A Multiagent Treatment of Agenthood*

Michael N. Huhns
Department of Electrical & Computer Engineering
University of South Carolina
Columbia, SC 29208, USA

huhns@sc.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7534, USA

singh@ncsu.edu

Abstract

There have been numerous attempts to provide a standardized definition of a computational agent, but little consensus has emerged. We propose a simple test for agenthood that can be applied to a putative computational agent. Roughly, this test seeks to capture the intuition that an agent is an entity that can function as part of a multiagent system. The test depends on the observed behavior of the supposed agent, and not on the internals of it. We apply the test to some well-known kinds of systems (supposed) agents, and discuss the results. We present a formulation of the test and some variants with a semantics based on sociability. Our treatment of agenthood can thus serve as a basis for a methodology for evaluating putative agents and agent toolkits.

* A previous version of some of the ideas developed herein appears as an instance of the column *Agents on the Web* in *IEEE Internet Computing*, volume 1, number 5, pages 78–79, September-October 1997.

1 Introduction

For reasons that are well-known, computational agents are drawing a lot of attention from researchers and practitioners [Jennings & Wooldridge, 1997; Huhns & Singh, 1997]. Each project or result inevitably includes—implicitly or explicitly—a definition of an agent. Definitions are important, because they enable people to communicate and understand each other. Precise definitions are even better, because they enable researchers to compare systems and evaluate developments and advancements. This is especially important for a young field, such as multiagent systems, where there is active research underway at many research centers around the world. Unfortunately, definitions of agents abound, and almost every research group has its own. Several properties and definitions of computational agents have been studied in the literature [Petrie, 1996; Franklin & Graesser, 1997].

Definitions Good definitions are not mere stipulations. They seek to capture some intuitive, pretheoretic notions. In the broad sense, definitions represent unstated consensus. There is at times a lot of interest in such definitions, but it usually dissipates in inconclusive results. For example, the definition of physics and chemistry were debated in the 19th and early 20th century, but are not any more. Fortunately, the resolution of the broader concepts is rarely practically important unless we are in the midst of a paradigm shift. For example, there are no good definitions of "life" and "disease," but the fields of biology and medicine continue to flourish.

Closer to the present topic, *computation* was formalized through Turing machines, but this definition is being questioned by those who consider *interaction* a key aspect of computation that is not handled by the traditional definition [Wegner, 1997]. We support the intuition that interaction is crucial, and seek to incorporate it below in our treatment of agenthood.

Tests It is widely recognized that definitions can never be perfect [Brachman, 1985]. This is because definitions are meant to be necessary and sufficient conditions. The concept of agents in general appears not to be crisp.

Consequently, and especially given the early stage of agents research, we propose not a definition but a *test* for agenthood. A test would identify some key features of agents, but without any implicature that it is necessary or sufficient. However, it can be used to contributed to determining membership in the class of agents.

2 The Agent Test Conceptually

Informally, we propose the following test:

> A system containing one or more reputed agents should change substantively if another of the reputed agents is added to the system.

We believe our claims apply to agents in general. However, for simplicity, we draw our examples primarily from software where the behavior of the systems modeled is expressible discretely. The above test requires some elaboration:

- By substantively, we do not mean that the system will simply slow down because another software process is running, but that the reputed agents will somehow be aware of each other and adjust their behavior accordingly.

- The added agent must be of the same type as the reputed agents. That is, it should have the same architecture and functionality, although it might have different goals, knowledge, or beliefs.

- The existing and added agents do not necessarily have to communicate. Nor do they have to be autonomous, persistent, or intelligent, although these qualities would help.

Here is how to apply the test. Suppose the agenthood of a piece of software, X, is in question. According to the above test, we would perform the following (mental) experiment. Imagine that there are two instances of X. The test states that if X is an agent type, each instance should behave differently in the presence of the other instance than when it is alone. If it doesn't behave any differently, then X is not an agent.

The agent test would pass the Distributed Vehicle Monitoring Testbed (DVMT) [Lesser & Corkill, 1983] and distributed sensor net agents, pass the agents in WARREN [Sycara, 1997], fail mail daemons, fail spreadsheets (and other software programs that just act on behalf of a user, which is a common definition for an agent), and fail simple Java applets. The test is independent of the mobility of the agents.

For example, the distributed Vehicle Monitoring Testbed (DVMT) [Lesser & Corkill, 1983] is a system of agents that sense their environment for the presence of moving vehicles. After detecting the possible presence of a vehicle, the agents communicate with each other to resolve ambiguities, refine their estimates of vehicle locations, and eliminate "ghost" vehicles. When a suspected vehicle enters a region where the coverage of two or more agents overlaps, the agents cooperate in confirming detection of the vehicle. Where the coverage is adjoining, the agents cooperate in determining the extended path of the agent. The agents communicate possible vehicle locations, partial paths, and raw sensory data.

If another sensing agent is added to DVMT, the new agent would help in confirming or eliminating vehicle tracks proposed by the other agents, thus affecting their behavior, and most likely improving their performance. These are agents according to the Agent Test.

The blackboard-based distributed HEARSAY-II [Erman *et al.*, 1980] is a system of agents for speech understanding. The agents are specialized, with some performing signal processing, while others are performing syntactic, semantic, or prosodic analyses. The agents concurrently process selected portions of an utterance, and then exchange hypotheses and results via a blackboard. If another lexical or signal-processing agent is added to the system, it changes what gets put on or taken off the blackboard, and the other agents are affected. These also are agents.

As a third example, utility-based agents for domains involving the delivery of goods [Rosenschein & Zlotkin, 1994] rely on negotiation mechanisms to determine their behavior. The negotiation mechanisms require direct interactions among the agents in a domain. Any additional agent of this kind would affect the negotiations. Hence, these agents would pass the Agent Test.

By contrast, some information-retrieval "agents" that find web documents for a user, if you add another one, it would do exactly the same thing as the first "agent" and the user would end up with two copies of each document. This is because the two agents would be completely unaware of each other, and could not take advantage of each other's activities. According to our test, these are not agents.

3 The Agent Test Specified

Fundamentally, the proposed test is based on the sociability of agents as manifested in changes in their behavior. How can we specify this property well enough that the test could be meaningfully applied by any developer or analyzer of agents. Although the test is not a simple mathematical result, the following discussion introduces the key ideas necessary to applying it in a uniform manner. The main components of the specification are the environment in which the agents exist, and a formulation of sociability. These are combined to yield a statement of the test, which can then be applied. Some properties and additional concepts follow from this statement.

3.1 The Environment

Consider the *state* of an environment. This evolves as agents act and spontaneous events occur in the environment.

- The state of an information environment consists of resources, connections and paths among the resources, and which of the resources are accessible, locked, opened, closed, enabled, deleted, visited, and so on. An agent's actions include locking files, reading or writing to a database or index, and creating resources.
- A *history* is a sequence of environment states. This corresponds to the history of what has transpired—what the state was at a designated initial time, and how it has evolved through agent actions and environmental events.
- A *configuration* is an environment along with some set of programs executing in the environment.
- The *generated set* of a configuration is the set of histories that may result from that configuration. It tells us all that *can* happen in a configuration.

Our test is based on comparisons of the generated sets of suitably related configurations.

Assumptions The following assumptions are crucial. The first two are a form of *realism*.

- The environment can make spontaneous changes, which are usually negative. That is, there are no Maxwellian demons. For example, in an information environment, spontaneous events include occurrences such as a file being locked or a CPU being overloaded because of some underlying tasks. An environment that spontaneously reduces disorder would be desirable but rather unrealistic!
- The environment cannot fake the existence of other agents. That is, there are no Cartesian demons.
- The supposed agent must be sufficiently delineated from the environment.
- Enough of the environment and the software—states of resources, and actions of putative agents—must be observable.

3.2 Sociability Semantics of the Test

Assume we are given an environment and some purported agent—an executing program—that exists and functions in that environment. What would happen if more programs of the same type were introduced into the environment? Obviously there would be some change in the behavior of the program already there, regardless of whether the new program was an agent. For the program to pass our agent test, however, the changes must be appropriate, or agent-like, in the following way.

An appropriate change relates to agents themselves, not to the infrastructure. For example, as the number of agents inhabiting an environment increases, so do the chances that there will be resource conflicts among them. If the programs share a CPU or network, they may run slower; they may livelock or deadlock in trying to access the underlying files. These are not in themselves agent-specific interactions; they can also occur solely as a result of the environment. To be agents, they should behave as if they recognize when other agents are being introduced.

If the program design in question has anything to do with agency, its instances will be smart enough to recognize and interact with each other. If they are not, they can still have some interference, of course. But interference is inherently a kind of interaction that goes through the infrastructure, and can occur readily in the environment without intervention by an agent.

A *distinguished resource* is a resource that has been selected for some special reason. Intuitively, these resources have some bearing upon the problem itself. The other resources could essentially be functioning as communication channels among the computations and thus would have different trajectories even if the changes are irrelevant from the social perspective.

Thus a more careful formulation of the test for agenthood could rely on the key characteristic that adding more of the purported agents causes some change in the possible sequences. To make sure that the changes are not gratuitous, we must check that the changes could not be caused solely by adjusting the environment. When we add this refinement, the test becomes more robust.

The basic idea in the test is to define two configurations that are identical except for the presence of an additional supposed agent instance in one of them. If we observe a difference in the trajectory of any distinguished resource, we can declare that the supposed agent type passes. Examples of this are given in section 2.

3.3 Some Properties and Extensions

The above test requires an agent's behavior to change when encountering other agents of the same type. Thus this is a test for *autosocial* agents.

We can generalize the idea to *heterosocial* agents, i.e., agents who can socialize with agents of different types. The extension is not straightforward, because we must ensure that the second agent is not the cause of whatever modifications we observe. To ensure the proper scope, we require that if X appears to behave differently because of Y, then Y should also behave differently because of X. Thus, we must consider three configurations that are identical except for the fact that one has just X, the other just Y, and the third both X and Y. If the joint configuration differs from each of the others, then we can declare that both X and Y pass.

We think of this as an important property of social interactions and term it the *symmetry thesis:* if we claim that A is an agent because it socially interacts with B, then B is also social because it interacts with A.

We can consider some properties of the autosocial and heterosocial versions of the test. Each is symmetric, although the autosocial test is trivially so. However, the heterosocial version is not transitive over agent types.

How an agent achieves its change in behavior, such as whether or not it communicates with the new agent, is not important. This is a useful property, because the internal details of an agent may be unknown or extremely complex. The proposed test tries to identify a property that can be verified or disproved without knowing the details of the agents' construction.

4 Discussion

A test is not a prescription so much as a filter. And whereas definitions in the classical Socratic sense are expected to be necessary and sufficient conditions, a test may be one or the other or even neither. Also, a definition would typically identify the essence of its subject, whereas a test—such as the one we propose—has only to identify properties that can be observed and tested. In some sense, this test represents a performance criterion, which is more powerful than a definition and a lot easier to manage.

The agent test is like the Turing test for intelligence in that it can be used as a sufficiency test. However, it is not like the Turing test in that it is not up to a human being to make a subjective assessment. If we said an agent is sociable if it appears so to a human, that would be a closer analogy to the Turing test.

It is like the Church Hypothesis in attempting to relate an informal concept (in this case, agenthood) to a formal structure or entity—passing our formal test. Such claims can never be proved formally, because one key part is informal. However, they can be supported by anecdotal evidence. In Church's case, since every "reasonable" notion of computing (prior to interaction [Wegner, 1997]) was shown equivalent to Turing machines, that was evidence in favor of Turing machines being a good canonical view of computing. In our case, the anecdotal evidence is provided by relating concepts of agenthood—as realized in systems such as DVMT and others—to our formal definition.

There are some important benefits from engaging in the present exercise. A formal notion can be used as the object of further study—its merits and demerits can be analyzed and debated leading to a clearer picture of the concepts one is trying to explicate. The proposed test is an important step in the present stage of the science of agents, because by attempting to make our intuitive notions precise, it promises to guide us closer to our goal as a research community. More practically, a good test can serve as a basis for a methodology for evaluating putative agent-based systems as well as the toolkits with which they are built.

References

[Brachman, 1985] Brachman, Ronald J.; 1985. I lied about the trees. *AI Magazine* 6(3).

[Erman *et al.*, 1980] Erman, Lee; Hayes-Roth, Frederick; Lesser, Victor R.; and Reddy, Raj; 1980. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys* 12(2):213–253.

[Franklin & Graesser, 1997] Franklin, Stan and Graesser, Art; 1997. Is it an agent or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III: Agent Theories, Architectures, and Languages*. Springer-Verlag. 21–35.

[Huhns & Singh, 1997] Huhns, Michael N. and Singh, Munindar P., editors; 1997. *Readings in Agents*. Morgan Kaufmann, San Francisco.

[Jennings & Wooldridge, 1997] Jennings, Nicholas R. and Wooldridge, Michael J., editors; 1997. *Agent Technology: Foundations, Applications, Markets*. Springer-Verlag, Berlin.

[Lesser & Corkill, 1983] Lesser, Victor R. and Corkill, Daniel D.; 1983. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine* 4(3):15–33.

[Petrie, 1996] Petrie, Charles J. Jr.; 1996. Agent-based engineering, the web, and intelligence. *IEEE Expert* 11(6).

[Rosenschein & Zlotkin, 1994] Rosenschein, Jeffrey S. and Zlotkin, Gilad; 1994. *Rules of Encounter*. MIT Press, Cambridge, MA.

[Sycara, 1997] Sycara, Katia; 1997. Warren: Intelligent agents for financial portfolio management. http://www.cs.cmu.edu/˜softagents/warren/.

[Wegner, 1997] Wegner, Peter; 1997. Why interaction is more powerful than algorithms. *Communications of the ACM* 40(5):80–91.