Indirect Interaction in Environments for Multi-Agent Systems

David Keil, Dina Goldin[1]

University of Connecticut, Storrs, CT USA
{dkeil,dqg}@engr.uconn.edu

Abstract. The E4MAS community is leading an effort to accept environments of multi-agent systems as a first-class entity, distinguishing *indirect interaction* via the environment from the environment's role in message transport. This paper defines classes of interaction (sequential and multi-agent, direct and indirect) and environments (physical and virtual, persistent and amnesic, dynamic and static). These notions provide an underpinning for proper acknowledgement of the roles of MAS environments and for powerful MAS design techniques that use indirect interaction. We explore the limitations of MAS models that are restricted to message passing and suggest research directions for constructing more powerful models.

1 Introduction

In situated multi-agent systems (MAS), the environment is independently active, in effect providing a shared memory, and making possible decentralized coordination [41]. For these reasons, research in multi-agent systems has begun to accept the environment of a MAS as a first-class entity. The E4MAS community [7] is at the forefront of this effort, pointing out how agents may use the environment in coordinating their activities.

The CFP for E4MAS 2005 distinguishes interaction via message transport from interaction via the environment, and points toward the more active conception of the environment in the latter case. *Indirect interaction* is precisely the nontrivial use of the environment in the way that the E4MAS CFP suggests. The trivial use of the environment for message transport is a case of what we call *direct interaction.*

Currently accepted models of concurrency explicitly represent only direct interaction (message passing). A major challenge for MAS research is to break out of the restricted framework that limits the environment to the role of message transport. Meeting this challenge entails providing a theoretical underpinning for indirect interaction among agents via the environment. To support research in multi-agent systems and their environments, new formalizations are required that explicitly represent indirect interaction.

[1] Supported by NSF award 0545489.

After surveying of research in environments for multi-agent systems, we outline steps toward formalizations that will address the needs of MAS research. We make a formal distinction between use of the environment for message transport and use of it for indirect interaction, and define a useful taxonomy of environments. In particular, we categorize environments along three dimensions:

- *persistent* vs. *amnesic*
- *dynamic* vs. *static*
- *physical* vs. *virtual*

Persistent environments have memory of past interactions with agents; persistence is a prerequisite for indirect interaction. Dynamic environments change their responses to agents over time; as a result, agents in dynamic environments may need adaptive behavior to meet utility criteria. Physical environments are ones in which situated agents operate; these environments can offer the greatest challenges, having memory with which to counter-adapt and lacking the predictability of computing entities.

Research in indirect interaction via the environment points to the computational power of this form of interaction. For example, natural systems of extremely simple agents, such as amoebae and ants, perform complex tasks via indirect interaction, also known as *stigmergy* (Sect. 3.3). Indirect interaction offers a powerful tool for MAS design (Sect. 4.3). The limitations of the standard models of concurrency, which are based on message passing (direct interaction), prevent them from modeling this form of interaction (Section 5.3).

Our work is part of an effort within the theory community to account for and to model interactive forms of computation [40, 12, 38], analogous to the successful modeling of *algorithmic* computing [19] with Turing machines [36]. We believe that new models of multi-agent interaction, based on indirect interaction, are needed to adequately underpin research in multi-agent systems.

Outline. In Sect. 2 we survey environments in MAS research, provide a definition of the notion of an environment, and classify environments. As a foundation for formal discussion of MAS environments, we offer in Sect. 3 a set of definitions of interactive computation, the environment of a single computing agent, persistence in environments, and direct and indirect interaction. Properties of indirect interaction and examples of it in E4MAS research are presented in Sect. 4. In Sect. 5, we highlight the limitations of the message-passing model of Milner. Sect. 6 sums up and points to related work and future research challenges.

2 MAS Environments

In this section, we survey E4MAS research (Sect. 2.1). This research has produced a number of characterizations and definitions of the term "environment." Based on this survey, and on our own work in models of interaction, we formulate a unifying definition of MAS environments (Sect. 2.2). We also provide a taxonomy, i.e., a set of dimensions along which to classify MAS environments

(Sect. 2.3). Our definitions apply to physical environments of embedded agents as well as virtual environments of software agents.

2.1 Environments in E4MAS Research

Research in environments for multi-agent systems has produced many examples of environments, with some common features.

A web site may be described as an environment for agents that visit it via HTTP. Its spatial structure is defined by pages hyper-linked together. Users move through this environment; hence linking and unlinking to a page is a form of mobility. The Multilayered Multi-Agent Situated Systems (MMASS) model [3] represents such an environment as an undirected graph where pages are vertices and links are edges. Since multiple clients may visit the same web site concurrently, and may change the state of data stored at the site, it is possible not only for agents to interact with the site, but also for agents to interact with each other via the site.

The web site here is an environment via which clients interact with each other, not by passing messages to each other, but anonymously and without handshake. This research is recognized as pointing to "a new form of interaction" among simultaneous visitors to a web site [3]. Traditional discourse about web-based interaction focuses on direct file transfers between servers and individual clients (browsers viewing pages).

An environment may be specifically the *physical* (analog) setting in which agents are situated. Alternatively, it may be a *virtual* (digital) setting for agents, subject to being shaped by system designers. Indirect interaction via the physical environment is also known as *stigmergy*. MAS-based manufacturing control serves as an example of stigmergy and of *self-organization*, the attribute of a *decentralized* MAS that displays behavior at a global level that is more than the composition of the behaviors at the lower level, of subcomponents. The Product-Resource-Order-Staff Architecture (PROSA) [37] supports self-organization by agents.

In a MAS of situated agents, the environment of each agent is often a combination of virtual and physical. An example of such a combination is the system of automatic guided vehicles (AGVs) that transport loads through a warehouse, adapting flexibly to changing conditions [42]. The application described supports a *local virtual environment* for each vehicles, which constitute a decentralized system. These virtual environments contain three types of information: *static* (e.g., layout of factory floor); *observable* (from other local virtual environments; e.g., position of an AGV); and *shared* (modifiable in two local virtual environments concurrently; e.g., traffic map). When a nearby load (located in the physical environment) requires transport, the location of that load is communicated to the AGV via its virtual environment; agents can also communicate with each other with the help of their local virtual environments.

Another example is a PDA-based museum-visitor scenario for access to museum information [43]. Visitors may interact with the museum environment and may also interact via a shared environment of tuples, such as when expressing

the desire for a group meeting, Artwork and visitor agents may enable human visitors to move toward art pieces of interest, or toward other visitors for group meetings.

Sometimes, however, situated agents may live in a purely virtual environments, such as the the distributed ant algorithm framework of [22].

It is possible to view the virtual environment as (non-agent) entities *in* a MAS that affect agents [34, 41]. Often, it is a single software entity with individual objectives, which communicates with the agents in the MAS. Unlike the agents, it is an independent and self-supporting entity, offering support services to the agents, including services of mediation between agents [39].

Seen as an *organizational layer*, a MAS environment may be designed to impose *constraints* on agent behavior. A common constraint is that of *locality*; that is, agents in a MAS each have their own *local* environment, and their interaction is limited to this environment. This notion of locality adds a *spatial* aspect to a MAS environment [41].

Some discussion of environments refers to *software or hardware platforms* as "environments," and other discussion refers to entities that interact with systems of agents as their "environments." Consider a MAS running as an application on a computer. We may say that the MAS's environment is the set of entities with which the agents interact, or we may alternatively say its environment is the computer and its operating system, and all the data streaming through ports of the computer. These are completely different senses of the word "environment."

Sometimes, a MAS environment is referred to as *application environment.* The application environment may be defined as "the logical entity that represents the space in which the Application Agents perform their job." This notion of environment can be contrasted with the execution infrastructure of a MAS, sometimes referred to as its *execution environment*; examples are those provided by Jade, LIME, JavaSpaces, or Retsina. The term "Distributed Agent Environment" (DAE) has been used to refer to the support infrastructure [43].

Both notions of the environment are meaningful, but to use the same word for both within the same field of research presents a conflict and risks being highly misleading. Our interest in this paper lies only with application environments; we will not be considering execution environments.

2.2 Defining Environments

As researchers in models of interactive computation, we suggest ways to unify and reconcile the various descriptions of environments presented in Sect. 2.1. This work is aimed towards providing foundations for new models of multi-agent interaction. We also suggest below some dimensions of a taxonomy of multi-agent system environments.

The following concise definition emerges from the examples in Sect. 2.1:

Definition 1. *An* environment *of a multi-agent system is a* physical *or* virtual *setting that acts as the producer of the system's inputs and consumer of its outputs. The environment of a single computing agent is simply the environment of the multi-agent system that consists only of this agent.* □

Our approach to conceptualizing environments positions them as a *relative* notion that is defined by *interaction*. An environment of a system is identified with the set of agents or other entities with which it interacts, and the environment of an agent is identified likewise. Suppose that agents in a multi-agent system interact with each other and with the system environment; then, these agents are part of each other's environment, though they are not part of the system's environment. In fact, if the system were to consist of two agents that interact exclusively with each other, then each would *be* each other's environment. This underscores the relative nature of the concept of environment.

It might be objected that environments and agents are distinct because agents are *active* and environments may be *passive*. But consider the standpoint of agent A that interacts with a "passive" entity B, which also interacts with agent C. Then as observed by A, B changes its state autonomously (when C changes B's state); hence from A's perspective, B does not behave passively. Thus, to agents such as A, passive environmental entities may be indistinguishable from active agent entities.

When two agents share the same environment, or have overlapping local environments (see Sect. 2.1), it follows naturally that these agents may end up affecting each other, by reading from and writing to their common environment. We say that they interact *indirectly* via their environment; it is this form of interaction that is of particular interest to us.

2.3 A Taxonomy of Environments

We focus on some *properties* of environments that help establish a taxonomy of MAS environments. In their popular textbook on artificial intelligence [31], Russell and Norvig have listed the following dimensions of environments, among others: *dynamic versus static*, *accessible* versus *inaccessible*; *deterministic* versus *non-deterministic*; *discrete* versus *continuous*.

We suggest here the dimensions of *persistent versus amnesic*, *dynamic versus static*, and *physical versus virtual* as the basis for a taxonomy of agent environments [17]. Other dimensions to be considered include *centralized versus decentralized* [16, 9].

We begin with the *static/dynamic* dimension in our taxonomy of MAS environments:

Definition 2. *An environment E is* static *with respect to an agent A if A's inputs from E are strictly dependent on A's outputs to E. A* dynamic *environment is one that is not static.* □

For example, a lamp that lights dependably when plugged in and goes out when unplugged defines a static environment w.r.t. the agent operating the lamp. A lamp with a light sensor, which lights when plugged in only if the room is dark, defines a dynamic environment with respect to the agent.

. Dynamic environments are characterized by their capacity to change autonomously. If agents A and C are both interacting with an entity B (but not with each other), and C's interaction with B affects B's behavior, then B is a

dynamic environment with respect to A. Clearly, an environment that is static is more predictable than a dynamic one.

We now define a class of environments that *can* remember from previous interactions:

Definition 3. *An environment is* persistent *if its outputs depend not only on its immediately preceding inputs, but also on earlier inputs. An environment is* amnesic *if it is not persistent.* □

For example, an electric light with a button switch, that lights when it is off and the user presses the button, but turns off when it is on and the user presses the same button, defines a persistent environment. A piece of paper defines a persistent environment w.r.t. a person writing on the paper. The air is amnesic w.r.t. a person singing to it, and persistent w.r.t. a person spraying perfume into it.

A persistent virtual environment can be viewed as a *reactive system* [21]; so can *intelligent agents* (as opposed to *reflex agents* [31]). When systems of intelligent agents operate in persistent environments, their design is concerned with utility maximization, rather than with satisfaction of predicates as in the design of functional systems [45].

When an environment is persistent, it is sometimes modeled as a Markov decision process (MDP); when it is also dynamic, it is modeled as a partially observable Markov decision process (POMDP); see [14].

Our taxonomy distinguishes real-world environments from the digital environments of finite computing devices:

Definition 4. *A* physical *environment is one observable only by analog sensors. A* virtual *environment is accessed digitally.* □

Note that our definition of environments (Definition 1) is general enough to encompass both virtual and physical environments. When an agent's environment is physical (the real world), the kinds of input and output are different from those in virtual environments. In artificial intelligence, these inputs and outputs are known as *percepts* and *actions*, respectively. An example is an automatic car [8], whose percepts consist of video camera snapshots, and whose actions consist of turning the wheel or pushing on the brake pedal. According to our definition, the road constitutes the environment of an automatic car.

Since we are interested in indirect interaction, our special concern is with environments that have persistence, a necessary precondition for indirect interaction. The notion of persistence is consistent with Piaget's definition of *behavior*, whose purpose is to change the state of the environment in a way favorable to a species or organism [29]. Persistence enables a higher long-term expectation of reward from the environment, not just a higher immediate reward.

The environment in which we drive a car is dynamic, persistent, and physical. It changes with respect to the car whenever we cause the car to move. Cars, pedestrians, and other potential obstacles appear and vanish in a way that our actions determine.

Not all dynamic environments are *persistent*, in the sense that they "remember" what happens to them, and this influences their later input to agents. Consider the environment of a theater stage, with either a play or a comedy routine; the audience is the agent. Both the play and the comedy routine are dynamic. The play is amnesic, in that the viewers' reaction cannot affect its progress. By contrast, the comedy routine is persistent, in that the comedian may make comments about audience members, or change his jokes in response to audience reaction.

3 Interactive Computation

As we have suggested, identifying the environment of an agent or system of agents presupposes the existence of *interaction*. In this section, we formalize the notion of interaction, making the possibility of mutual *causality* a condition for applying it (Sect. 3.1). We then distinguish two forms of interaction, *direct* and *indirect* (Sect. 3.2), with examples (Sect. 3.3).

3.1 Sequential and Multi-Agent Interaction

Interaction is a form of concurrent computation where communication is viewed as occurring *during* the computing process, rather than *before* or *after* it [13].

Definition 5. Interactive computation *is the ongoing exchange of data among the participants (agents or their environment) such that the output of each participant may causally influence its later inputs.* □

Exchange of data that never affects the actions of the recipient, and never has a later effect on the inputs of the sender, is not true interaction between them. A person who responds to what is shown on a television screen, by talking to or shouting at the television, is not interacting with it. A microphone and an amplifier do not interact unless *feedback* is present. Research in cybernetics fifty years ago recognized feedback or mutual influence as a feature that distinguishes an important kind of coupling of systems, both natural and artificial ones [44, 2, 32].

We consider feedback essential to interaction. One-way communication without feedback should be distinguished from interaction; likewise, input alone is not interaction. Interaction must allow mutual causality, hence the semantics of interaction demand that feedback be present.

Note that the outputs of two agents may causally influence their later inputs without the agents communicating directly; they may communicate via an intermediary. In that case the causality and interaction are *indirect* (Sect. 3.2).

In interactive computation, the role of the environment is heightened, relative to the role of the environment in the execution of an algorithm. An *algorithm* is a description of the steps for effectively transforming an input to an output, so that the output is a (computable) function of the input [19]. Once the environment

determines the input value passed to an algorithm, its job is done. By contrast, the environment participates *throughout* an interactive computation.

For algorithmic (function-based) computation, captured by Turing machines, the properties of an environment are of no consequence, because one execution of an algorithm simply computes a function on whatever single input arrives from the environment. For interactive computation, on the other hand, how an action by a computing agent will change its environment can be crucial to the choices made by the agent.

The simplest form of interactive computation is when the communication between a system and its environment takes place via a single stream (channel or interface). This type of interaction is referred to as *sequential*:

Definition 6. Sequential interactive computation *is interaction involving two participants, at least one of which is a finite computing agent (machine, device).* □

By definition, the other participant in sequential interactive computation serves as the environment of the first.

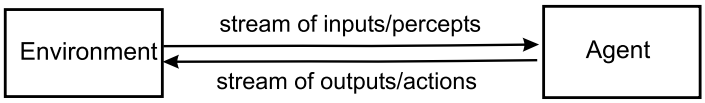

Fig. 1. Agent and environment

In sequential interaction, there is a single input stream and a single output stream between the computing entity and its environment (see Fig. 1). The temporal interleaving of these two streams creates a single *interaction stream* that consists of pairs of input and output tokens. In a physical environment, the notion of *sensing* by a computing agent corresponds to *getting input*, and the notion of *actuating* corresponds to *emitting output*. Sequential interactive computation has been formalized by *Persistent Turing Machines* (Sect. 5.1).

As we explain in Sections 3.2 and 4.2, persistence of state in an environment makes possible indirect interaction among agents, which in turn facilitates *systems* of agents, i.e., MASs. Sequential interaction with its single interaction stream is distinguished from *multi-agent interaction*, where multiple autonomous streams may exist simultaneously between the system and its environment. This is illustrated in Figure 2; the dashed box indicates the boundary between the system and the external environment.

We define multi-agent interaction as follows:

Definition 7. Multi-agent interaction *is interactive computation involving* more *than two agents; the agents are assumed to be* asynchronous. □

It is conjectured that models of multi-agent interaction such as the *multi-stream interaction machine* (MIM) are more expressive than models of sequential

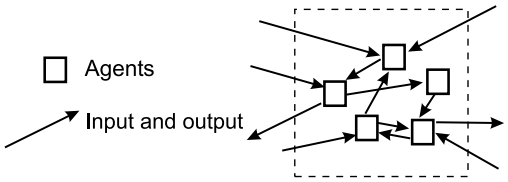

Fig. 2. Multi-agent interaction

interaction [40]. The remainder of this paper focuses on multi-agent interaction and MASs.

3.2 Direct and Indirect Interaction

It is commonly thought that interaction is the same as communication, associated with the notion of *message passing* [26] or *targeted send/receive* (TSR) [10]. We refer to this form of interaction as *direct*:

Definition 8. Direct interaction *is interaction via messages, where the identifier of the recipient is specified in a message.* □

Message passing is appropriate when two entities possess identifying information about each other and have the intent to communicate with each other. However, entities may also interact without knowing about each other, and without even having any intent to communicate. This happens when they share a common environment, and the changes made by one of them to the environment can be observed by the other. This second category of interaction is known as *indirect*, originally defined in [15]:

Definition 9. Indirect interaction *is interaction via persistent, observable changes to a common environment; recipients are any agents that will observe these changes.* □

Note that the acts of making changes (output) and of observing those changes (input) are decoupled in time; persistence of the environment allows the change to endure, allowing for a lapse of time before it is observed. Furthermore, the identity of the observer(s) may not yet be determined when the environment is changed, allowing for *anonymous* interaction. In fact, the change need not be motivated by the need to communicate, but may occur as a byproduct of the first agent's computation.

While indirect interaction differs significantly from direct interaction (communication), both clearly satisfy the definition of interaction (Definition 5). However, in computing, indirect interaction seems to be treated as a poor cousin of the direct form, as it is not supported by current models of interaction.

Indirect interaction deserves the same level of recognition and the same attention from the theoretical community as does direct interaction. It is not by accident that multi-agent systems occurring in nature often rely on indirect interaction to "get their job done." Likewise, indirect interaction via work in progress,

shared writing, pictures, and publicly viewed symbols of all kinds, is the basis for most forms of human production and for shared culture and markets of all kinds. Some examples of such multi-agent systems are presented next.

3.3 Examples of Indirect Interaction

In this section, we discuss several examples of indirect interaction via the environment. The first three are based on naturally occurring multi-agent systems: ant trails [4, 5], termite piles [30], and slime molds [18]. The last is a classic example of distributed computation, the *Dining Philosophers* problem [6].

Most of us have seen a column of ants forming a busy highway on our floor or our counter top. Ant colonies solve the problem of efficiently foraging for food sources by a decentralized approach involving multi-agent indirect interaction, where each ant deposits *pheromones* (slowly evaporating scent chemicals) as it carries food, and each ant follows pheromone trails as it searches for food. The pheromone trail to a food source gets reinforced over time, turning into a highway. Once the food source is exhausted, the trail is no longer reinforced, and it evaporates over time. Without a plan, the ants find paths to the food that tend toward optimality.

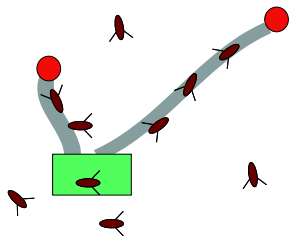

Fig. 3. Ant trails

The problem of building an ant hill or a termite pile is likewise solved in a decentralized fashion, relying on indirect interaction. In the StarLogo termite simulation [33], the termites build a single circular pile of wood chips, despite having no capacity for planning or coordination, and with minimal ability to perceive. They continuously apply a simple protocol: move at random, pick up a chip whenever one is encountered, and put it down when the termite bumps into another chip. Eventually, a single pile emerges. Remarkably, the termites accomplish this task without having an internal representation of their eventual goal.

Another naturally occurring multi-agent behavior that depends on indirect interaction is the formation of *slime molds*. When their food is scarce, *slime mold* amoeba organisms emit a "distress" chemical that causes them to gravitate to one another. The chemical signal is relayed among these microscopic organisms,

and they migrate in a spiral toward the center of such signals, eventually aggregating in huge numbers into a single crawling slug-like organism that is large enough to see with the naked eye. Again, aggregate behavior emerges from individual behavior through indirect interaction, without centralized direction.

In each of the above examples, the global behavior of the population is more than the composition of the individual behaviors. No one in that population ever decides where the trail will lie, where the pile will be located, or where the slime mold slug will raise its head. Individually, these organisms are too simple to have an "understanding" of such complex phenomena, yet their behavior as an aggregate predictably results in these phenomena. *Emergent behavior* occurs in such complex systems, which exhibit *self-organization.*

The classic *Dining Philosophers* problem is another example of indirect interaction. In this problem, philosophers sit at a circular table, with one chop-stick between each pair of diners. Each one occasionally interrupts her thinking to pick up two chop-sticks – first one, then another – and eat, then put them down, repeating these steps when hungry. The problem is to avoid starvation by deadlock; if each diner simultaneously picks up the left chop-stick, for example, and holds it until the other one is available, then all will starve. One solution involves *exogenous coordination* by managing *channels* [1].

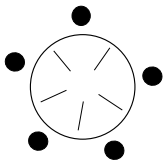

Fig. 4. Dining Philosophers

This problem is often formulated in terms of concurrent communicating processes that share common resources. The objective is to define a protocol for a ring-shaped arrangement of processes, each communicating only with its two neighbors, such that all processes are allowed to move forward under the constraint that no two adjacent ones may execute simultaneously. Yet in the original problem, the philosophers never communicate with each other; in fact, it is not even clear that they are aware of each other, as they sit deep in thought or search hungrily for their chop-sticks. Despite the lack of direct communication, each philosopher's behavior is affected by the others, since she cannot eat while her neighbor is eating. This is due to indirect interaction; the philosophers communicate *indirectly* via the chop-sticks.

4 Indirect Interaction and Multi-agent Systems

Indirect interaction is ubiquitous. In this section, we discuss the use of indirect interaction in research in environments for multi-agent systems (Sect. 4.1). We

then focus on the properties of indirect interaction, which include *name decoupling* (anonymity), *time decoupling* (asynchrony), *space decoupling* (locality), and *non-intentionality* (Sect. 4.2) and sketch an approach to MAS design that makes use of indirect interaction (Sect. 4.3).

4.1 Indirect Interaction in MAS Research and Beyond

The examples in Section 3.3 have inspired the design of MAS systems based on indirect interaction. *Digital pheromones*, which are data structures inspired by the insect model, have been used for coordination of autonomous vehicles [28]. A related approach is to use *force fields* [20], which are generated by the agents, propagated via some embedded infrastructure, and perceived locally by those agents in the vicinity of the field.

MAS's for manufacturing control are an example of indirect interaction, where the environment is "a first-class abstraction" [37]. Rather than communicating only with each other, the various agents involved (resources, orders, products, etc.) all interact with the common factory environment and adjust their behavior according to what they observe in that environment.

In another example, a web site serves as a shared environment for the agents that visit it via HTTP. Its spatial structure is defined by pages hyper-linked together. Users move through this environment; hence linking and unlinking to a page is a form of mobility. Since agents can both write to, and read from, the web pages, it is possible not only for agents to interact with the site, but also for agents to interact with each other indirectly, via the site. This interaction is anonymous, without handshake. This research is recognized in [3] as pointing to "a new form of interaction"; we can now see that this refers to indirect interaction.

[42] introduces a virtual persistent environment for situated MASs of autonomous guided vehicles (AGVs) that is an alternative to the approach of [28]. The virtual environment is alongside the physical environment; it is distributed over physical agents and synchronized using middleware. It maintains a shared memory of the agents, and its role is to set rules of dynamic relationships among agents. This is in effect *exogenous coordination* [1].

From the above examples it is clear that creation of a persistent virtual environment, adequate for supporting indirect interaction, is useful for engineering certain applications. We also believe that for some forms of system behavior, indirect interaction is not only useful but necessary; direct interaction is not sufficient.

Indirect interaction, where agents (organisms) interact via their *shared environment* rather than by exchanging messages directly, is ubiquitous outside of MAS research as well. In addition to above examples, there are many others, both within and outside computer science. Here is a sampling from various fields:

- *Operating systems:* Processes exchange information via semaphores in shared memory;

- *Programming languages:* The Linda language uses tuple spaces to enable coordination by indirect interaction [10];
- *Anatomy:* Cells exchange information via hormones in the bloodstream;
- *Social biology:* In stigmergy, social insects interact indirectly by leaving trails of pheromone chemicals [35];
- *Sociology:* Most group dynamics consist of actions or percepts whose destinations or sources are other than one's immediate partner in a communication;
- *Multi-agent systems:* Agents communicate indirectly either through intermediary agents or through changes in the environment;
- *Economics:* the storage and publication of stock market listings enables large numbers of buyers and sellers to interact indirectly to negotiate prices.

In the next section, we discuss indirect interaction in multi-agent systems at greater depth, focusing on its properties.

4.2 Properties of Indirect Interaction

Several characteristics of indirect interaction distinguish it from message passing (direct interaction):

- *time decoupling* (asynchrony): due to persistence of the environment, there may be a delay between the state change and its observation;
- *anonymity*: the observer's identity need not be known to the originator of the state change;
- *late binding of recipient*: the identity of the observer of changes to the environment may be determined dynamically by events occurring after the change is made;
- *space decoupling* (locality): indirect interaction of mobile agents need not imply co-location; one may leave after making state changes, and the second state may arrive later to observe them;
- *non-intentionality*: indirect interaction does not require an intent to communicate;
- *analog form*: the medium of indirect interaction may be the real world, e.g. for embedded or situated agents (robots, sensors).

Asynchrony in indirect interaction follows from time delay due to the persistence of the observable changes in the environment. By contrast, message passing is synchronous; synchronization occurs when one process stops and waits until the other has reached a certain point in the computation or interaction. Synchronization by handshake enables processes to begin communicating with each other.

Examples of synchronous interaction are conversations and the TELNET protocol. Asynchronous interaction includes exchanges of email and communication among ants via pheromone trails. Message queues enable asynchronous communication when interaction is direct.

Anonymous interaction occurs when computing entities interact without knowledge of each other's identities. In indirect interaction, anonymity is not only

possible but often necessary, when the identity of the observer (recipient) is not determined until after the change to the environment has been made. When an ant leaves a pheromone trail, it is not yet known which of the ants milling nearby will be the one to pick up this scent. Similarly, after a termite drops a wood chip, many things can happen that will eventually determine which termite will pick it up.

The Dining Philosophers problem also exhibits the above characteristics. This problem has the properties of (1) *locality*, because diners can only see neighboring chop-sticks; (2) *anonymity*, because diners need not know each other's names or even whether their neighbors exist; (3) *asynchrony*, because a diner doesn't necessarily pick up a chop-stick as soon as it is put down; and (4) *non-intentionality* of communication, because diners pick up (put down) chop-sticks so as to eat (think), not to communicate.

A *mobile* version of the Dining Philosophers problem can also be defined, where the philosophers may leave to go sleep, then come back to occupy the first empty chair they can find. This mobile version exhibits the additional property of *late binding of recipient*, since the identity of one's neighbor cannot be known ahead of time.

4.3 Indirect Interaction and MAS Design

While indirect interaction serves as a valuable communication paradigm for all types of agency, its properties make it ideal for the design of large complex systems from small simple agents. Ants, termites, and slime molds from the previous section are all examples of such systems that occur in nature.

In II-MAS, it is assumed that the agents are much simpler than their environment; in the extreme case, agents are finite-state automata (single-celled organisms), while the environment is the real world. The agents in II-MAS are therefore expected to lack the cognitive abilities to observe and act on the state of the complete environment. As a result, they must limit themselves to a small part of their environment that we call their *locality*, where locality may be either physical or virtual. In either case, if the locality of an agent may change during the computation, we refer to the agent as *mobile*. As a result of mobility, the recipient of indirect interaction need never share the same locality as the initiator of that interaction; this is what we call *space decoupling*.

In an II-MAS, each agent's protocols for its interaction with the environment are simple and repeatable. While there may be more than one type of ant (worker, drone, etc.), there is no need to provide a unique program for each ant or termite. The agents in II-MAS are not concerned with other agents, except as those agents make changes to their shared environment. There are no issues of scalability or redesign if agents are added to, or removed from, the system during the computation.

Anonymity ensures that agents in II-MAS are interchangeable, while asynchrony allows systems to be adaptable. Unlike algorithms, where a single deviation from the plan means an incorrect result for the computation, II-MAS can be designed to withstand dynamic changes in the agent population, or to

the environment. If the food source moves, the ant trails will eventually lead to the new location; if the wood chips are disturbed, the termites will eventually reassemble them.

5 Modeling Multi-Agent Systems and Their Environments

Just as a gap has been observed between the *theory of computation* and *concurrency theory* [13], we also see a gap between research in *multi-agent systems* and the modeling of *concurrent distributed systems*. Whereas indirect interaction via the environment can play a key role in MASs, the traditional message-passing model of concurrency provides no first-class representation of indirect interaction.

Below we present a model of sequential interaction (Sect. 5.1) as well as Robin Milner's model of concurrency, based on message passing (Sect. 5.2), and highlight their limitations with respect to MAS research (Sect. 5.3).

5.1 A Formal Model of Sequential Interaction

Multi-agent interaction (Definition 7) has to date been modeled as as a set of concurrent instances of sequential interactions (Definition 6). Let us begin with a model of sequential interactive computation based on Turing machines, the *Persistent Turing Machine* (PTM) [11, 12]. The PTM is a Turing machine with three special features that distinguish it from a TM:

- The TM's executions are *iterated* so that the PTM performs an infinite series of TM computations;
- the input/output semantics of the PTM differ from those of the TM in that input and output are dynamically generated *streams*; where later input tokens to the PTM may depend on its earlier output tokens;
- The PTM has a *persistent* tape, which retains its contents between one execution of the TM and the next.

A PTM is a nondeterministic 3-tape Turing machine (N3TM) with a read-only input tape, a read/write work tape, and a write-only output tape. Upon receiving an input token from its environment on its input tape, a PTM computes for a while and then outputs the result to the environment on its output tape, and this process is repeated forever. A PTM performs *persistent computations* in the sense that a notion of "memory" (work-tape contents) is maintained from one computation step to the next, where each PTM computation step represents an N3TM computation.

Persistence extends the effect of inputs. An input token affects the computation of its corresponding macrostep, including the work tape. The work tape in turn affects subsequent computation steps. If the work tape were erased, then the input token could not affect subsequent macrosteps, but only "its

own" macrostep. With persistence, an input token can affect all subsequent macrosteps; this property is known as *history dependence.*

The treatment of PTMs has proceeded along the following lines. Our team first formalized the notions of interaction and persistence in PTMs in terms of the persistent stream language (PSL) of a PTM. Given a PTM, its persistent stream language is coinductively defined to be the set of infinite sequences (interaction streams) of pairs of the form (w_i, w_o) representing the input and output strings of a single PTM computation step. Persistent stream languages induce a natural, stream-based notion of equivalence for PTMs. *Decider PTMs* are an important subclass of PTMs; a PTM is a *decider* if it does not have divergent (non-halting) computations.

Our team defined the class of *amnesic* PTMs and a corresponding notion of amnesic stream language (ASL). In this case, the PTM begins each new computation with a blank work tape. It was shown that the class of ASLs is strictly contained in the class of PSLs, and that ASL-equivalence coincides with the equivalence induced by considering interaction-stream prefixes of length one, the bottom of our equivalence hierarchy; and that this hierarchy collapses in the case of amnesic PTMs. ASLs are representative of the classical view of Turing-machine computation. One may consequently conclude that, in a stream-based setting, the extension of the Turing-machine model with persistence is a non-trivial one, and provides a formal foundation for reasoning about programming concepts such as objects with static attributes.

In an analogous fashion to the Church-Turing Thesis, our team hypothesized that anything intuitively computable by a sequential interactive computation can be computed by a persistent Turing machine. This hypothesis, when combined with other results, implies that the class of sequential interactive computations is more expressive than the class of algorithmic computations, and thus is capable of solving a wider range of problems.

5.2 Milner's Model of Concurrency

Robin Milner is a pioneer of models of interaction, winner of the 1992 Turing award. Milner observed that *reactive* systems are unlike *algorithmic* ones in that they do not compute *functions* [23]. His Calculus of Communicating Systems (CCS) formalized some notions about interaction, including the notion of *concurrent composition* in which systems are composed (placed in communication), equivalent to the notion that they *observe* each other [24]. Alongside (mutual) observation, a second fundamental idea of the CCS was *synchronized communication.* Thus under Milner's theory, communication is *sequential interaction* between *two* agents.

Milner's π-calculus is a successor to CCS that leaves unchanged the basic assumption that interaction is reducible to binary communication. Agents take turns emitting and receiving data, synchronized by a *handshake.* When agents lack handshake protocol (as, for example, virtually throughout nature), they may use *buffers* to avoid loss of data. Models mentioned by [24] as alternatives

to synchronized models include the Actor Systems of Hewitt and the model of Kahn and McQueen based on unbounded buffers and queues.

In Milner's model, communication is in the form of *messages*, where the emitting agent knows the identifier of the receiving agent. Hence this form of communication is sometimes called *targeted send/receive* (TSR). This assumption is explicitly stated in [25].

Milner has consistently made clear that interaction between agents via their environment is to be modeled by treating shared memory as *processes*: "I also insisted that memory registers be modeled as processes..." [26]. In this way, the underlying assumption is maintained that all interaction is message passing between mutually identified processes.

A higher development of this theory occurred with the π-calculus, created to model *mobility*, i.e., the dynamic creation or breaking of links between pairs of processes in a multi-agent system. The mobility of π-calculus was motivated in [27] by positing the case where one agent (*source*) wishes to send a value to a second (*destination*) via a third (*intermediary*). The model supports this indirect but targeted communication by enabling the source to send to the intermediary both the data intended for the destination, and a reference or link to the destination.

While adding mobility, π-calculus does not deviate from CCS's original philosophy of modeling all interaction as *synchronized* communication via *message passing* from agents to *pre-identified recipients*. In the next section, we challenge this approach.

5.3 Limitations of the Message-Passing Model

Can message passing simulate indirect interaction, as concurrency theory assumes? Is it always possible to replace the environment, via which agents in a MAS interact indirectly, with a mere transport medium for direct interaction (message passing)?

The assumption that all multi-agent interaction is modeled adequately by message passing has never been proved. [25], for example, does not assert that a message-passing model is suitable for interaction in general. Rather, it assumes that shared variables constitute the only alternative to direct interaction, and shows how a finite number of discrete shared variables can be replaced by corresponding communicating processes.

The most obvious criticism of this model is its limited *expressiveness*. In particular, it excludes situated and embedded systems, whose environments are physical (real-world), or a combination of virtual and physical. Physical environments are *continuous* and *analogue*; in the case of the real world, they may be infinite as well. No finite set of shared discrete variables can adequately represent all possible observations over such environments.

In fact, we believe that it can be shown that message passing is not as expressive as indirect interaction. That is, any model based on message passing can be simulated by a model based on indirect interaction. On the other hand,

there exist stigmergic systems not captured by any model based on message passing.

Another criticism concerns the *scalability* of direct interaction, especially in the context of very large systems of very simple agents (Sect. 4.3). Assume that the size of the system (the number of agents) increases, while the complexity of each agent remains fixed. By representing the communications within the system by a graph, we see that this graph must either continue increasing its density (average number of links adjacent on any single agent) and/or its diameter (average number of links between any pair of agents). It can be shown that either of these cases results in a gradual increase in the agents' cognitive load. Since the complexity of agents is fixed, the system must eventually reach a crisis point where it can no longer carry out its job.

The final criticism is on more aesthetic grounds. Models based on direct interaction simply fail to explicitly capture the properties of indirect interaction. Indirect interaction has several properties that distinguish it from direct interaction, such as dynamic binding, anonymity, and asynchrony (Sect. 4.2). A formalization of indirect interaction must explicitly reflect these properties; models of interaction as message passing do not.

The Dining Philosophers problem (Sect. 3.3) serves as an illustration. This problem can be modeled as direct interaction between philosophers and chop-sticks, as in the original solution; each chop-stick is a process, which communicates with the two philosophers on either side of it. In this problem, however, chop-sticks are not autonomous computing entities, like philosophers; they are passive, initiating no action. Their only roles are to reflect the states of the philosophers next to them. The semantics of the problem are of indirect interaction among diners, not direct interaction between diners and utensils.

6 Conclusion

Communication (direct interaction) is not the only type of interaction. A multitude of indirect interaction examples exist. Indirect interaction is appropriate for MAS design, as evidenced by examples in E4MAS research, including cases of agents interacting via web sites, warehouse-floor transport, and PDA-based coordination of human activity in museums. We introduced a design strategy, II-MAS, that uses agents that are much simpler than their environments but that use their environments to interact as components of highly-adaptive multi-agent systems.

We have shown that indirect interaction via the environment is a distinct and useful part of multi-agent systems operation, meriting formalization via definitions and models. The notions of *dynamism*, *persistence*, and *physicality/virtuality* of environments are part of a taxonomy of environments for multi-agent systems that feature indirect interaction.

In assessing current models of multi-agent interaction, we showed that *message-passing* cannot adequately model multi-agent interaction, which includes both direct and indirect interaction. Therefore, we showed, a model of multi-agent

interaction that is adequate for reasoning about, and providing an underpinning for, multi-agent systems, cannot limit itself to representing message passing (direct interaction), but must explicitly include indirect interaction within its scope.

New formalisms are required that incorporate indirect interaction explicitly, supporting the notion of environments as first-class entities in multi-agent systems, rather than just as transport media. They must also support the notions of *anonymity, asynchrony, locality, non-intentionality*, and *mobility*.

Future research challenges include:

- formalizing multi-agent systems in a way that explicitly incorporates indirect interaction;
- formally proving the greater expressiveness of models that allow indirect interaction over those that do not.

References

1. Farhad Arbab. Reo: A Channel-Based Coordination Model for Component Composition. *CWI Report SEN-0203*, 2002.
2. W. Ross Ashby. *An Introduction to Cybernetics.* University Paperbacks, 1964.
3. Stefania Bandini, Sara Manzoni, and Giuseppe Vizzari. Web Sites as Agents' Environments: General Framework and Applications. In *Second International Workshop on Environments for Multiagent Systems (E4MAS)*, 2005.
4. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford Univ. Press, 1999.
5. Eric Bonabeau and Guy Theraulaz. Swarm Smarts. *Scientific American*, pages 72–74, March 2000.
6. Edsger Dijkstra. Hierarchical Ordering of Sequential Processes. *Acta Inform.*, 1:115–138, 1971.
7. Environments for Multi-Agent Systems 2005 (E4MAS). *http://www.cs.kuleuven.ac.be/ distrinet/events/e4mas/2005/.*
8. Eugene Eberbach, Dina Goldin, and Peter Wegner. Turing's Ideas and Models of Computation. *In Christof Teuscher, ed.,* Alan Turing: Life and Legacy of a Great Thinker, *Springer*, 2004.
9. Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence.* Addison Wesley Longman, 1999.
10. D. Gelernter and N. Carriero. Coordination Languages and Their Significance. *CACM*, 35(2):97–107, 1992.
11. Dina Goldin. Persistent Turing Machines as a Model of Interactive Computation. *in: K-D. Schewe and B. Thalheim (Eds.),* Foundations of information and knowledge systems, First Int'l Symposium (FoIKS'2000), *LNCS 1762*, pages 116–135, 2000.
12. Dina Goldin, Scott A. Smolka, Paul Attie, and Elaine Sonderegger. Turing Machines, Transition Systems, and Interaction. *Information and Computation Journal*, 194(2):101–128, Nov. 2004.
13. Dina Goldin and Peter Wegner. The Church-Turing Thesis: Breaking the Myth. In *Proc., CiE 2005, Amsterdam, June 2005 LNCS 3526, Springer*, pages 152–168, 2005.

14. C.V. Goldman and S. Zilberstein. Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.
15. David Keil and Dina Goldin. Modeling Indirect Interaction in Open Computational Systems. *TAPOCS Workshop, Proc. WET ICE 03*, 2003.
16. David Keil and Dina Goldin. Indirect Interaction and Decentralized Coordination. *Extended draft, http://www.cse.uconn.edu/ dqg/papers/indirect.pdf* 2004.
17. David Keil and Dina Goldin. Adaptation and Evolution in Dynamic Persistent Environments. In *Proc. FInCo2005, Edinburgh*, 2005.
18. James Kennedy and Russell Eberhart. *Swarm intelligence*. Morgan Kaufman, 2001.
19. Donald E. Knuth. *The art of computer programming, Vol. 1: Fundamental algorithms*. Addison-Wesley, 1968.
20. Marco Mamei, Franco Zambonelli, and Letizia Leonardi. Distributed Motion Coordination with Co-Fields: A Case Study in Urban Traffic Management. In *Proceedings of the The Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)*, 2003.
21. Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer Verlag, 1992.
22. Koenraad Mertens and Tom Holvoet. CSAA: A Distributed Ant Algorithm Framework for Constraint Satisfaction. In *FLAIRS Conference 2004*.
23. Robin Milner. Processes: A Mathematical Model of Computing Agents. *H. E. Rose and J. C. Shepherdson, Eds.*, Logic Colloquium '73, *North-Holland*, 1975.
24. Robin Milner. *A Calculus of Communicating Systems*. LNCS 92, Springer, 1980.
25. Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.
26. Robin Milner. Elements of Interaction. *Comm. ACM*, 36(1):78–89, 1993.
27. Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge Univ. Press, 1999.
28. H. Van Dyke Parunak, Sven Brueckner, and John Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems Bologna, Italy*, 2002.
29. Jean Piaget. *Behavior and evolution*. Pantheon, 1978.
30. Mitchel Resnick. *Turtles, Termites, and Traffic Jams*. MIT Press, 1994.
31. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Addison-Wesley, 1995.
32. Herbert Simon. *The Sciences of the Artificial*. MIT Press, 1970.
33. Starlogo site at MIT Media Lab. *http://starlogo.www.media.mit.edu / people / starlogo*. 2000.
34. Renee Steiner, Gary Leask, and Rym Z. Mili. An Architecture for MAS Simulation Environments. In *E4MAS*, 2005.
35. G. Theraulaz and E. Bonabeau. A Brief History of Stigmergy. *Artificial Life*, 5:97–116, 1999.
36. Alan Turing. On Computable Numbers with an Application to the Entscheidungsproblem. *Proc. London Math Society*, 2(DK: GET PUB):Reprinted in Martin Davis, Ed., The Undecidable, pp. 173–198, 1936.
37. Paul Valckenaers and Tom Holvoet. The Environment: an Essential Abstraction for Managing Complexity in MAS-based Manufacturing Control. In *E4MAS*, 2005.
38. Jan van Leeuwen and Jiri Wiedermann. The Turing machine paradigm in contemporary computing. *In B. Enquist and W. Schmidt, Eds., Mathematics unlimited – and beyond, Springer-Verlag, 20*, 01, 2001.

39. Mirko Viroli, Andrea Omicini, and Alessandro Ricci. Engineering MAS Environment with Artifacts. In *E4MAS*, 2005.
40. Peter Wegner. Why Interaction is More Powerful Than Algorithms. *Communications of the ACM*, 40(5), 1997.
41. D. Weyns, H. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for Multiagent Systems: State-of-the-Art and Research Challenges. In *Proc., E4MAS*, 2004.
42. Danny Weyns, Kurt Schelfthout, and Tom Holvoet. Exploiting a Virtual Environment in a Real-World Application. In *E4MAS*, 2005.
43. Danny Weyns, Giuseppe Vizarri, and Tom Holvoet. Environments for Situated Multi-Agent Systems: Beyond Infrastructure. In *E4MAS*, 2005.
44. Norbert Wiener. *Cybernetics, or control and communication in the animal and the machine, 2nd Ed.* MIT Press, 1961.
45. M. Wooldridge. On the Sources of Complexity in Agent Design. *Applied Artificial Intelligence*, 14(7):623–644, 2000.