Topic 8: Algorithmic vs. interactive computation

- Algorithmic computation and the paradigm shift to interaction
- Indirect and multi-stream interaction

Algorithmic computation

- *Algorithm*: an old mathematical concept (Al Kuwarizmi, Baghdad, ca. 800 A.D.)
- *Turing machines*: identified with the notion of algorithms (Turing, 1936)
- Features of algorithmic computation: finite input, followed by finite computation, followed by finite output

- “The classical TM paradigm may no longer be fully appropriate to capture all the features of present-day computing” (Van Leeuwen-Wiedermann, 2000)
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
Interactive computation

- Feature of computing today: Computation as an ongoing service, not assumed to terminate
- Must solve problems whose inputs cannot be completely specified a priori
- Dynamic input and output during computation
- Persistence of state between interaction steps
- Environment is an active partner in computation

Driving is nonalgorithmic

- The problem of driving a car is interactive, not reducible to an algorithm
- Given input of a map, algorithmic approach would compute a series of outputs to steering wheel, gas, brake, etc.
- No one could drive anywhere this way
- External conditions can affect car’s motion during driving
The paradigm shift to interaction

<table>
<thead>
<tr>
<th>Algorithmic</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transforming input to output (by TMs)</td>
<td>Providing a service over time (by agents)</td>
</tr>
<tr>
<td>Structured design</td>
<td>Object-oriented design</td>
</tr>
<tr>
<td>Logic and search in AI</td>
<td>Agent-oriented AI</td>
</tr>
<tr>
<td>Rule-based reasoning</td>
<td>Adaptation, control</td>
</tr>
<tr>
<td>Compositional behavior</td>
<td>Emergent behavior</td>
</tr>
<tr>
<td>Closed systems</td>
<td>Open systems</td>
</tr>
</tbody>
</table>

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.
**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

  ![Diagram of PTM]

- *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:* \( \text{ASL} \subset \text{PSL} \) (Goldin, Smolka *et al*, 2004), hence PTMs are more expressive than TMs
**Formal specification of problems**

- Algorithmic problem: a set of \((\text{input, output})\) string pairs
- Sequential-interaction problem: a set of dynamically generated \(\text{streams}\) 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)
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
Agents, environments, and persistent state

- Environment, like interacting agents, has persistent state (memory)
- Environment-agent relation features symmetry
- Indirect interaction uses persistent state of environment as medium
- When environment is passive (relays inputs between agents, unaltered), it is medium of pure indirect interaction

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)
Some indirect interaction patterns in nature

1. Termites gathering chips
   Protocol: Move at random, pick up chip when encountered, put down when another found

2. Ants foraging for food
   Ants leave chemical trail, prefer existing trails, blaze shorter and shorter trails to and from food

3. Slime mold dividing and aggregating
   These amoeba may aggregate by emitting chemical, migrating toward its greatest concentration

Self-organization and emergent behavior

- **Definition**: Self-organization is the interaction of a set of processes or structures at a lower level of a system to yield global structures or behavior at a higher level
- **Example**: Chemical reactions
- **Contrast to**: Centralized, algorithmic behavior
- System behavior that is not the sum of component behaviors is emergent
Stigmergy

• **Definition**: A variety of self-organization in which mobile agents interact via the environment

• **Contrast to**: direct interaction; centralized interaction

• **Examples**:
  – termites gathering chips,
  – ants foraging,
  – slime mold aggregation

Overcoming limitations of algorithmic problem solving

• **Internal** and **indirect** interaction can support general-purpose adaptive behavior capable of attaining fitness in arbitrary dynamic persistent environments

• **Evolutionary** techniques appear to be the main way forward, provided the evolution occurs online, as the evolved objects interact with their environment
Why identify self-organization and stigmergy with indirect interaction?

- Whereas *centralized* systems may employ a *star* structure:

  ![Diagram: Star Structure]

- … *self-organizing* systems require a *network* structure, in which remote agents can only interact indirectly, via local direct exchanges.

Indirect interaction and multiagent systems

- In a MAS characterized by locality of interaction and mobility of agents, it is only possible for agents to influence overall system behavior remotely, i.e., indirectly.
- Richness of multiagent interaction:
  - due partly to ability of each agent to interact with multiple others
  - hence 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.

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**: $\text{ASL} \subset \text{PSL}$ (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)

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, \((\text{parallel system, parallel algorithm})\), 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 $n_A$ or environmental entities $n_E$
- *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
Complexity of interactive systems and problems

• [See 12/06 defns]

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
Conclusion

- What is computation?
- What is a computational problem?
- Problem: algorithms $\rightarrow$ tasks
- Computation: closed box $\rightarrow$ open system working concurrently with environment (Peter Wegner, CACM, 5/97)

References


References

Bauer, Muller, Odell, 2000.


References (cont’d)


