

# Scalable models of multi-stream interaction

*David Keil*

September 28, 2006

Framingham State College  
Math / Computer Science Colloquium

## Outline

- 1. Introduction**
- 2. Algorithms and sequential interaction**
- 3. Indirect interaction and multi-agent systems**
- 4. Models**

# 1. Introduction

## Food foraging problem for ants

- Food is scattered randomly
- Task is to take it to the nest
- Ants are small and limited in intelligence and communicating power
- Food may appear or disappear dynamically
- *A solution:*
  - Ants walk semi-randomly dropping pheromone
  - Ants tend also to follow pheromone trails
  - Ants carrying food drop special pheromone
  - Trails evolve toward short paths between nest and food

## This research

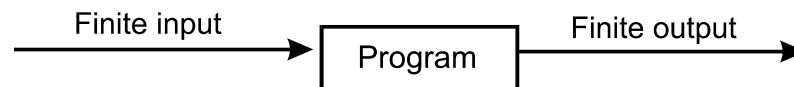
- We assert that the *mission* of this multi-agent system can only be accomplished using *indirect interaction* via the ants' environment (stigmergy)
- We seek to define a class of such missions requiring *scalability*
- Current models of mobile systems are based on *direct interaction* (message passing)
- We show that these models are *unscalable*
- Our interest is in *expressiveness of models of computation*; an expressiveness hierarchy may be defined

## 2. Algorithms and sequential interaction

## Algorithms

### ***Algorithmic computation (Knuth):***

The effective transformation of a finite, pre-specified input, to a finite output, in a finite number of steps.



- Algorithms *compute functions*
- A system that executes an algorithm is *closed*
- Algorithms are equivalent to Turing-machine computation

## Scalability in algorithms and interaction

- Time efficiency of algorithms is expressed as scalability (rise in running time relative to quantity of data operated on)
- *Intractable* problems are completely unscalable in that running time seems to be exponential in data size
- A robust reactive system likewise should respond in time that does not increase “too much” in proportion to size of its input

## Communication

- *One-way communication* is the sending of strings, over a finite alphabet of symbols, from one entity to another
- *Two-way communication* is the concurrent activity of two entities engaged in one-way communication with each other
- Two-way communication does not assume that either entity waits for an input string before emitting output, or that either entity has an exclusive communication relationship with the other.

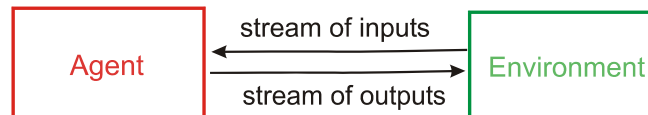
## Interaction and synchrony

- *Direct interaction* is two-way communication in which some outputs of each entity may causally affect the entity's later inputs from the other
- Computing entity *interacts synchronously* with environment  $E$  if  $A$  interacts with  $E$  and both  $A$  and  $E$  wait for only one input token before emitting an output token.
- *Asynchronous interaction* occurs in the absence of synchrony as defined here

## Sequential interaction

*(Synchronous) sequential interactive computation:*

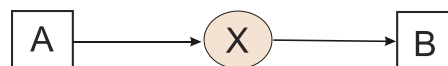
Interaction involving two participants, at least one of which is a finite computing agent (machine, device).



- Characterized by a single interaction stream of input alternating with output
- If one participant is an agent, the other is its *environment*
- Interaction may involve changes of state

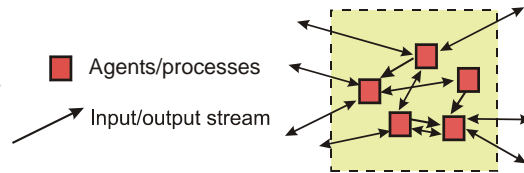
## Multi-stream and indirect interaction

- *Multi-stream interaction* occurs when an entity is concurrently interacting with more than one other entity
- Let  $A$  and  $E$  interact asynchronously. If  $E$  may be decomposed into  $E'$  and  $B$ , where  $E' = E - \{B\}$ , then  $A$  and  $B$  *interact indirectly* via  $E$  iff mutual causality holds between the behaviors of  $A$  and  $B$ .



## Multi-stream interaction

- In contrast to sequential interaction, multi-stream interaction may feature:
  - *Nondeterminism* when attempts to write collide
  - *Dynamic linking and unlinking*, creation/destruction of nodes
  - *Indirect* interaction via a shared environment



## 3. Indirect interaction and multi-agent systems

## Direct and indirect interaction

### *Direct interaction:*

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

### *Indirect interaction:*

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

- Sequential interaction is direct
- Preconditions for indirect interaction:
  - Agents share access to parts of the environment
  - Persistence of environment
- *Example of indirect interaction:* use of semaphores in process synchronization (critical section problem)

## Stigmergy in nature

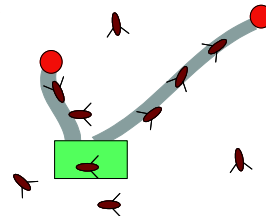
**1. Ants foraging** (see slide 4)

**2. Termites gathering chips into pile:**

Move at random, pick up chip when encountered, put down when another chip found; the pile structure is used to coordinate creation of pile (*StarLogo*)

**3. Slime mold dividing and aggregating:**

These amoeba may aggregate by emitting a chemical, migrating toward its greatest concentration



***Q: Is stigmergy essential for some missions?***

## Ubiquity of indirect interaction

- **Social biology:** Social insects interact by modifying common structures or through pheromones
- **Operating systems:** Processes communicate via *semaphores* in shared memory
- **Coordination languages:** Shared *tuple spaces* enable coordination in Linda
- **Anatomy:** Cells exchange information via *hormones* in the blood stream
- **Economics:** A *market* is an environment for buyers and sellers that serves as a medium for indirect interaction

## Properties of indirect interaction

- **Time decoupling (asynchrony):**  
State changes persist
- **Anonymity:** Recipient ID not used in access
- **Space decoupling:** Agents need not meet
- **Non-intentionality:** Agents need not have goal of communicating
- **Hybrid nature:**  
Physical environment may play role
- **Late binding** of recipient

## The power of persistence

- *Discussion:* It can be shown that computing agents with persistent state are capable of a wider range of behaviors than ones without persistent state
- *Example:* By remembering past answers, prosecutor may ask questions that force a witness to tell the truth or contradict self
- Persistence of environment is what enables termites, ants, slime mold to coordinate actions

## Mission of a multi-agent system

- Algebra and propositional logic provide rules for *evaluation* of formulas
- Algorithms compute recursively definable *functions*
- Sequential-interactive agents offer *services*
- Multi-agent systems accomplish *missions* requiring *quality of service* for all users
- Interaction in MASs may be *asynchronous*
- Mission may require a minimum QoS regardless of number of users (*scalability*)

## Indirect interaction and multi-agent systems

- In a MAS characterized by *locality* of interaction and *mobility* of agents, it is only possible for agents to influence overall system behavior by use of indirect interaction
- Richness of multiagent interaction:
  - It is due partly to ability of each agent to interact with multiple others
  - Hence each agent interacts indirectly with *all* others (otherwise system partitions)

## Decentralized, self-organizing systems

- Decentralized and self-organizing systems lend themselves to flexibility and adaptiveness
- *Where required:* in environments that are dynamic, persistent, multi-agent, decentralized, and self-organizing.

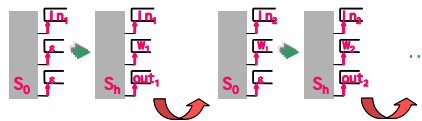
***Decentralized system:*** a multi-agent system whose components do not respond to commands from an active director or manager component, and do not execute prespecified synchronized roles under a design or plan.

***Self-organizing system:*** a multi-agent system with a coherent global structure or pattern shaped by local interactions among components, rather than by external forces.

## 4. Models

## Persistent Turing Machines

- A minimal extension of TMs expressing *sequential* interactive behavior (Goldin, *et al*)
- A *PTM* is a 3-tape TM with
  - I/O as dynamically generated *streams of interleaved inputs and outputs*
  - TM executions (*macrosteps*) iterated
  - A persistent worktape, called a *memory*, preserved between macrosteps



- *Example:* automatic car

## Stream behavior of PTMs

- The *persistent stream language* (PSL) of a PTM is the set of streams  $L \subseteq (\Sigma^* \times \Sigma^*)^\infty$  observable on it
- The set of all I/O streams over alphabet  $\Sigma$ :  
 $(\Sigma^* \times \Sigma^*)^\infty = \{ (a, x) \mid a \in (\Sigma^* \times \Sigma^*), x \in (\Sigma^* \times \Sigma^*)^\infty \}$
- PSL is the set of all persistent stream languages
- *Amnesic* PTMs do not make use of their memory, i.e., are equivalent to TMs in that sense
- ASL: The set of *amnesic* stream languages
- *Theorem:  $ASL \subset PSL$*  (Goldin, Smolka et al, 2004), hence **PTMs are more expressive than TMs**

## Formal specification of problems

- Algorithmic problem: a set of (*input*, *output*) string pairs
- Sequential-interaction problem: a set of dynamically generated *streams* of I/O pairs
- Multi-stream interaction problem: a set of possibly asynchronous I/O streams, possibly in real time and with dynamic creation/destruction of connections
- Part of spec for multi-stream interaction problem may include number of streams, constraints on computing power of agents, and time constraints

## Autonomous agents and models of sequential interaction

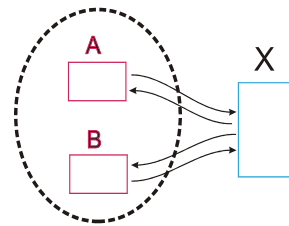
- Unified Modeling Language (UML) models sequential-interactive systems not supported by algorithm-based notations like flowcharts, module hierarchies, pseudocode
- *Autonomous agents* may initiate actions and may or may not synchronize with their environments
- To model autonomous agents, standard UML must be extended (Bauer, Muller, Odell, 2000)

## The message-passing model of concurrency

- Due to Robin Milner: CCS,  $\pi$  Calculus; associated with theory of concurrency and with process algebra
- These models capture the notion of *direct interaction by message passing*
- Axiom of concurrency theory:  
*interaction = message passing*  
i.e., atomic communication of a *message* from one *process* to another (targeted send/receive)
- Shared variables are deemed *processes*

## Limitations of the message-passing model

- Message passing does not support properties of indirect interaction: anonymity, asynchrony, space decoupling, non-intentionality, and late binding
- Embedded and situated systems aren't supported
- Suppose agents  $A$  and  $B$  communicate via shared variable  $X$ 
  - The message-passing model accounts for *direct*  $A \leftrightarrow X$  and  $B \leftrightarrow X$  interaction .
  - ...but not between  $A$  and  $B$  via  $X$



## Research goals

- We seek formal results to establish some limitations of the message-passing model
- We seek an *expressiveness result* analogous to the one for sequential interaction by Goldin-Smolka et al.
- *Setting*: A large system of simple agents
- We propose to use three proof approaches:
  - Unscalability
  - Formal behavioral specifications
  - Simulation asymmetry

## Showing unscalability of message passing

- *Motivation*: As unscalable architectures in AI are *brittle* and will fail in realistic settings (R. Brooks), so for unscalable MAS architectures and models
- *Hypothesis*: As the number of agents rises asymptotically, either number of connections grows too fast, or else paths between agents become too long
- Other dimensions to show unscalability:
  - Synchronization vs. asynchrony
  - Centralized vs. decentralized storage

## Scalability in parallel systems

- A scalable pair, (*parallel system*, *parallel algorithm*), is one in which speedup is roughly linear in number of processors (Gustafson's Law)
- This requires small *serial fraction* (fraction of unparallelizable steps in algorithm); many algorithms have significant serial fraction
- *Speedup*: serial time / parallel time
- *Efficiency*: speedup / # processors
- *Isoefficiency*: a metric of scalability, the ratio of problem size to minimum # processors  $p$  needed to obtain an increase in speedup proportional to  $p$

## Scalability in distributed systems

- *Communication time* is a more significant factor in determining scalability of distributed systems
- *Quality of service* must be maintained for many data streams in a scalable system
- By one definition, scalable systems are ones whose productivity (throughput times average value of response, divided by cost per second) is maintained as scale varies
- By another definition, relative scalability is proportion of power-cost ratios of two systems at different scales

## Scalability in multi-agent systems

- Notions of *autonomy* and *asynchrony*, as implied in the notion of *agents*, shape concept of scalability of MASs
- *Condition*: Each agent must provide a level of quality of service
- Models of scalability note *mesh* and *hierarchy* topologies, but not topologies made possible by *shared variables*
- Research notes that scalability is limited by any extra load that is due to increase in number of agents in the system

## Unscalability of message passing

- When the system's mission requires adaptability under conditions where each agent is simple, the massive communication load requires use of the agents' environment (indirect interaction)
- *Example:* Under message-passing assumption, all agents may communicate with all other agents, requiring  $O(n)$  memory overhead per agent and  $O(n^2)$  connections among agents, prohibitive where  $n$  is large

## Some notions of MAS scalability

- *Scalable MAS instance:* one that can perform a class of missions (hence satisfying their constraints) regardless of the number of agents  $nA$  or environmental entities  $nE$
- *Statically-scalable MAS* (w.r.t. a class of missions): one that is scalable under the assumption that agents and environmental entities are present at startup time
- *Dynamically-scalable MAS:* one that is scalable under the more rigorous assumption that agents and environmental entities may appear or disappear during execution

## Scalable architectures and models

- *Scalable MAS architecture* (w.r.t. a class of missions): a design architecture whose instances are all scalable w.r.t. that class of missions
- *Scalable computational model* of multi-stream interaction: one capable of serving as the formal foundation of MAS architectures and instances that are scalable w.r.t. nontrivial classes of missions

## Formal specification of problems that entail indirect interaction

- We propose to find a class of useful *missions or tasks* that would require indirect interaction
- *Setting*: A large system of simple agents
- *Initial idea*: to look at insect stigmergy examples – would tasks be impossible without stigmergy?
- If indirect interaction is *needed* to meet these specs, then an adequate model must represent that interaction explicitly
- *A tool*: specification languages and notations

## An asymmetric simulation relation

- ... exists between message-passing-based models and models based on indirect interaction
- *Motivation*: Simulation asymmetry would imply that current models are inadequate
- *Hypothesis*: Direct interaction *cannot* simulate indirect interaction in setting of large system of simple agents
- One possible simulation of direct interaction by indirect:
  - An agent puts a tuple into the shared environment
  - Tuple contains the both message and addresses
  - Recipient reads tuples that contain its ID

## Summary

- Sequential interaction is a variant of asynchronous communication; sequential interaction may be *direct or indirect via the environment*
- We present evidence that *multi-stream interaction* is richer than sequential interaction
- We define *missions* that can only be accomplished using indirect interaction (stigmergy)
- We have argued that for systems that accomplish these missions, message-passing models of interaction are inadequate because they are unscalable

## References

- Bauer, Muller, Odell, 2000.
- Eric Bonabeau, Marco Dorigo, Guy Theraulaz. *Swarm intelligence: From natural to artificial systems*. Oxford, 1999.
- Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.
- 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.
- Dina Goldin, Scott A. Smolka, Paul Attie, and Elaine Sonderegger. Turing Machines, Transition Systems, and Interaction. *Information and Computation* 194(2): 101-128, Nov. 2004.
- Dina Goldin and David Keil. Toward Domain-Independent Formalization of Indirect Interaction. *TAPOCS Workshop, Proc. WET ICE 04*, 2004.

## References (cont'd)

- David Keil and Dina Goldin. Modeling Indirect Interaction in Open Computational Systems. *TAPOCS Workshop, Proc. WET ICE 03*, 2003.
- David Keil and Dina Goldin. Indirect Interaction and Decentralized Coordination. *Extended draft*, 2004.
- David Keil and Dina Goldin. Adaptation and Evolution in Dynamic Persistent Environments. In *Proc. FInCo2005, Edinburgh*, 2005.
- David Keil and Dina Goldin. Modeling Indirect Interaction in Environments for Multi-Agent Systems In *Proc. E4MAS*, 2005.
- Robin Milner. *Communicating and Mobile Systems: The  $\pi$  Calculus*. Cambridge, 1999.
- Mitchel Resnick. *Turtles, Termites, and Traffic Jams*. MIT Press, 1994.
- Peter Wegner. Why interaction is more powerful than algorithms. *CACM* 40 (5), 1997.