

Toward Domain-Independent Formalization of Indirect Interaction

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

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

Abstract

Unlike direct interaction based on message passing, indirect interaction takes place between agents as they make and observe persistent changes to their shared environment. The Dining Philosophers and Foraging Ants problems illustrate the characteristics that distinguish indirect from direct interaction, such as persistence, space and time decoupling, dynamic binding of recipient, and lack of intent. We point out that indirect interaction is an important phenomenon in many different fields within and outside computer science, including open systems, and merits explicit formal modeling.

1. Introduction

Indirect interaction occurs when computing entities communicate by acting on, and observing, the persistent state of *their common environment* [4]. An example is the use of shared memory in concurrency.

We present indirect interaction as a first-class concept, requiring its own explicit *domain-independent* models on par with those that exist for direct interaction. We identify properties of indirect interaction, such as dynamic binding of communication recipients and time/space decoupling, that give it a different semantics from direct interaction.

Indirect interaction enables decentralized coordination. In biological systems, indirect interaction via the environment is known as *stigmergy* [5]. Indirect interaction may be found in many other fields. Models of open systems require proper formalization of indirect interaction; the more open a system is, the greater the role of indirect interaction

2. Characteristics of indirect interaction

Algorithms are characterized by separation of computation and interaction. Input is determined prior to computation and output follows computation. By contrast, *interactive computation* is characterized by input/output streams that are dynamically interleaved during the computation [6].

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

One kind of interaction is associated with the notion of *message passing* or *targeted send/receive* (TSR).

Definition 2. *Direct interaction* is interaction via *messages*; the recipient's identifier is specified in the message.

Often, agents affect each other's computation without direct interaction, when one of them makes changes to their shared environment that the other later observes:

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

Indirect interaction may exhibit many characteristics not present in message passing.

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

The persistent character of the environment (i.e., the ability to make and observe persistent changes to it) is a prerequisite for indirect interaction. Other properties are a natural consequence. The decoupling between sender and receiver in indirect interaction makes it natural for systems consisting of a multitude of simple agents whose ability to perceive and act upon their environment is localized [7]. We present two examples next.

3. Dining Philosophers and Foraging Ants

Dining Philosophers [2] is a classic problem. Philosophers sit around a table, with one chopstick between each pair of diners. They autonomously pick up the two chopsticks next to them; once full, they put down the chopsticks and think, until hungry again. If each philosopher

simultaneously picks up the left (or right) chopstick, and holds it until the right one is available, they will all starve.

More abstractly, the problem is to define a protocol for a ring-shaped arrangement of communicating processes, each communicating only with its two neighbors, such that all processes may move forward only under the constraint that no two adjacent ones execute simultaneously.

Dining Philosophers can be modeled as direct interaction between philosophers and utensils, as in the original solution and in [1]. However, the semantics of the problem are of interaction between diners, not between diners and utensils. In these semantics, chopsticks are not autonomous computing entities, like philosophers; they are *passive*, not capable of any actions.

Indirect interaction allows us to model the Dining Philosophers problem more naturally. Each philosopher interacts with his neighbors *indirectly*, by changing the state of their shared chopstick (on-table or with-diner). This problem has the following properties associated with indirect interaction:

- *Anonymity*: diners need not know each other's names or even whether their neighbors exist;
- *Asynchrony*: diners don't necessarily pick up a chopstick as soon as it is put down;
- *Non-intentionality*: diners pick up chopsticks to eat with and put them down to think, with no intent to communicate;
- *Locality*: diners can only see neighboring chopsticks.

To exemplify the properties of *late binding* and *space decoupling*, we can create a *mobile* version of this problem, where philosophers may leave or switch places. Their neighbors continue to rely on the state of the chopstick to make their decisions, not aware of the change in their neighbor's status.

Another example of indirect interaction is *Foraging Ants*. Ant colonies solve the problem of efficiently foraging for food sources by creating *pheromone trails*. Each ant deposits pheromones as it carries food, and each ant tends to follow the pheromone trails it finds [5]. Ants strengthen existing trails in a way that reinforces shorter paths. These trails are analog features of the ants' real-world environment; the ants (which might be robotic) interact indirectly via this analog medium.

Foraging Ants exemplify a *decentralized* multiagent system that is nevertheless highly *coordinated*. To accomplish the ants' task in a centralized way would require a 'command center' that is aware of the locations of food sources and decides which location each ant should exploit next. Without such a center, decentralized direct interaction mechanisms would force the ants to communicate via a message system of great complexity. Indirect interaction enables decentralized coordination by allowing the shared environment to act as a passive participant in the computation.

4. Towards Formalizing Indirect Interaction

The examples of ants interacting via a real-world environment and philosophers interacting via binary semaphores present two very different applications of indirect interaction. Indirect interaction occurs in many fields under many names, within and outside computer science:

- *Operating systems*: Processes interact via semaphores;
- *Programming languages*: The Linda language uses tuple spaces to enable coordination [3];
- *Anatomy*: Cells exchange information via hormones in the bloodstream;
- *Social biology*: Social insects interact indirectly by leaving trails of pheromone chemicals [5];
- *Sociology*: Most group dynamics consist of actions or percepts whose destinations or sources are other than one's immediate partner in a communication;
- *Multiagent systems*: Agents communicate either through intermediaries or changes in the environment;
- *Economics*: Stock market listings enable buyers and sellers to negotiate prices indirectly.

While indirect interaction can be simulated with message passing by employing special protocols, its formalization will allow us to model its properties explicitly, without need for an intermediate protocol layer. In the case of an analog environment, the argument for explicit models of indirect interaction is even stronger.

The Foraging Ants example shows that indirect interaction for multiagent decentralized systems is both more scalable and more effective than direct interaction. It is also better at reflecting the semantic distinction between active agents and their passive interaction medium. If the characteristics of a problem include indirect interaction, a model that can directly reflect these features is better than one that does not. A formalization of indirect interaction will contribute towards truer models for open systems.

5. References

- [1] F. Arbab. Reo: a channel-based coordination model for component composition. *Math. Structures in Comp. Sci.* 14 (3), Cambridge Journals Online, 2004, pp.329-366.
- [2] E. Dijkstra. Hierarchical ordering of sequential processes. *Acta Inform.* 1, 1971, pp. 115-138.
- [3] D. Gelernter, N. Carriero. Coordination languages and their significance. *Comm.ACM* 35(2), 1992, pp. 97-107.
- [4] D. Keil, D. Goldin. Modeling indirect interaction in open computational systems. TAPOCS Workshop, *Proceedings, WETICE'03*, 2003, pp. 355-360.
- [5] G. Theraulaz, E. Bonabeau. A brief history of stigmergy. *Artificial Life* 5, 1999, pp. 97-116.
- [6] P. Wegner. Interactive foundations of computing. *Theor. Comp. Sci.* 192, 1998, pp. 315-351.
- [7] F. Zambonelli, H. Parunak. Signs of a revolution in computer science and software engineering. *Proc. ESAW02*.