

# Modeling Indirect Interaction in Open Computational Systems

David Keil  
University of Connecticut  
dkeil@engr.uconn.edu

Dina Goldin  
University of Connecticut  
dgg@engr.uconn.edu

## Abstract

*Open systems are part of a paradigm shift from algorithmic to interactive computation. Multiagent systems in nature that exhibit emergent behavior and stigmergy offer inspiration for research in open systems and enabling technologies for collaboration. This contribution distinguishes two types of interaction, directly via messages, and indirectly via persistent observable state changes. Models of collaboration are incomplete if they fail to explicitly represent indirect interaction; a richer set of system behaviors is possible when computational entities interact indirectly, including via analog media, such as the real world, than when interaction is exclusively direct. Indirect interaction is therefore a precondition for certain emergent behaviors.*

## 1. Introduction

Openness in computational systems is closely related to the capacity of *agents* to interact with their *environments* during computation. Open systems are part of a paradigm shift from *algorithmic* to *interactive* computation [20].

Openness of systems may be defined in a number of ways, including: (1) openness to new information; i.e., learning; (2) openness to new components or agents [3]. Indirect interaction within such a system can enhance its openness by allowing anonymous, nonspecialized interfaces that invite participation of new knowledge sources.

### 1.1 Direct and indirect interaction

Organisms and societies in nature offer instances of indirect interaction that can inspire new ideas to enable human collaboration.

*Direct* interaction is characterized by the exchange of data over time between computing agents that possess identifying information about each other, or between an agent and its environment (Section 2). This message-passing notion is the way interaction is traditionally modeled, e.g., in concurrent systems [10].

A theory of open systems must also incorporate *indirect* forms of interaction, which rely on *persistence* and *observability* of changes in the environment. The property of persistence in interactive systems is a powerful one. For example, extending Turing machines with persistence and stream-based semantics yields a model that is more expressive than TMs without persistence [6].

One of the simplest ways an environment with persistent state can be altered by agents in it is by serving as the *medium* for indirect interaction among the agents, e.g., *stigmergy*, a feature of natural systems in which agents' behavior is shaped by interactions with unknown other agents, as occurs in ant colonies (Section 3.2).

The *anonymous* communication enables *openness* of a system, because participants may enter and exit the system without drastically changing it. Anonymity is consistent with *space decoupling*, making possible *non-locality* of interaction between mobile agents that interact indirectly.

### 1.2 Some interaction patterns found in nature

This subsection provides motivating examples of indirect interaction in nature.

To support design for collaboration technologies, we need to model as rich a variety of behaviors as possible. The following are examples of simple interaction patterns in nature that yield complex results. We will refer to these cases later.

Multiagent systems in nature that exhibit emergent behavior and stigmergy offer inspiration for research in open systems and enabling technologies for collaboration (Section 3). To address the perceived need for an integrated theory of collaboration [15], this paper will present examples to support a case for inclusion of models of indirect interaction in a theory of open systems.

#### Example 1: Termites gathering wood chips

In the StarLogo termite simulation [13, 18], the termites build a 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. This global behavior of the termite population is more than the composition of the individual chip-carrying behaviors. The termites accomplish this task, in a self-sustaining manner, without an internal representation of the goal [13].

### Example 2: Ants foraging for food

Ant colonies solve the problem of efficiently foraging for food sources by a decentralized multiagent interaction in which each ant deposits *pheromones* (evaporating scent chemicals) as it walks, and each ant follows pheromone trails as well as food odors. Heavily traveled (hence strong, hence attractive) pheromone trails correspond to short paths to food. As food is exhausted at a site, the trails to it evaporate. Without a plan, the ants find a set of paths to the food that tends toward optimality [1].

### Example 3: Slime mold aggregation

When their food is scarce, slime mold amoeba organisms gravitate to one another by use of a chemical signal emitted into the environment. The signal is relayed among the organisms, which migrate toward the center of a spiral of such signals, eventually aggregating into a single crawling slug-like organism [9]. Again, aggregate behavior emerges from individual interaction without centralized direction.

**Outline.** In Section 2 we define direct interaction and present current models of concurrency and interaction, such as Robin Milner’s. Section 3 defines indirect interaction and presents a critique of models that represent a shared environment as a *process*. Here a theorem is presented that indirect interaction using the real world can display a greater range of behaviors than direct interaction among computational entities, and a conjecture is offered that even models of artificial computing systems are incomplete without explicit representation of indirect interaction. Section 4 presents theory and examples of indirect interaction in multiagent systems: emergent behavior, stigmergy, blackboard systems, and self-organizing sensor networks. Finally, in Section 5 we summarize the case for our conjecture and suggest future work.

## 2. Direct interaction

This section defines interaction in general, interaction in its direct variant, and Milner’s “new conceptual framework” for concurrency theory, and contrasts interactive to algorithmic computation.

Algorithms are characterized by a strict time ordering, in which finite input is followed by computation, which is followed by finite output. By contrast, interactive computation is characterized by the following features:

- (a) interleaving of inputs and outputs during computation;

- (b) stream input/output;
- (c) history-dependent dynamic behavior of the computing agent.

**Definition 1.** *Interaction* is the ongoing two-way or multiway exchange of data among computational entities, such that the output of one entity, perceived by another one, may causally influence the later outputs of the second entity.

Algorithmic computation is associated with structured programming, batch systems, and Turing machines; interactive computing is associated with object-oriented programming, graphical user interfaces, networked and distributed systems, and emerging post-Turing-machine formalisms.

In his 1991 Turing Award lecture, Robin Milner explained that a “new conceptual framework” is needed to model concurrency and interaction [11]. He focused on how the *semantics* of concurrency sharply distinguishes concurrent from sequential programs; namely, in that the *compositionality* of sequential programs is lost with concurrency. Whereas sequential processes cannot mutually interfere, concurrent ones may do so, and the semantic mapping from memories to memories does not apply in the presence of concurrency.

To express interaction requires *non-algorithmic* models of computation. *Labeled transition systems* (LTSs), in which state transitions accompany observable actions, provide a foundation for an operational semantics of interactive agent behavior [19]. *Conditionings* of agents due to inputs, and spontaneous moves by agents, yield *configuration* (state) changes and *manifestations* (outputs). Patterns of such interactions may be devised to specify the behaviors of agents [19].

Milner went on to model the interaction of concurrent processes with shared variables by “elevating” memory “to the status of a process” [11].

**Definition 2.** *Direct interaction* is interaction via *messages*; destinations are those agents specified in the message.

Direct interaction may be synchronous or asynchronous. The interaction between concurrent processes, studied in *concurrency theory*, is direct; it consists of message passing via handshake between asynchronous processes [10]. Day-to-day examples of direct interaction include in-person conversations, conversations by telephone, email, and messages delivered between acquaintances via intermediary.

## 3. Indirect interaction

In this section, we define indirect interaction and argue that any model that represents a shared environment as a *process* is incomplete. In particular we point out that

indirect interaction among computational entities, using the real world, has a strictly greater range of behaviors than direct interaction.

### 3.1 Dynamic environments with memory

This subsection motivates indirect interaction by pointing out the greater richness of behaviors observable in a dynamic environment with persistent memory.

A sharp distinction may be drawn between kinds of dynamic environments that are not affected by agents, and real-world-type environments that are shaped by the agents within them. One objective of artificial complex and adaptive systems is to act to influence the environment in a way favorable to an agent's performance measure in the long run.

An adequate semantics of interaction will recognize a *symmetry* between agents and their environments, whose actions alter each other's states. Just as an agent's actions are dependent on its previous percepts, in an environment with *memory*, or persistent state, the agent's percepts may depend on earlier actions. Likewise, symmetrically, actions of an agent that has memory may depend not only on the agent's current percept but also on earlier ones.

It is readily evident that *persistent state* is a fundamental attribute both of dynamic real-world-like environments and of agents that operate in them. Two agents may interact with their common environment in such a way that they interact indirectly with each other.

### 3.2 Indirect interaction

This subsection defines indirect interaction and discusses its properties.

Interaction in nature is often *indirect*, through agent actions that cause persistent and observable changes that later affect other agents. Social organisms in nature, mentioned in Section 1.2, provide instances of communication via the environment.

**Definition 3.** *Indirect interaction* is interaction via *persistent, observable state changes*; destinations are any agents that will observe these changes.

By our definition, indirect interaction features *anonymity*, where the identity of the recipient is not known at the time the interaction is initiated. Indirect interaction can also involve *time delay (decoupling)*, due to the *persistence of the observable changes over time*. Space decoupling can also be involved, when the agents participating in the interaction never share the same location. Notably, use of the real world as an interaction medium in indirect interaction is not precluded.

Whereas in direct interaction a message's destination is fixed, in indirect interaction the identity of the receiver depends on dynamically generated events such as the

entry of agents into the vicinity of data emitted. Indirect interaction is therefore a natural model for systems of *mobile* agents whose ability to perceive and act upon their environment is localized, in contrast to random access models of computation.

The examples in Section 1.2 all rely on indirect interaction for achieving desired system behavior. For instance, termites interact indirectly via the chips that they pick up and drop in the environment.

Note that *persistence* plays a crucial role in the definition of indirect interaction. A formal model of interactive computation that incorporates persistence is the *Persistent Turing Machine* (PTM), equivalent to the *Interactive Transition System* (ITS). This model combines the constraint of *computable transitions*, as in Turing machines, with the extension to non-algorithmic, interactive processing associated with persistent state [5, 6].

### 3.3 Why models limited to direct interaction are incomplete

This subsection discusses why models of complex dynamic systems are incomplete if they lack explicit representation of indirect interaction.

**Theorem 1.** The behavior of computational agents that make use of indirect interaction via the real world is strictly richer than the behavior of agents that interact directly.

**Proof.** The real world is not a computing agent and may be analog in nature. It has been shown that analog systems may compute algorithmically uncomputable functions [16].

**Corollary.** Models that represent indirect interaction in an explicit way are strictly more expressive than ones that do not.

Clearly, even the slightest error of value storage in a real-world analog medium can drastically change the outcomes of indirect interactions, due to chaotic behavior.

We next address the failure of existing models to convey the *semantics* of indirect interaction.

The changing state of the environment as processes interact may be represented as shared memory. In Milner's contribution, by contrast, shared variables are represented as *processes* [11].

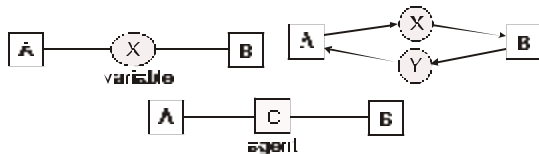
We believe that models that represent shared variables as if they were separate agents (processes) fail to convey the *passive* role of data in the environment, and thus fail to adequately model indirect interaction.

If we take the natural step of referring to interaction (specifically, coordination) among *agents* instead of *processes*, it is quite clear that the medium of information

exchange is *second-tier* (“Coordination media are not agents” [14]).

The semantics of interactions of agents *A* and *B* via shared memory in effect consist of an interaction stream between *A* and *B*; the semantics of *A*’s and *B*’s interactions with shared memory by themselves are of only trivial interest.

Furthermore, whether the interaction medium here takes the form of one variable or two, or a third agent, is immaterial and can be abstracted from in modeling indirect interaction.



For two reasons, a model of interaction that fails to incorporate indirect interaction is incomplete:

- It will appear, misleadingly, that the design goal of the emitter of an output is to affect a shared variable if the shared variable is modeled as a process
- It will appear that the design goal for the receiver of input is to be affected by the shared variable, rather than the emitter of the output.

In both cases, the behavior of the two agents *toward each other* is not modeled properly.

For example, a model of slime mold aggregating that represented the chemical deposits in the environment as “processes,” without accounting for the communication between and among the amoeba cells, would hardly be an adequate one of the biological process occurring.

## 4. Indirect interaction in multiagent systems

This section explains how *stigmergy* uses indirect interaction, and applies our definitions to collaboration systems and sensor networks.

A richer form of interaction than the *sequential* kind is *multiagent interaction*. When events within a multiagent system are *non-serializable*, no sequential interaction stream can simulate them.

The behavior of multiagent systems is often characterized by the special features of *self-organization* and *emergence* (Section 4.1). A kind of emergence involving indirect interaction is *stigmergy* (Section 4.2). Two cases of indirect interaction in multiagent systems are the *blackboard-style architecture*, and *sensor networks* (Section 4.3).

### 4.1 Self-organization and emergent behavior

This section defines the notions of self-organization and emergent behavior.

Open computational systems can be more effective in enabling collaboration if they exhibit *self-organization*, defined as the interaction among a set of processes or structures at a lower level of a system to yield global structures at a higher level [7].

One of the features of multiagent systems of autonomous agents in general, and ones exhibiting indirect interaction in particular, is that control may be *decentralized* with gain, rather than loss, of effectiveness. Assumptions that all well-performing systems have “leaders” and centralized control are erroneous [13]. Multiple autonomous agents and the evolving environment, rather than a single “leader,” may drive a process.

*Emergence* is observed when the behavior of a whole system is more than the sum of the behaviors of the individual components. Termites gathering chips, ants foraging for food, and slime molds dividing and aggregating (Section 1.2) are instances of emergent behavior. Other instances are the way intelligent behavior emerges from unintelligent neurons in the brain, and the way humans collaborating can produce results of a different *qualitative* nature than when working separately.

The special features of emergent behavior are precisely due to the interactions of the component processes and agents. In a hierarchical or algorithmic system, such as an assembly line, all interactions are strictly controlled within the hierarchy.

Self-organization and its variant, emergence, are features of open systems in which indirect interaction is often crucial. Some types of emergence are inconceivable without indirect interaction, because the world is physically arranged in such a way that *distance* entails *time delay*, hence all direct interactions are *local* in one way or another. Thus, for a system *as a whole* to be influenced by one of its components, that component can only interact with remote components indirectly.

A distributed system, whose nodes interact indirectly via shared memory or the environment, is capable of emergent behavior.

### 4.2 Stigmergy

This subsection suggests that stigmergy, as a kind of indirect interaction, provides examples that will be useful in designing systems of autonomous computing agents.

*Stigmergy* is a variety of self-organization in which agents are *mobile*. In stigmergic communication, agents alter the state of the environment in such a way that other agents’ behavior is altered as a result [9]. Thus stigmergy presupposes indirect interaction.

The notion of *persistence* in the environment is crucial to stigmergy, because mobile agents must be able to make local changes to the environment that last long enough for other mobile agents to detect and be affected by them.

Stigmergy explains much complex behavior in social animals, even ones that are incapable of direct planning and coordination.

The simple individual behaviors of insects can yield complex aggregate behaviors and outcomes. Open computational systems can make use of this principle to leverage collaborative technology to yield results unobtainable from individual efforts. Termites may be known to each other only through indirect means; their interface is limited to the construction underway. This principle of indirect interaction raised to the level of human experts brings us to the blackboard model on which knowledge-based problem-solving applications are based (Section 4.3).

In the general case *pheromones* may be defined as local state changes to the environment, caused by mobile agents, where the local state returns to a stable value over time. Among the advantages of pheromone use is fault tolerance: backup plans are automatically present in scenarios such as the ant-foraging one [1].

Complex behaviors that emerge from indirect interaction of simple agents are sometimes known as *swarm intelligence*. In indirect interaction using pheromones, the environment exhibits *semipersistence*, rather than persistence. In semipersistence, a change in the environment dissipates over time but still enables separation in time.

The evident power of emergent behavior suggests that the design of technologies for collaboration will be aided by models of indirect interaction, because such models will express the wider range of behaviors that are possible with emergent behavior. The conjecture below spells out the foundation for this claim.

**Conjecture:** Even without making use of the real world, a system that executes indirect interaction via digital media may have a richer set of system behaviors than is possible with only direct interaction.

This implies that a model that gives explicit support only to direct interaction, using processes to represent the interaction medium, is incomplete in that it cannot express some set of behaviors generated by indirect interaction.

The behaviors not modeled will include emergent and stigmergic behaviors. These are cases in which the sum of a set of individual direct interactions is less than the aggregate behavior of the system, including that which emerges from indirect internal interactions.

### 4.3 Blackboard systems and sensor networks

This subsection explains how collaboration systems make use of indirect interaction.

Direct interaction between arbitrary agents imposes a need for protocols and languages to support communication. These may tend toward specialization; the more isolated a closed pair of agents is, the more specialized its interface may be. Anonymous indirect communication, on the other hand, generates flexible interfaces [3].

*Blackboard systems* have aimed to enable cooperation in solution development since the 1970s. Their component agents are each viewed as black boxes, but communicate with the blackboard using a common language. The managing element of such a system is a *control component* that specializes in coordination of agents. Interaction among autonomous and concurrent agents is via a centralized blackboard repository and triggered by events at the blackboard [3].

The control component of a blackboard system reflects the capacity of a multiagent system, not possessed by simple sequential interactive agents, to reconfigure itself. [20].

*Coordination languages* like Linda [4] support indirect interaction among independent cooperating processes, though a shared global environment. *Coordinator processes* manage interactions among *computation processes*, which collaborate only via the coordinator processes [12].

Blackboard-style software architectures called *repositories* are part of the framework of collaboration evolving in the software-engineering field.

Research inspired by biological phenomena has investigated self-organization in *sensor networks* for the task of clustering sensor data and learning to specialize at various aspects of state recognition [2]. Other sensor-network research has singled out stigmergy and swarm intelligence as the core of a technology to support robust, autonomous routing and optimization [8].

Unplanned and unplannable self-organizing behavior in simulation and in nature suggests that *specification by constraint*, and bottom-up evolution, of multiagent reactive systems may replace their top-down *design*. Blackboard systems and self-organizing sensor networks are cases of artificial open systems in which indirect interaction and persistence of state play central roles.

## 5. Conclusion

This paper has offered three reasons why models that fail to represent indirect interaction are incomplete:

- Behavior of systems in which the real world is an interaction medium is richer than that of systems limited to computing agents (Theorem 1, Section 3.3);
- The semantics of indirect interaction among two or more agents cannot be conveyed by models limited to direct interaction, in which the passive intermediaries are represented as processes (Section 3.3);

- The range of behaviors observable in systems where indirect interaction occurs among computing entities using a digital interaction medium, is greater than the range observable in systems featuring only direct interaction (Conjecture, Section 3.3).

Future work should further explore the relation between indirect interaction and analog computing, define a formal semantics of indirect interaction, and move toward formal verification of the above conjecture.

A “revolutionary” paradigm shift is occurring, driven partly by research in multiagent systems and motivated by qualitatively new features of computing systems: situatedness, openness, locality in control, and locality in interactions [22]. To account for the role of locality, new ways of modeling must incorporate indirect interaction explicitly.

Openness has a common foundation in the separate fields of computing, mathematics, and physics [20]. The limits of algorithmic notions of computing, of first-order logic, and of classical physics, are different expressions of the same closed-system assumptions. In the social and organizational realms, *centralized thinking* [13] is being replaced by new insights into the power of decentralized, multiagent behavior such as that found in swarms in nature. These systems were not designed, but rather specified by the constraints of their environments. Yet the power of the multiagent interactions found in them exceeds what the top-down design of artifacts has been capable of so far.

Lessons from the study of these systems can be incorporated into collaboration technologies. Existing and to-be-developed formal models of interaction, which will extend the transition-system-based formalizations found in concurrency theory, will be part of a firmly grounded theory of collaboration and open computational systems.

## Acknowledgement

We thank Daniel Corkill of the Multi-Agent Systems Laboratory at the University of Massachusetts, Amherst, for many useful comments on a draft of this work.

## 6. References

- [1] Eric Bonabeau and Guy Theraulaz. Swarm smarts. *Scientific American*, March 2000, pp. 72-79.
- [2] Elaine Catterall, Kristof Van Laerhoven, Martin Strohbach. Self-organization in ad hoc sensor networks: An empirical study, [www.comp.lancs.ac.uk/~strohbach/alife\\_2002.pdf](http://www.comp.lancs.ac.uk/~strohbach/alife_2002.pdf), 2002.
- [3] Daniel D. Corkill. Blackboard systems. *AI Expert* 6 (9), pp. 40-47, 1991.
- [4] David Gelernter, Nicholas Carriero. Generative Communication in Linda. *ACM TOPLAS*, vol. 7, no. 1, Jan. 1985.
- [5] Dina Goldin. Persistent Turing Machines as a Model of Interactive Computation. FoIKS'00, Cottbus, Germany, Feb. 2000.
- [6] Dina Q Goldin, Scott A. Smolka, Paul C. Attie, Peter Wegner. Turing machines, transition systems, and interaction. 8th Int'l Workshop on Expressiveness in Concurrency, Aarlborg, Denmark, August 2001.
- [7] Owen Holland and Chris Melhuish. Stigmergy, self-organisation, and sorting in collective robotics. *Artificial Life* 5 (2), pp. 173-202, 1999.
- [8] Ioannis N. Kassabalidis, Arindam K. Das, Mohamed A. El-Sharkawi, Robert J. Marks II, Payman Arabshahi, Andrew A. Gray. Intelligent routing and bandwidth allocation in wireless networks, 1/03.
- [9] James Kennedy and Russell C. Eberhart. *Swarm intelligence*. Morgan Kaufmann, 2001.
- [10] Robin Milner. Processes: A Mathematical Model of Computing Agents. In H. E. Rose and J. C. Shepherdson, Eds., *Logic Colloquium '73*, Amsterdam: North-Holland, 1975.
- [11] Robin Milner. Elements of Interaction. *Communications of the Association for Computing Machinery* 36 (1), pp. 78-89, 1993.
- [12] G. A. Papadopoulos and F. Arbab. *Coordination Models and languages*. Academic Press, 1998.
- [13] Mitchel Resnick. *Turtles, Termites, and Traffic Jams*. MIT Press, 1994.
- [14] Alessandro Ricci, Andrea Omicini, Enrico Denti. Activity Theory as a Framework for MAS Coordination ESAW'02.
- [15] Michelle P. Steves, Robert H. Allen. Evaluating collaborative enterprises -- A workshop report, WETICE 2001.
- [16] Hava Siegelmann. *Neural networks and analog computation: Beyond the Turing limit*. Birkhauser, 1999.
- [17] Herbert Simon. *The Sciences of the Artificial*. MIT Press, 1970.
- [18] Starlogo site at MIT Media Lab. <http://starlogo.www.media.mit.edu/people/starlogo>.
- [19] Mirko Viroli, Andrea Omicini. Specifying agent observable behavior. AAMAS'02, July 15-19, 2002, Bologna, Italy.
- [20] Peter Wegner. Interactive Foundations of Computing. *Theoretical Computer Science* 192, pp. 315-351, 1998.
- [21] Peter Wegner. A Research Agenda for Interactive Computing. <http://www.cs.brown.edu/people/pw/papers/icalp.ps>
- [22] Franco Zambonelli and H. Van Dyke Parunak. Signs of a revolution in computer science and software engineering. Proceedings, ESAW02.