

Modeling indirect interaction for evolving adaptive multi-agent systems in dynamic persistent environments

Doctoral thesis proposal

David M. Keil

April 11, 2006

Department of Computer Science and Engineering

University of Connecticut

Advisor: Dina Q Goldin

Three research communities

- Multi-agent systems (MAS)
- Evolutionary computation (EC)
- Models of computation

Subcommunities:

- Theory and Practice of Open Computational Systems (TAPOCS)
- Environments for Multi-Agent Systems (E4MAS)
- Foundations of Interactive Computation (FInCo)
- EC in dynamic environments
- Coordination

Common trends in EC and MAS research

- Trend toward addressing environments that are *dynamic and persistent* (to be defined)
- Trend toward using agents in MASs that communicate via their environments
- We call this communication via the environment *indirect interaction*
- The theory of these fields is emerging

A gap between practice and theory in MAS and EC research

- Whereas *in practice*, agents in MASs and EC often interact indirectly via their environments...
- ...*theory of concurrency* models all interaction as *direct message passing*
- *Gap*: Indirect interaction in practice, direct interaction in theory
- *Q*: Is indirect interaction *necessary* to solve certain classes of problems?
- *A* (our central hypothesis): *Yes*. Hence new, more expressive models are needed to close the gap

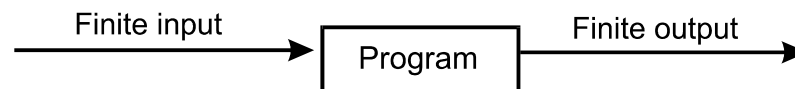
Outline

1. Relevant definitions
 - Algorithmic computation
 - Interactive computation
 - Multi-stream interaction
 - Direct vs. indirect interaction
2. Indirect interaction in MAS research
3. Indirect interaction and adaptation in EC
4. Formal models of interaction
5. Our research goals

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

Interactive computation

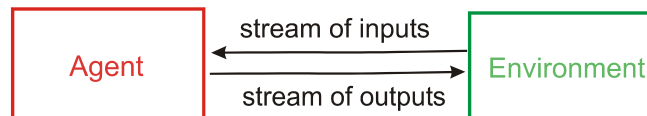
Interactive computation (Wegner):

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.

- Interaction involves *feedback from environment* during the computation
- Interaction is assumed to be unending
- *Example:* An automatic car driving from point *A* to point *B*

Sequential interaction

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

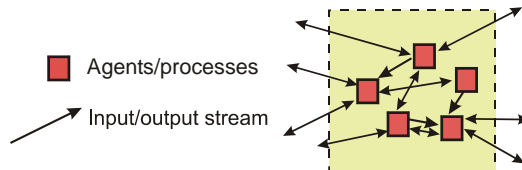


- Characterized by a single interaction stream of input/output; input alternates with output
- If one participant is an agent, the other is its *environment*
- Interaction may involve changes of state

Multi-stream interaction

Multi-stream interaction: Interactive computation involving *more* than two entities; the entities may be *asynchronous*.

In contrast to sequential interaction, multi-stream interaction may include:



- Nondeterminism when attempts to write collide
- Dynamic linking and unlinking, creation/destruction
- *Indirect* interaction via a shared environment

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.

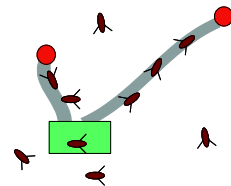
- Sequential interaction is direct
- Preconditions for indirect interaction:
 - Agents share access to parts of the environment
 - Persistence of environment
- *Example of indirect interaction:* use of semaphores in process synchronization (critical section problem)

Outline

1. Relevant definitions
 - Algorithmic computation
 - Interactive computation
 - Multi-stream interaction
 - Direct vs. indirect interaction
2. **Indirect interaction in nature and MAS research**
3. Indirect interaction and adaptation in EC
4. Formal models of interaction
5. Our research goals

Stigmergy in nature

1. **Ants foraging for food:** Ants leave pheromone trail, prefer existing trails, blaze shorter and shorter trails to and from food
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 tasks?

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

What is an environment?

An *environment* of a system of computing entities is a *physical* or *virtual* setting that acts as the producer of the system's inputs and consumer of its outputs.

- The environment is a participant and a memory, not just a medium for message transport
- This creates a need to elevate the MAS environment to first-class status
- EU conferences (e.g., E4MAS) have called attention to role of environments

Environments for multi-agent systems

E4MAS 2005 Proceedings cited as examples the environments of:

- visitors to a web site;
- a system of autonomous guided vehicles;
- a system of manufacturing control;
- a PDA-based system of agents to help support activities of museum visitors.

All involve indirect interaction

A taxonomy of environments

Amnesic vs.	Static vs.	Virtual vs.
Persistent	Dynamic	Physical
<ul style="list-style-type: none"> • An environment is <i>amnesic</i> if its outputs depend only on its <i>immediately preceding</i> inputs • An environment <i>E</i> is <i>static</i> with respect to an agent or MAS <i>A</i> if its outputs to <i>A</i> are <i>strictly dependent</i> on its previous inputs from <i>A</i> • A <i>virtual</i> environment is accessed digitally; a <i>physical</i> environment is observable only by analog sensors. 		

Adaptation in difficult environments

- The most difficult problem environments are *persistent*, *dynamic*, and *physical*
- MASs can offer powerful *adaptive*, *flexible* solutions in such environments
- *Conjecture*: Indirect interaction provides added power in MAS solutions because of anonymity, asynchrony, space decoupling, non-intentionality

Adaptation and multi-agent systems

- MASs enable *distributed AI* (Ferber)
- *Behavior*: action to change the environment
- *Adaptation*: learning that changes behavior – occurs in dynamic persistent environments
- MASs are often flexible enough to adapt well
- Three ways to view adaptation:
 - Ontogenetic (adaptive agent)
 - Sociogenetic (adaptive population)
 - Phylogenetic (adaptation by species)
- *Sociogenetic adaptation* = adaptation by multi-agent systems

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.

Outline

1. Relevant definitions
 - Algorithmic computation
 - Interactive computation
 - Multi-stream interaction
 - Direct vs. indirect interaction
2. Indirect interaction in nature and MAS research
- 3. Indirect interaction and adaptation in EC**
4. Formal models of interaction
5. Our research goals

The evolutionary algorithm

- A *population-based* approach to function optimization
- Solutions are evolved, using *selection, mutation, crossover*
- Traditional EC uses *objective (fitness) function* to evaluate an element of a population

```

t ← 0           // time
initialize (P0) // evolving population
y ← evaluate (P0) // fitnesses of population members
while not terminate (y, t) do
  t ← t + 1
  Pt ← select (Pt-1, y) // choose a good new generation
  Pt ← alter (Pt) // involves mutation, crossover
  y ← evaluate (Pt) // generates a vector of fitnesses

```

Based on (Michalewicz, 1996)

- The evolution occurs *offline*, not embedded in environment

Example: checkers heuristics

- A set of checkers-playing heuristics (weights of attributes of a board layout), is evolved (Samuels '59)
- Fitness: rate of wins that a set of heuristics obtains
- Population P consists of sets of weights (values) of different attributes of a *checkers board configuration*
- E.g., opportunity to jump is of weight 5, opportunity to king is 3, etc.
- EC here refines heuristics that help compute a function from board configurations to (good) moves
- Fitness function is applied by putting heuristics in competition with other heuristics

EC has addressed *static* environments

- Environment is *static* in the checkers example because the game rules don't change during evolution
- *Static environment* =
single (unchanging) fitness function
- In a *dynamic* environment, fitness or reward will *change* as the environment changes
- An interactive agent in a changing environment must adapt its response as environment changes
- **Single fitness function in EC \Rightarrow
Environment cannot be dynamic**

Policy in a dynamic environment

- When environment changes *policy* must evolve; *policy search* is a reinforcement-learning concept
- A rational policy: one that maximizes reward

The *policy* of agent M , with respect to environment E , is a computable function from possible perceptions, or models, of E , to M 's set of outputs.

The *fitness of a policy in environment E* , is the expected long-term reward in E of an agent with that policy.

- In dynamic environment, reward function evolves
- Policy must change as the environment's responses to agent change; policy search is online

Dynamic-environment example

- Suppose we play *cat-and-mouse* on a grid
- *Agent* is mouse; *Environment* is cat and grid
- Goals of mouse policy: escape by fleeing or hiding
- Assume mouse policy is to be evolved; fitness function is survival rate of a policy
- If cat speeds up over time, then mouse policy must switch from *flee* to *hide*
- Traditional evolutionary algorithm *fails* here because it assumes static environment

The evolutionary algorithm revisited

- When environment E is dynamic, EA must be parameterized with it

```

t ← 0
initialize (P0, E0)
y ← evaluate (P0, E0)
while not terminate (y, t) do
  t ← t + 1
  Et ← update-environment (Et-1, y)
  Pt ← select (Pt-1, y)
  Pt ← alter (Pt)
  y ← evaluate (Pt, Et)

```

- Goal is to evolve solution population P to better fitness relative to changing environment
- If *update-environment* is autonomous, then evolution of the population is not an algorithm!

No Free Lunch theorem (1996)

- No algorithmic procedure can optimize cost functions better than any other algorithmic procedure, averaged over all cost functions.

$$\sum_f P(\vec{c} \mid f, m, a_1) = \sum_f P(\vec{c} \mid f, m, a_2)$$

- c = histogram of cost function f ;
 a_1, a_2 : arbitrary function-optimization algorithms
 P = probability of histogram
 m = a sample size
- NFLT corollary*: If a given optimizing algorithm does well on one problem, it will do poorly on another one
- Result*: Human domain knowledge is needed for most evolutionary computation

Resolving the NFLT paradox

- *Paradox*: Whereas by NFLT good general-purpose problem-solving *algorithms* can't exist...
- ...still, such *processes* are known to exist, such as natural evolution of life, and the scientific method
- *Solution*: NFLT applies to *algorithms* in *static* environments; does not apply to interactive learning processes occurring in dynamic persistent environments
- *Adaptation* to environments (learning of policies) is interactive, not algorithmic

Multi-agent interaction in EC research

The two research areas (MAS and EC) intersect in research on:

- *Swarm or ant computing*
- *Coevolution*: Evolution of species whose instances interact in multi-agent systems
- *Particle swarm optimization*: Particles are candidate solutions to a problem in n -dimensional space, particles are accelerated through this space in relation to each other and to objective function

Outline

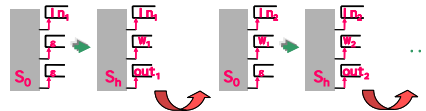
1. Relevant definitions
 - Algorithmic computation
 - Interactive computation
 - Multi-stream interaction
 - Direct vs. indirect interaction
2. Indirect interaction in nature and MAS research
3. Indirect interaction and adaptation in EC
- 4. Formal models of interaction**
5. Our research goals

Contributions to the theory of interactive computing

- *c*-machine (Turing), finite transducer (Moore)
- Cybernetics: models of feedback systems (Wiener)
- Information theory/communication theory (Shannon)
- Concurrency with message passing: CSP (Hoare), CCS (Milner), π calculus (Milner)
- Recent models of sequential interaction:
I/O Automata (Lynch), Abstract State Machines (Gurevich), Site Machines (van Leeuwen, Wiedermann)
- Interaction Machines and Persistent Turing Machine (Wegner, Goldin)
- *Emerging intuition*: Interaction is part of computation

Persistent Turing Machines

- A minimal extension of TMs expressing *sequential* interactive behavior (Goldin, *I&C*)
- 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

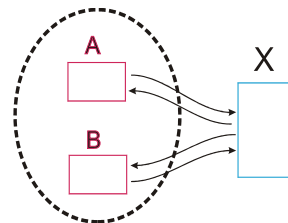
- The *persistent stream language* (PSL) of a PTM is the set of streams $L \subseteq (\Sigma^* \times \Sigma^*)^\infty$ observable on it
- The set of all I/O streams over alphabet Σ :
 $(\Sigma^* \times \Sigma^*)^\infty = \{ (a, x) \mid a \in (\Sigma^* \times \Sigma^*), x \in (\Sigma^* \times \Sigma^*)^\infty \}$
- **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, *I&C*, 2004), hence **PTMs are more expressive than TMs**

The message-passing model of concurrency

- Due to Robin Milner: CCS, π 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*



Outline

1. Relevant definitions
 - Algorithmic computation
 - Interactive computation
 - Multi-stream interaction
 - Direct vs. indirect interaction
2. Indirect interaction in nature and MAS research
3. Indirect interaction and adaptation in EC
4. Formal models of interaction
5. **Our research goals**

Research goals

- We propose to obtain 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:
 - Formal behavioral specifications
 - Unscalability
 - Simulation asymmetry

Goal: 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

Goal: to show 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

Goal: to show an asymmetric simulation relation

- ... 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

1. Common trends in EC and MAS research
2. A *gap* separates the practice and the theory of these fields
3. NFL Theorem does not apply in dynamic environments
4. *Properties* enabled by indirect interaction: *anonymity*, *asynchrony*, *non-intentionality* – models must support them
5. *Goal:* Expressiveness results showing the need for explicit models of indirect interaction;
6. *Approaches:*
 - show *behavioral specifications* that entail indir. inter.
 - show *unscalability* of message-passing models
 - show an *asymmetric simulation relation* between models of message-passing and indirect interaction.

References

- Dina Goldin and David Keil. Evolution, Interaction, and Intelligence. In *Proc., Congress on Evolutionary Computation (CEC-2001), Seoul, Korea, 2001*.
- Dina Goldin and David Keil. Toward Domain-Independent Formalization of Indirect Interaction. *TAPOCS Workshop, Proc. WET ICE 04, 2004*.
- 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, <http://www.cse.uconn.edu/dqg/papers/indirect.pdf>*.
- 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*.

References (cont'd)

- Rodney A. Brooks. Intelligence without reason. *MIT A.I. Memo*. www.ai.mit.edu/people/brooks, 1991.
- 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 Journal* 194(2): 101-128, Nov. 2004.
- Dina Goldin and Peter Wegner. The Church-Turing Thesis: Breaking the Myth. Presented at CiE 2005, Amsterdam, June 2005. LNCS 3526, Springer 2005, pp. 152-168
- Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*, 3rd Ed. Springer, 1996.
- Arthur Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research & Development*, (3):211-229, 1959.
- Starlogo site at MIT Media Lab. <http://starlogo.www.media.mit.edu/people/starlogo>.
- David H. Wolpert and William G. Macready. No free lunch theorems for search. *Santa Fe Institute technical report SFI-TR-0, 10, 1996*.

Evolution and distributed AI

Old AI/EC Find a computational path Reasoning in a closed world Human intervention Heuristics Toy environment Brittle solutions	New Learn a policy AI/EC Rational reactive behavior Autonomy Emergent behavior Real-world environment Scalable solutions
---	--

- AI has moved away from systems that reason in a closed algorithmic world, toward rational-agent behavior
- The same can happen with EC and MAS as they converge
- AI is more and more associated with distributed and MASs
- A hierarchy of difficulties of interactive problems:

