

Toward a theory of interactive computation

David Keil
*University of Connecticut
 and Framingham State College*

Joint work with Dina Goldin,
 University of Connecticut

David Keil NES/MAA meeting, 11/22/02 1

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)

David Keil NES/MAA meeting, 11/22/02 2

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

David Keil NES/MAA meeting, 11/22/02 3

Overview

- Algorithmic computation and the paradigm shift to interaction
- Models of algorithmic computation
 - Deterministic finite automata
 - Turing machines
- Models of interaction
 - Finite transducers
 - Persistent Turing Machines, Interactive Transition Systems
- Streams and coinduction
- State equivalence and minimality

David Keil NES/MAA meeting, 11/22/02 4

The paradigm shift to interaction

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

What about theory of computation?

David Keil NES/MAA meeting, 11/22/02 5

Finite automata, finite transducers

- *Deterministic finite automaton*: $\langle Q, \Sigma, \delta, q_0, F \rangle$ where Q is a finite set of states, Σ an input alphabet, $\delta : Q \times \Sigma \rightarrow Q$ a transition function, q_0 a start state, F a set of accepting states
- DFAs are *recognizers* of sets of *finite* sequences
- *Mealy machine*: $\langle Q, \Sigma, \Gamma, \delta, q_0 \rangle$ where Γ is a finite *output* alphabet, $\delta : Q \times \Sigma \rightarrow Q \times \Gamma$ is a transition function
- These models were both well-known through the first standard CS curriculum (ACM, 1968)

David Keil NES/MAA meeting, 11/22/02 6

Stream I/O

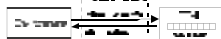
- Mealy machines model finite non-halting *interactive devices*: e.g., integrated circuits, control devices
- A Mealy machine computes a function $\Sigma^\infty \rightarrow \Gamma^\infty$ where Σ^∞ is the set of *streams over Σ* :
 $\Sigma^\infty = \{ ax \mid a \in \Sigma, x \in \Sigma^\infty \}$
- This is a *coinductive* definition, associated with non-well-founded set theory
- Language of a Mealy machine: $L(M) \subseteq (\Sigma \times \Gamma)^\infty$

Algorithmic model: Turing machines

- *Turing machine*:
 $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$
 where Σ and Γ are *input* and *tape* alphabets (Sipser)
- *Transition function* $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ where L, R denote left or right moves on the tape
- *Configuration*: 3-tuple consisting of state, tape contents, head location
- Infinitely many configurations exist
- Tape is erased between computations
- TMs compute the set of recursively definable functions on natural numbers or, alternatively, strings

Interactive model: Persistent Turing Machines

- A *Persistent Turing Machine* is a 3-tape TM with a worktape that is preserved between interactions
- Inputs and outputs are dynamically generated streams
- A minimal extension of TMs expressing interactive behavior
- One *macrostep* of a PTM is a TM computation consisting of *microsteps*
- PTM's persistent work tape is called its *memory*



PTM example: answering machine

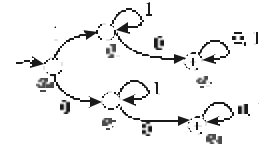
- An *answering machine A* is a PTM whose worktape contains a sequence of recorded messages and whose operations are *record message*, *playback*, and *erase*.
- Its TM-computable function f_A is:
 $f_A(\text{record } Y, X) = (\text{ok}, XY)$
 $f_A(\text{playback}, X) = (X, X)$
 $f_A(\text{erase}, X) = (\text{done}, \epsilon)$
- For input stream (record m_1 , erase, record $m_2 m_3$), A generates (ok, done, ok, ok, $m_2 m_3$)
 (Goldin, 1999)

Stream language and equivalence

- Interactive stream language (*ISL*) of a PTM M : the set of interaction streams $(i_1, o_1), (i_2, o_2), \dots, (i_n, o_n) \in \Sigma^*$, in which for every k , there are memories w, w' such that $f_M(w, i_k) = (o_k, w')$
- Two PTMs are *stream equivalent (observationally equivalent)* iff $ISL(M_1) = ISL(M_2)$
- Two memories of M , w_1 and w_2 are equivalent iff sub-PTMs with w_1 and w_2 as starting memories have the same ISL
- Other equivalences:
 - bisimilarity
 - isomorphism

Notions of minimality

1. A finite-state machine is *minimal* if it has the fewest number of states over all equivalent machines
 2. A machine is minimal if it has no two distinct equivalent states
- *Example*: states q_1, q_2 are indistinguishable in this non-minimal DFA, where $L(M) = (0|1)1^*0(0|1)^*$.



Equivalence and minimality for PTMs

- Three increasingly refined forms of equivalence: stream, bisimulation, isomorphism
- For stream-based computing, two states are stream-equivalent iff the same I/O streams may be observed from both states
- For equivalence relation R , define R -minimal ITS as one with no two R -equivalent states

David Keil NES/MAA meeting, 11/22/02 13

Isomorphism of minimal equivalent interaction machines

- Given equivalence relation R , construct the R -minimal PTM for a given PTM, using state equivalence classes as the reduced version's state set
- Homomorphism exists from any deterministic PTM A to any minimal equivalent PTM A_0
- Result: any two stream-minimal stream-equivalent PTMs are isomorphic

David Keil NES/MAA meeting, 11/22/02 14

Conclusion

- What is computation?
- What is a computational problem?
- Problem: algorithms \rightarrow tasks
- Computation: close box \rightarrow open system working concurrently with environment (Peter Wegner, *CACM*, 5/97)
- Here we have discussed *sequential* computation
- Future work: Multiagent computation

David Keil NES/MAA meeting, 11/22/02 15

OVERFLOW

David Keil NES/MAA meeting, 11/22/02 16

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

David Keil NES/MAA meeting, 11/22/02 17

Inductively defined sets

- (i) 0 is a natural number (initiality);
- (ii) Every natural number n has a unique successor, n' (iteration)
- (iii) i and ii are the only ways to obtain a natural number (minimality)
- Example: Expressions using numerals, +, and ():
 - expression* :
 - numeral*
 - numeral + expression*
 - (expression)*
- Inductive definitions may conditionally use selves

David Keil NES/MAA meeting, 11/22/02 18

Streams are defined *coinductively*

- Whereas induction is characterized by use of *least* fixed point semantics, coinductive definitions use *greatest* fixed point
- Iteration condition:
 $\Sigma^\infty = \{ ax \mid a \in \Sigma, x \in \Sigma^\infty \}$
- Maximality condition (GFP)
- No base case
- Induction models process of construction of finite objects
- Coinduction models *open* processes yielding infinite objects, e.g., interaction

Stream sets are non well founded

- Foundation Axiom excludes notion of sets belonging to themselves
- A non-well-founded set may contain itself, directly or indirectly (Anti-Foundation Axiom)
- Example:* $A = \{ B, C \}; B = \{ A, D \}$
- Every stream of characters is a character, followed by a stream of characters
- Streams contain streams, which contain streams, which contain streams, ...
- Early work: Paul Finsler, 1920s; see also P. Aczel, 1988; J. Barwise, L. Moss, 1996

Can theory make the shift?

- Needed:* a model of interaction for TM-powerful devices, analogous to the transducer model for DFA-powerful devices
- Note:* Far from using the transducer model for inspiration, current theory tends to drop all mention of it

Interactive transition systems

- An *interactive transition system* (ITS) is a quadruple $\langle S, \Sigma, m, r \rangle$ where $S \subseteq \Sigma^*$ is the set of *states*; Σ is a *finite alphabet*; $m \in S \times \Sigma^* \times S \times \Sigma^*$ is the *transition relation*; $r \in S$ is the *initial state* (root)
- m is required to be *recursive* (computable), i.e., its interpretation as the function $m : S \times \Sigma^* \rightarrow 2^{S \times \Sigma^*}$ is computable
- ITS model is equivalent in expressiveness to PTM model, i.e., infinite-state transducers
- Stream languages of PTMs, ITSs with persistent memory are a strict superset of those that discard worktape between steps

Reduced form and indistinguishability

- Let M be a deterministic finite automaton
- Then its *reduced form* is a DFA constructed by merging *indistinguishable states* of M .
- States q_1 and q_2 are *indistinguishable* if on any input x to the two sub-DFAs that start with these states, x is accepted by both or rejected by both.

Homomorphism from any ITS to minimal equivalent ITS



- Proposition:* If $A_0 \equiv A_1$ and both are minimal, then an isomorphism exists between them
- Proof sketch:* Sets of equivalent states of A_1 show same behavior, define equivalence classes that are the state set of A_0 . Mapping of states of A_1 to equivalent states of A_0 is homomorphic. If both machines are minimal, reverse mapping is possible, hence isomorphism exists.