that bees estimate distances by gauging the energy they expend traveling back and forth—the longer the bee flew, the longer its dance. But tail or headwinds would make such estimates highly unreliable, so Srinivasan and his colleagues at ANU's Centre for Visual Science (cvs.anu.edu.au) have found that a very different mechanism is at work. As reported recently in *Science* (4 February 2000, pp. 851–853), they argue instead that the bees gauge distance traveled by the amount of image motion they experience during the trip. Because new bees will tend to follow the same route as the experienced forager, this sort of visual odometry would prove to be quite reliable regardless of wind conditions.

Fooling Mother Nature

To test their theory, Srinivasan's group trained marked honeybees to forage a feeder containing sugar solution positioned in a narrow, well-lit wooden tunnel (see Figure 2). When the tunnel's surface was covered with a visually intense random texture, the bees would be fooled into believing they had flown much further than they actually had.

Figure 1. Honeybee (*Apis mellifera lingustica* Spinola) entering the test tunnel. Canberra scientists have been studying honeybee behavior for almost 15 years using hives on the ANU campus. (Photograph by Jeff Wilson, Research School of Biological Sciences, ANU.)

Figure 2. Closed at one end so the bees have only one point of entry, the tunnel is covered with black insect-screen cloth to permit observation and to give the bees a view of the sky. The test tunnel measures 6.4 meters long, 11 cm wide, and 20 cm high. (Photograph by Teresa Belcher, Research School of Biological Sciences, ANU.)

In one the series of related experiments reported in *Science*, for instance, their 441-ms-long waggle dance indicated a flight of 184 meters outdoors, when they had actually flown just 12 meters—six meters outdoors and six meters down the tunnel. Srinivasan surmises that the tunnel forced the bees to fly closer to nearby objects than they would ordinarily do in flying outdoors. "And, as you fly along a straight line," he says, "images that are very close to you provide a very great image velocity, whereas objects that are further away don't appear to move nearly so much." By flying so closely to the patterned surface, they received visual motion cues equivalent to a much longer trip. Furthermore, when the tunnel's walls were covered with horizontal lines running parallel to their line of light, thus creating negligible image motion cues, the bees more accurately indicated the distance travelled with their round dance.

From bees to UAVs

Srinivasan's group is working with funding from DARPA's Controlled Biological and Biomimetic Systems group under Alan Rudolf to apply these and related results to their work with unmanned autonomous vehicles. Currently, in trying to see how robust this visual odometry will be for robotic vehicles, they are encouraged by initial results using a land-based robot. "Certainly, it's immune to headwinds or tailwinds, which you've got to worry about with other methods."

Earlier work by Srinivasan showed that bees tend to fly down the middle of the tunnel by balancing the optical flow from the two sides, rather than by using stereo mechanisms traditionally used in robotics. "They use very low-level image motion computing mechanisms that are a easier algorithmically to do in real time than, for example, using stereovision, which involves solving the correspondence problem," he says. "There's no object recognition as such going on. There are just very low-level image-motion cues being picked up." With these findings, ANU researchers have developed robot navigation systems that require much less computational power—just that of a standard laptop—than would otherwise be needed.

Srinivasan and his colleagues are also working on a panoramic vision system that emulates insect vision for use in helicopters. Rather than attempting to develop an insect-like compound eye, they use a specially shaped reflector that works with a single camera to capture a panoramic view for stabilizing the vehicle's flight and improving landings and takeoffs.

Insect landing strategies is another area of interest at ANU. Traditional experiments involve insects landing head on: By measuring the rate at which the surface expands, the insect would gauge its deceleration and extend its legs appropriately. A more interesting problem is the grazing landing on a horizontal surface, which is a more typical landing strategy for a UAV.

"Under those conditions, the looming cues are not very strong, as the insect is basically moving almost parallel to the surface," he says. The animal mainly experiences translational flow, which turns out to again involve a fairly simple algorithm. "They keep the angular velocity of the image in the eye constant as they approach the ground, so the lower they are, the slower they're flying automatically." Consequently, when the insect finally touches down, it is flying with zero forward velocity—without having to do complex ranging.

"The trajectory then should be an exponential time curve, which is exactly what you get," he says. "We're currently trying to put that into a flying vehicle to see how well it works."

To biology and back

Trained originally as an electrical engineer, Srinivasan was intrigued by the capabilities of insects' relatively small brains while doing his PhD at Yale. "They don't have a lot of cognition, the way we do, but in the low-level aspects of vision, they have everything it takes. If you watch a fly land on a teacup, for instance, you find that it does a very nice job."

The knowledge flow in Srinivasan's work goes both ways: findings from insect research—and from work with wallabies, birds, crabs, and humans that others are undertaking at his institute—feed into robotics work, but also flow back in the other direction. "If the robot doesn't work the way we think it should, that puts us back on the drawing board as far as the biology is concerned as well."

IEEE Intelligent SYSTEMS & their applications

Editor-in-Chief
Daniel E. O'Leary
University of Southern California
oleary@rcf.usc.edu

Associate Editor-in-Chief
Stephen E. Cross
Carnegie Mellon University
sc@sei.cmu.edu

EDITORIAL BOARD

Richard Benjamins
University of Amsterdam
Douglas Dyer
DARPA
Richard J. Doyle
Jet Propulsion Lab
Dieter A. Fensel
Vrije Universiteit Amsterdam
Yolanda Gil
University of Southern California
C. Lee Giles
NEC Research Institute
Kristian I. Hammond
University of Chicago
James V. Hansen
Brigham Young University
Marti A. Hearst
University of California, Berkeley
James A. Hendler
DARPA
Haym Hirsh
Rutgers University
Se June Hong
IBM T.J. Watson Research Center
Eric Horvitz
Microsoft Research
Craig Knoblock
University of Southern California
Robert Laddaga
DARPA
Robert Plant
University of Miami
Paul S. Rosenbloom
University of Southern California
Howard E. Shrobe
Massachusetts Institute of Technology
William R. Swartout
University of Southern California
Mark L. Swinson
DARPA
Milind Tambe
University of Southern California
Austin Tate
University of Edinburgh
David L. Waltz
NEC Research Institute
Ronald Yager
Iona College

CS MAGAZINE OPERATIONS COMMITTEE

Sorel Reisman (chair), William Everett (vice chair), James H. Aylor, Jean Bacon, Thomas J. (Tim) Bergin, Wushow Chou, George V. Cybenko, William I. Grosky, Steve McConnell, Daniel E. O'Leary, Ken Sakamura, Munindar P. Singh, James J. Thomas, Yervant Zorian

CS PUBLICATIONS BOARD

Sallie Sheppard (vice president), Sorel Reisman (MOC chair), Rangachar Kasturi (TOC chair), Jon Butler (POC chair), Angela Burgess (publisher), Laurel Kaleda (IEEE representative), Jake Aggarwal, Laxmi Bhuyan, Lori Clarke, Alberto del Bimbo, Mike Liu, Mike Williams (secretary), Zhiwei Xu

Voice recognition

Voice recognition technology is starting to add up

Shani Murray, smurray@computer.org

Henry "Buddy" Gray, a mathematics and statistics professor at Southern Methodist University (hgray@mail.smu.edu), has developed MathTalk—a software program that turns words into equations. Gray originally created his voice recognition program to save keystrokes, but he soon realized that his software could help not only other professional mathematicians, but also disabled or blind people.

"Instead of you learning the computer's language, it learns yours," explains Gray, who now works in collaboration with Metroplex Voice Computing of Arlington, Texas (www.mathtalk.com). MVC offers several different products, all based on Gray's initial program. MathTalk/Scientific Notebook uses DragonDictate and Scientific Notebook and can be used for all levels of math. It recognizes over 2,000 voiced math commands, including formulas such as the quadratic formula, 3D wave equation, and Bessel function. "You say 'quadratic formula,' and you save 30 to 40 keystrokes," says Gray. He adds that the voice recognition is very good because it's directed toward phrase commands, decreasing the use of smaller words such as "in" and "to," which are easily misunderstood. The program also has graphing capabilities.

While MT/SN helps paraplegics and quadriplegics, Gray's MathBrailleTalk aids instructors of visually impaired students by translating mathematical formulas and embossing them into Braille. Gray is also working on MathTalk for the Visually Impaired. MT/VI users type in the mathematical equations, the program reads them back, and the users can then correct the program with commands such as "erase" or "wrong word." In addition, MT/VI will assist both students and teachers with its ability to print the equations in either Braille or regular text. According to Jason Balusek, a blind student earning his master's degree in mathematics at Stephen F. Austin State University, "This will open a lot of doors in math for blind people."

ArithmeticTalk, geared towards grade-school students, will also strengthen the prospects for disabled students. Says Gray, "I realized that most disabled children don't even get to algebra, which cuts out their opportunities." Using MS Word97, the program better introduces these children to mathematics. The students can tell it to add, subtract, multiply, and divide, and the program will then show them their work.

Yet another potential program stemming from Gray's initial software is aimed at those with speech impairments. Kathy Whipple (kathy_whipple@baylor.edu), chair of the Department of Communication Sciences and Disorders at Baylor University, has applied for a grant and hopes to use Gray's technology for people with good cognitive skills but poor articulation—for those who have suffered a stroke or have a tumor, for example. An inflection in a person's voice or a series of noises could indicate various needs and trigger certain icons representing anything from food to entertainment. Whipple feels such a voice-activated communication system—as opposed to the often cumbersome and awkward point-and-click models—"is desperately needed." She adds, "The technology is clearly there. We're not far away."

Web agents

Shopbots: help or hindrance?

David Clark, dwclark@earthlink.net

Generally speaking, a *bot* is a software component that does the work of a human or another bot (or agent). In terms of shopping, this can mean a Web portal (*shopbot*) that can display product price and availability from multiple online vendors. The hype surrounding shopbots asserts that they can help the online shopper make educated purchases. But often shopbots have only delivered prices, without other pertinent information, such as contrasting features and detailed specifications—leaving the shopper informed, but not necessarily educated.

Market research has indicated that Web buyers like to comparison shop, instead of relying on high-profile or random sites. Consequently, as e-commerce flourishes, so does the demand for comprehensive, useful shopbots.

To the future and beyond

The first shopbots Web buyers used were relatively simple, in most cases they spidered vendor's sites for basic (mostly price) information. The information they discovered was limited and the process could tie up bandwidth and servers, causing resentment among vendors (see "Shopbots Become Agents for Business Change," *Computer*, Feb. 2000). Today, instead of spidering, many shopbot developers either arrange with cooperative vendors to access their database schema (in return for the likelihood of increased visitors and payments), or shopbot sites use sophisticated parsing techniques and scripts to train their bots to gather more robust information from the HTML on vendors' product pages.

Neither approach delivers on the ultimate promise of shopbots. One day, according to Steve White of IBM's Institute for Advanced Commerce, shopbots will automatically engage in complex transactions on behalf of their human owners—or on other agents.

Attempting to deliver these shopbots of the future, industry and university research groups are scrambling to develop bot technology that is effective and autonomous, without introducing economic chaos. As one commercial example of this kind of research, take Rob Guttman and his colleagues' work at MIT. Their work on normalization retrieval forms the technical basis for Frictionless Commerce, a shopbot engine used by high-traffic Web sites such as Lycos. According to Guttman, the company's CTO, the commercial version of this ontology-based shopbot offers extended function over most current shopbots, giving shoppers the ability to make apples-to-apples comparisons.

Data collection is the key

"We're able to parameterize product details to a rather low level," says Guttman. The company employs teams of information architects who develop ontologies for each of the product domains served by the companies who use the Frictionless bot engine. Their job is to define the common feature sets and the semantics used by manufacturers, vendors, and buyers of a particular product category—digital cameras, for example. This compiled body of terms and its semantic relationships are then used to build templates that spiders fill in with data collected from the Web to create a rich product knowledge base.

On the front end, Frictionless uses a value-comparison engine—which uses technology based on knowledge-representation theory research—that analyzes the product knowledge base according to a shopper's input. In practice, the Frictionless bot leads shoppers through a series of questions, which helps them focus on priorities, ultimately building a list of priorities. After personal priorities are established, the shopbot displays a Web page that lists a selection of similar products (or the same product sold by different vendors), sorted according to the buyer's established priorities. In the works now is a sell-bot that can could interact with shopbots to negotiate price and other features based on customer preferences.

Ontology means autonomy

Meanwhile, from an academic angle, Katia Sycara's research group at Carnegie Mellon University also takes the ontological approach to give shopbots (and other types of agents) the ability to operate autonomously.

Her group vision is a future for agent interactivity that includes what she calls *middle agents* that act as agent registries. In this model, a company or individual puts a shopbot on the network, and the bot registers its services or submits a request with the one or more existing middle agents it knows how to find. The middle agents would then know how to either advertise that bot's services or how to locate a bot that can supply the requested service or product.

Sycara compares the infrastructure that would support middle agents as similar in concept to Sun's "spontaneous networking" Jini, which provides protocols and services for resource lookup in LAN environments. However, not only would the infrastructure for her middle-agent model need to work on a WAN, but it would require a more powerful and syntactically forgiving matching mechanism than simple lookup tables provide.

"That's where it gets interesting," says Sycara. The agents need to be able to describe the notion or concept of their capability; they need to be able to match services with service requests—possibly over a broad range of terms. For that, her group has developed Larks (Language for Advertisement and Request for Knowledge Sharing), an agent capability description language. Agents use this language to match service-requesting agents with service-providing agents. Requests are filled when the provider's advertisement sufficiently resembles the requested service's description.

Larks uses information retrieval and AI techniques—along with distributed object programming—to discover syntactical and semantic similarity among agent capability descriptions. In a typical configuration, a middle agent's Larks-based matching engine would use five different filters, one each for context matching, word-frequency profile comparison, similarity matching, signature matching, and constraint matching. Users could configure their bots by changing filter parameters to reach satisfactory performance levels. In other words, you could set your shopbot to find product information that was neither too general nor too specific.

Bot behavior

As bots become more efficient in terms of autonomously finding information they'll stimulate competition, giving buyers more choices. But, as Sycara speculates, if prices drop below profitability, vendors will be forced out of business, or perhaps forced into forming cartels and price-fixing schemes to stabilize prices.

Speculation about what effect the introduction of billions of shopbots will have on our already rapidly changing economic models is the subject of Jeff Kephart's work as manager of IBM's agents and emergent phenomena group (part of IBM's Institute for Advanced Commerce, www.research.ibm.com/infoecon). Kephart's team is developing prototypes of an agent economy and its supporting infrastructure. The programmers built various shopbot *actors* (buyers, sellers, and several types of intermediaries), each of which you can modify by plugging in pricing-algorithm and negotiation-protocol components. The team's first prototype economy comprised protobots that modeled the book-selling market.

"It's interesting to see how a bunch of really simple individual agents, each created by a different human programmer, which have a little bit of infrastructure and the usual magic of economic incentives in common, can produce a market or economy that actually works," says Kephart.

However, in some experiments with simple bot prototypes, bots create cyclical price wars. In these cases, the team observes that prices are driven down over a period of time, revert to a high value, and then start decreasing again. Price wars happen when seller agents have a lot of information about buyers and also have current information about other seller's prices. "You can see that price wars occur because the bots aren't taking into account the likelihood that other sellers are going to retaliate," explains Kephart. So the obvious solution, he says, is to give agents foresight. "This needs to happen far short of the Turing challenge—it has to be a relatively simple solution."

The need for a simple solution led the research team to use a technique called q-learning, a form of reinforcement learning theory. Using q-learning algorithms, the bot can be taught that the ultimate goal is to optimize the future discounted profit as it's determined over a series of steps, not optimize the immediate profit. This "learning" is accomplished because, theoretically, at each established time interval, the bot receives reinforcement information that motivates it to evolve in the desired behavior.

But there's a hitch. Q-learning theory was developed for agents that are interacting in a static environment, says Kephart, and if these other bots are continuously and simultaneously learning and updating, they're forming distinctly nonstationary environments. That begs the theoretical question: Will the bots' price-setting behaviors converge, modeling flatter price fluctuations or will it result in a whole lot of tail chasing?

Kephart and his team are working on the answers to this and other questions. They strongly believe that we need to understand how autonomous interactive bots will behave by observing them in the laboratory before we turn them loose on the world economy, and common sense would agree with them.

Ultimately, the current bot researchers and developers aren't just trying to model the world economy, they're trying to automate it. Kephart echoes many who say that information is what drives the world economy. Current prosperity is often attributed to the rise of the Internet and e-commerce, but it's a result of information flowing better. Information that's not in the right place at the right time is inefficient. What bots will do for us, say these researchers, is eliminate the inefficiencies.

Next time we shop for a Pokémon or an MP3 player on the Internet, we'll have to remember we're really shopping for information. And increasingly, the value we pay for in our shopping experience will be the *information* delivered—quickly and transparently—by shopbots.