Persistence as a Form of Interaction

D. Goldin and P. Wegner

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-98-07
July 1998

Persistence as a Form of Interaction

D. Goldin
Univ. of Massachusetts - Boston
dqg@cs.umb.edu

P. Wegner
Brown University
pw@cs.brown.edu

July 14, 1998

Abstract

This paper establishes a relation between *interaction* and *persistence*. Interactive computation is characterized by Interaction Machines (IMs) [We1], with a restricted subclass of *Sequential Interaction Machines* (SIMs) controlled by a single sequential stream of interactions, modeling transducers and simple data abstractions. *Persistent Turing Machines* (PTMs) are introduced by extending Turing Machines (TMs) with a persistent working tape. It is shown that SIMs and PTMs are expressively equivalent but more expressive than TMs where, following Milner [Mi2], *expressiveness* is defined in terms of observerability of system behavior. This work is part of a larger effort towards a more comprehensive formalization of interactive computation, with the aim of characterizing empirical computation by interaction.

1 Introduction

In this paper, we adopt Milner's *interactive view* of system behavior [Mi1]. This view is more general than the *algorithmic view* [Kn, Pa], extending the one-input one-output computational model of Turing Machines (TMs) to computations that involve sequences of interactions. It is interesting to note that the distinction between *automatic machines* and *choice machines*, where the latter allow choices by an external operator, was already present in Turing's seminal paper [Tu], but was not followed up by Turing.

Whereas algorithmic expressiveness is defined by the transformational power of systems, interactive expressiveness is defined by the ability of observers to make observational distinctions. In [MP], it was shown that reactive (interactive) systems cannot be modeled by TMs. However, interactive behavior specifications can easily be specialized from multiple to single interactions. Thus, observational expressiveness provides a common framework for expressing the behavior of algorithmic and interactive systems within which they can be compared.

In the interactive framework, whether two systems are equivalent depends not only on the systems but on the mode of observation. Observations may include multiple interactions with the system, with one of several pragmatics [GW]:

on-line vs. *off-line*, leading to distinctions known in process theory as *linear time* vs. *branching time* equivalence [Pn]. The Brock-Ackerman anomaly [BA] is a classic example of how these modes of observation affect the observer's ability to distinguish between systems.

black-box vs. *glass-box*, known in process theory as as *trace equivalence* vs. different types of *simulation.* The equivalence classes of system behavior induced by different glass-box observational modes for labeled transition systems are well studied [Gl]

The transformational view of *system expressiveness*, where some classes of automata are more powerful than others due to their ability to express a wider set of functions, generalizes to the interactive framework as well.

In this paper, we introduce *Persistent Turing Machines* (PTMs) that extend TMs with a persistent working tape whose content is preserved for successive interactions. From the viewpoint of a system observer, systems such as PTMs that preserve the state between interactions are more natural than TMs that reinitialize their state for every interaction. Persistence is necessary to model history dependent behavior, implying that non-persistent systems are necessarily limited in their usefulness. PTMs can model data abstractions like stacks and queues, and we show that they are equivalent to *Sequential Interaction Machines* (SIMs), a subclass of IMs whose behavior is characterized by a single sequential stream of interactions.

The characterization of sequential interaction by persistence provides insight into nature of interaction, showing that interaction of a client agent with a server agent can be specified in terms of persistence of the server agent. Conversely the characterization of persistence by interaction provides the insight that persistence can be specified entirely in terms of how persistent agents interact with their observers.

In this paper, we adopt the *black-box* approach to observation, where system inputs and outputs are the only aspects of its behavior available to be observed. Furthermore, we assume that system observations consist of sequences of input-output pairs, akin to multiple successive calls to a function within a larger program. Though this is more limiting than the general observational frameworks studied elsewhere, it is sufficient to obtain our results.

Overview: In Section 2, we present the fundamental concepts for use throughout the rest of the paper. In Section 3, we define TMs and the semantics of their computations, followed by a similar treatment for PTMs in Section 4. We discuss the expressiveness of PTMs in Section 5, and compare it to the expressiveness of TMs in Section 6. In Section 7, we define SIMs and the semantics of their computations, and establish their equivalence to PTMs in Section 8. We conclude with a brief summary in Section 9.

2 Fundamental Concepts

In this section, we introduce concepts and definitions used throughout the rest of the paper.

2.1 System Equivalence and Distinguishability

Throughout the paper, we assume that system observation is *black-box* (only inputs and outputs are observed) and *sequential* (observations consist of sequences of inputs and outputs). Therefore, the observable system behavior can be characterized by a set of finite *interaction sequences*, defined as follows:

Definition 2.1 *Given a system S, the* observable behavior *of S (denoted by $\mathcal{B}(S)$) is a set of all finite interaction sequences observable for S, where an* interaction sequence of length *k is a a finite sequence of pairs of input-output values:*

$\{(i_1, o_1), (i_2, o_2), ...(i_k, o_k)\}$,

where each i is an input to a system, each o is the corresponding output, and each i_j is input after o_{j-1} is output but before o_j is output.

Two systems S_1 and S_2 are said to be *equivalent* if $\mathcal{B}(S_1) = \mathcal{B}(S_2)$. Likewise, they are said to be *distinguishable* if $\mathcal{B}(S_1) \neq \mathcal{B}(S_2)$. In this case, any interaction sequence that is observable for one of them but not for the other is known as a *distinguishability certificate* (SDC). Though there is no unique SDC for any pair of systems, the length of the shortest SDC is well-defined.

Definition 2.2 *Given a pair of systems S_1, S_2, any element in*

$$(\mathcal{B}(S_1) \cup \mathcal{B}(S_2)) - (\mathcal{B}(S_1) \cap \mathcal{B}(S_2))$$

is known as a distinguishability certificate *(SDC) for (S_1, S_2). The length of the shortest SDC for (S_1, S_2) is denoted by $dist(S_1, S_2)$.*

This notion of *dist* allows us to define finer classes of system equivalence, called *k-equivalence* for any k:

Definition 2.3 *For any natural k, systems S_1 and S_2 are* k-equivalent *if $dist(S_1, S_2) > k$. Conversely, S_1 and S_2 are* k-distinguishable *if $dist(S_1, S_2) \leq k$.*

It is easy to see that k-equivalence is transitive, reflexive and symmetric, thus inducing, for any system S and any k, the equivalence class of all systems k-equivalent to S:

Definition 2.4 *For any k, the* k-equivalence partitioning *of PTMs is the partitioning of PTMs into subsets such that two PTMs $\mathcal{M}_1$ and $\mathcal{M}_2$ belong to the same subset iff they are k-equivalent.*

2.2 Persistence and History Dependence

In this paper, we only consider system behavior that is observable by observers interacting with the system. We define interaction as:

Definition 2.5 (Interaction) *A system S_1 is said to* interact *with a system S_2 if S_1's output serves as S_2's input, or vice versa.*

The sequence of interactions of a system with observers is called its *interaction history*. In order to apply this notion, there must also be a notion of the *current interaction* which is not considered a part of its history:

Definition 2.6 (History dependence) *The behavior of a system S is said to be* history dependent *if S's interaction history may affect its behavior for the current interaction.*

Persistence can be viewed as the system's ability to remember while not performing computations, such as persistence of data in database systems.

Definition 2.7 (Persistence) *A system S is said to be* persistent *relative to observers with which it may interact if it continues to exist for multiple interactions of observers with S.*

Persistence is a necessary (though not sufficient) condition for systems to be history dependent. History dependence is an important property of intelligent system behavior, implying that non-persistent systems are necessarily limited in their usefulness.

3 Turing Machines

Turing Machines (TMs) are an example of systems whose behavior is neither history dependent nor persistent; computations of a TM do not depend on previous computations. In this section we define TMs and the semantics of their computations; later, we will define Persistent Turing Machines as TMs with a persistent working tape.

Definition 3.1 *A* Turing Machine *(TM) is a quintuple $\mathcal{M} = (S, \Sigma, \Delta, S_0, S_F)$, where S is the state set, Σ is the input alphabet, Δ is the state transition relation, S_0 is the initial state, and F is the set of final states. A Turing Machine has one infinite tape for performing input/output.*

Turing Machines perform *computations* by being given a finite input string, with the reading/writing head positioned at the beginning of the tape, and performing a sequence of state transitions to a final state. Whatever is on the tape at the end is considered output.

Definition 3.2 *A* TM computation *is a sequence of transitions allowed by the machine's state transition relation, starting from the initial state, with input string I on the tape, ending in a final state, with output string O on the tape. Each transition consists of moving the I/O head by one position, reading/writing one character at current position, and changing its internal state.*

Semantics of TM computations: Is is well known that a TM $\mathcal{M}$ defines a partial recursive function $f_{\mathcal{M}}$ [LP]:

$$f_{\mathcal{M}}(\text{input}) = \text{output}.$$

Turing machines can be generalized to multiple-tape TMs (MTMs), with a read-only input tape, an output tape, and a finite number of work tapes. It is well known that MTMs with any finite number of tapes are no more expressive than TMs [LP].

Definition 3.3 Multitape Turing Machines *(MTMs) perform computations by being given a finite input string on the input tape, with the other tapes being blank, and with the reading/writing heads positioned at the beginning of the tapes. Whatever is on the output tape at the end is considered output.*

The definition of a TM computation can also be generalized to permit multiple computations for different input tapes, just as a function in a program can be called multiple times.

Definition 3.4 *An* extended computation *of a TM is a sequence of computations for a sequence of input tapes.*

TMs are abstract machines, and we conceptualize TM computation by instantiating a new TM from its description any time we need to compute with it. Herein lies the difference with the real computation devices, which are created once but are used over and over again. Though adding working tapes to a TM does not increase its power, we contend that making their contents persistent between computations does. We define such an extension to TMs in the next section.

4 Persistent Turing Machines

In this section, we define a *Persistent Turing Machine* (PTM) as a multitape Turing machine where the contents of the work tape(s) persists between Turing machine computations.

Definition 4.1 *A PTM is a multitape Turing machine with a persistent work tape that is initially empty; the contents of the work tape at the end of each computation is maintained for the beginning of the subsequent one.*

The notion of *computation* for a PTM is identical to the one for a TM, except that the working tape may be non-empty at the start of computation. The notion of *extended computation* (Definition 3.4) also carries over.

Even though a PTM's work tape contents W affects its output value, W is completely determined by the PTM's interaction history, and is not under the control of the observer. Note that W is always finite; however, its size is unbounded. As a result, a PTM is similar in its behavior to a data abstraction with an unbounded state, such as a queue or a stack.

Example 4.1 We can define an Answering Machine PTM $\mathcal{A}$ which expects inputs of the following three types:

record message, *playback*, *erase*.

The *record* input causes $\mathcal{A}$ to append the message to the contents of the work tape and output *OK*; the *playback* input causes $\mathcal{A}$ to output the contents of the work tape; the *erase* input causes $\mathcal{A}$ to reset the work tape to *empty* and output *OK*. □

The above example is a very simple machine, yet it models very useful behavior. As seen from this example, the output of a PTM computation may depend both on the given input and on the contents of the working tape.

The PTM computational semantics are a generalization of the TM computational semantics:

Proposition 4.1 (Semantics of PTM computations) *A PTM $\mathcal{M}$ defines a partial recursive function $f_\mathcal{M}$:*

$f_\mathcal{M}(input,\ W) = (output,\ W')$.

Proof: Any PTM $\mathcal{M}_1$ can be simulated by a TM $\mathcal{M}_2$ such that a computation of $\mathcal{M}_1$ on input I with initial working tape W yielding output O with final working tape W' can be simulated by a computation of $\mathcal{M}_2$ on input $I\#W$ yielding output $O\#W'$. □

Example 4.2 If $\mathcal{A}$ is the Answering Machine of Example 4.1, then

$f_\mathcal{A}(\textit{record}\ \text{X},\ \text{Y}) = (\textit{OK},\ \text{Y X})$
$f_\mathcal{A}(\textit{playback},\ \text{X}) = (\text{X},\ \text{X})$
$f_\mathcal{A}(\textit{erase},\ \text{X}) = (\textit{OK},\ \epsilon)$

If $\mathcal{A}_1$ and $\mathcal{A}_2$ are two Answering Machines where we record "Hi" on $\mathcal{A}_1$ and "Hello" on $\mathcal{A}_2$, they produce different outputs for the same *playback* input. □

The above example demonstrates that the behavior of a PTM is *history-dependent* (Definition 2.6); identical PTMs with different interaction histories may exhibit different behavior, which is in contrast to the behavior of TMs. From the viewpoint of a system observer, systems that preserve the state between interactions are more natural than those that reinitialize their state for every interaction, suggesting that PTMs constitute a practical extension to the TM model of computation.

5 On the Expressiveness of PTMs

As the examples in Section 4 suggest, any *abstract data type* (ADT) can be modeled by a PTM whose work tape represents the ADT's internal value. The dependence of ADT's operations on its internal value corresponds to the dependence of PTM computations on the work tape contents. The fact that the work tape is initially empty corresponds to an assumption that all the fields of an ADT are initially uninitialized, and that a call to the ADT's *INIT* operation sets their value.

We present several propositions on the expressiveness of PTMs.

Proposition 5.1 *PTMs are at least expressive as TMs.*

Proof: PTMs can simulate any TM with a non-persistent work tape, by erasing the tape at the beginning of any computation. □

Proposition 5.2 *Without changing their expressiveness, we can extend PTMs to allow their current state to be persistent, so the initial state is not necessarily S_0.*

Proof: The state can be written as a part of the work tape contents. □

For the rest of the article, we assume that PTMs may have both state and tape persistent.

Proposition 5.3 *Without changing their expressiveness, we can extend PTMs to allow a non-empty initial work tape.*

Proof: We can always define an *INIT* operation which initializes the work tape if it is empty. □

6 Comparing PTMs and TMs

Both PTMs and TMs can be observed via interaction sequences, which allows us to compare the expressiveness of these two classes of systems.

We apply Definition 2.1 to characterize the observable behavior of TMs and PTMs as a set of interaction sequences: *given a TM or a PTM $\mathcal{M}$, its observable behavior $\mathcal{B}(\mathcal{M})$ is a set of all finite interaction sequences observable for $\mathcal{M}$.* We make the notion of observable interaction sequences more precise, for both classes of systems, in the following proposition:

Proposition 6.1 *Let σ be an arbitrary interaction sequence $\{(i_1, o_1), (i_2, o_2), ...(i_k, o_k)\}$, and let $\mathcal{M}_T$ and $\mathcal{M}_P$ be a TM and a PTM, respectively. $\sigma \in \mathcal{B}(\mathcal{M}_T)$ iff, for all j, $1 \leq j \leq k$,*

$$f_{\mathcal{M}}(i_j) = (o_j).$$

$\sigma \in \mathcal{B}(\mathcal{M}_P)$ iff there exists a sequence $\{W_O, W_1, \ldots, W_k\}$ such that for all j, $1 \leq j \leq k$,

$f_{\mathcal{M}}(i_j, W_{j-1}) = (o_j, W_j)$,

and W_0 is a reachable work tape configuration*, i.e., $\mathcal{M}_P$ may acquire W_0 as a result of some other interaction sequence that starts with an empty tape.*

Proof: Follows from the computational semantics of TMs and PTMs. □

Example 6.1 Consider interaction patterns of length three for the Answering Machine $\mathcal{A}$ of Example 4.1:

{(playback, hello), (record hi, OK), (playback, hello hi)} $\in \mathcal{B}(\mathcal{A})$;
{(playback, hi), (record hello, OK), (playback, hi hello)} $\in \mathcal{B}(\mathcal{A})$;
{(playback, hi), (playback, hi), (record hello, OK)} $\in \mathcal{B}(\mathcal{A})$;
{(playback, hi), (record hello, OK), (playback, hi)} $\notin \mathcal{B}(\mathcal{A})$ □

For any k, there exists a pair of distinguishable PTMs whose shortest SDC has length $k+1$:

Proposition 6.2 *For any $k \geq 0$, there exists a pair of PTMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $dist(\mathcal{M}_1, \mathcal{M}_2) = k+1$.*

Proof: Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two PTMs with bit inputs 0 and 1, and boolean outputs *true* and *false*.

$\mathcal{M}_1$: output *true* if the last k values it saw were 1's, *false* otherwise (i.e., if it has not seen that many values, or if some of them were 0).
$\mathcal{M}_2$: same as $\mathcal{M}_1$, but $k+1$ values instead of k.

It can be shown that $dist(\mathcal{M}_1, \mathcal{M}_2) = k+1$; the corresponding SDC σ has inputs $\{0, 1, 1, \ldots, 1\}$ (k 1's), and outputs $\{F, F, \ldots, F, T\}$ (k F's). σ is in $\mathcal{B}(\mathcal{M}_1)$ but not in $\mathcal{B}(\mathcal{M}_2)$. □

This proposition implies that the k-equivalence partitioning of the class of PTMs (Definition 2.4) is different for every k, producing ever finer classes of distinguishable machines.

Proposition 6.3 *Any pair of TMs is* 1*-distinguishable.*

Proof: Let $\mathcal{M}_1$, $\mathcal{M}_2$ be distinguishable TMs, and σ their distinguishability certificate:

$$\sigma = \{(i_1, o_1), (i_2, o_2), \ldots (i_k, o_k)\};$$

w.l.o.g., let $\sigma \in \mathcal{B}(\mathcal{M}_1)$. By Proposition 6.1, for all j, $1 \leq j \leq k$, $f_{\mathcal{M}_1}(i_j) = (o_j)$, but there exists m, $1 \leq m \leq k$, such that $f_{\mathcal{M}_2}(i_m) \neq (o_m)$. Therefore, $\{(i_m, o_m)\}$ is a distinguishability certificate of length 1 for $(\mathcal{M}_1, \mathcal{M}_2)$. □

In Proposition 5.1, we have shown that PTMs are at least as expressive as TMs. In addition, we have shown in Propositions 6.2 and 6.3 that TMs do not allow for the same hierarchy of k-equivalent machines as PTMs.

Theorem 6.1 *PTMs are strictly more expressive than TMs.*

Proof: Follows from Propositions 5.1, 6.2 and 6.3. □

7 Sequential Interaction Machines

In the classical Turing Machine model, all computation proceeds after the input value is completely determined and terminates once the output value is completely specified. This is to be contrasted with the *interactive model of computation* where I/O actions may take place in the middle of computation. What makes I/O actions special is that these are the only actions of the TM that allow it to interact with the external environment, obtaining values that are not under its direct control. It is interesting to note that Turing's original paper on automata [Tu] provided for the possibility of automata with interactive computational steps; these *c-automata*, for *choice* were defined and contrasted with *a-automata*, for *automatic*. However, the rest of the paper only considered a-automata, and the *a* prefix was discarded.

Definition 7.1 *The property of having I/O actions taking place during the computation is called the* interaction property.

Definition 7.2 *An* Interaction Machine *(IM) is a TM extended with the interaction property.*

Note that this is an intensional definition in terms of the interaction property rather than an extensional definition. There are many modes of interaction, as discussed in [We1]. We will consider those IMs where the computation associated with any interaction is completed before the next interaction can begin.

Definition 7.3 *A Sequential Interaction Machine (SIM) is an IM whose interactions are sequential.*

SIMs model serializable transaction systems, where there is no interference between sequential transactions.

Example 7.1 We can define a Bank Account SIM as an example of a simple transaction system, accepting three types of inputs:

deposit value, *withdraw* value, *check-balance*. □

Note that sequential acceptance of inputs is not a sufficient condition for SIMs, since a new input can interfere with ongoing computations from a previous input. For example, actors [Ag] that accept inputs sequentially from their mail queue are not SIMs because they allow computations of successive inputs to run concurrently.

Proposition 7.1 (Semantics of SIM computation) *A SIM $\mathcal{M}$ defines a partial recursive function $f_{\mathcal{M}}$ from a sequence of inputs to one output.*

Proof: Any SIM $\mathcal{M}_1$ can be simulated by a TM $\mathcal{M}_2$ such that the output of $\mathcal{M}_1$ on input i_0 with an interaction history $(i_1, ..., i_k)$ can be obtained by a computation of $\mathcal{M}_2$ on input $i_1\#i_2\#\ldots\#i_k\#i_0$. □

Example 7.2 If $\mathcal{M}$ is the Bank Account of Example 4.1, and the sequence of all the inputs it has received is $(i_1, ..., i_k)$, then we can figure out its output for a new input i_0 by computing the successive values of the bank account after each i_j. □

Note that this is a *black-box* model of computation, where the system's internal state is not involved in the computational semantics of the system. This contrasts with the semantics of PTM computations (Proposition 4.1), where a PTM's working tape contents W is assumed to be internal to the system, but is explicitly represented in the semantics.

8 Equivalence of SIMs and PTMs

In this section, we show the equivalent expressive power of SIMs, which are interaction machines restricted to sequential interaction patterns, and PTMs, which are TMs extended to have persistent memory.

We cannot claim that a computation of a PTM is equivalent to a computation of a SIM, since the latter may include I/O actions and the former may not. This represents two different views of a computation, as a transformation of inputs to outputs or as a reactive process that continuously receives new inputs and processes them [MP]. However, both SIM computations and *extended computations* of PTMs (Section 4), can be characterized by interaction sequences (Definition 2.1), and are therefore directly comparable.

Proposition 8.1 *For any PTM $\mathcal{M}_P$, there exists a SIM $\mathcal{M}_I$ such that $\mathcal{B}(\mathcal{M}_P) = \mathcal{B}(\mathcal{M}_I)$, and vice versa.*

Proof (sketch): Let $\sigma \in \mathcal{B}(\mathcal{M}_P)$ be some interaction sequence $\{(i_1, o_1), (i_2, o_2), ...(i_k, o_k)\}$. By Proposition 6.1, this implies the existance of $\{W_O, W_1, \ldots, W_k\}$ such that for all j, $1 \leq j \leq k$, $f_{\mathcal{M}_P}(i_j, W_{j-1}) = (o_j, W_j)$, and W_0 is reachable from the empty tape by an interaction history σ'. This allows us to define g, where for all j, $1 \leq j \leq k$, $g(\sigma', i_1, \ldots, i_j) = o_j$. Therefore, $\sigma \in \mathcal{B}(\mathcal{M}_I)$, where $\mathcal{M}_I$ is the SIM with the semantics g. The other direction is done similarly. □

The correspondence between the black-box notion of sequential interaction and the glass-box notion of persistence may at first seem surprising, but careful analysis of our definitions shows that relative persistence is defined by interaction, and the ability of persistence to express a form of interaction is therefore to be expected: *the notion of persistence is equivalent to the notion of sequential interaction.*

Corollary 8.1 *SIMs are more expressive that TMs.*

Corollary 8.1 provides a formal justification for the claim made in [We1] that IMs are more expressive than TMs. We expect that the general class of IMs is more expressive than SIMs; in particular, there is strong evidence that IMs with multiple input streams are more expressive than SIMs.

9 Towards Empirical Models of Computing

This paper formalizes a limited form of interaction by two alternative equivalent models: PTMs and SIMs. Our model is related to formalizations of process models of observability [Mi2], reactive systems by temporal logis [MP], specification models of data abstraction [Br], and input-output automata [Ly]. It shows how PTMs can be defined by a minimal extension of TMs and that making memory persistent (an inner property) corresponds to making the machine interactive (an outer property). The ability to remember elevates the expressiveness of PTMs above that of TMs, giving them a sense of identity.

This work is a part of a larger effort that aims to characterize empirical computation by interaction. We conjecture that unrestricted interactive computation is more expressive than the restricted form of interaction permitted by PTMs and SIMs. In particular, there is strong evidence that machines with multiple input streams are more expressive than SIMs; this contrasts with the result that

TMs with multiple tapes are no more expressive than TMs with a single tape. In [We2], we show that observational expressiveness for models of computation corresponds to explanatory power for models of physics, and propose the thesis that observational expressiveness captured by interaction machines characterizes the intuitive notion of empirical computer science just as algorithmic expressiveness characterized by TMs captures Church's notion of effective computability.

References

[Ag] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[BA] J.D. Brock, W.B. Ackerman. *Scenarios: A Model of Non-Determinate Computation*, pp. 252–259. LNCS, Springer-Verlag, 1981.

[Br] Manfred Broy. A Theory for Nondeterminism, Communication, and Concurrency. *Theoretical Computer Science* 45, 1986.

[Gl] R.J. Van Glabbeek. The Linear Time - Branching Time Spectrum. *CONCUR 90*, pp. 278–297. LNCS 458, Springer, 1990.

[GW] Dina Goldin, Peter Wegner. On the Expressiveness of Interactive Observers. Work in Progress.

[Kn] Donald Knuth *The Art of Computer Programming*, Volume 1. 1968

[LP] Harry Lewis, Christos Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, 1998.

[Ly] N. Lynch, M. Merritt, W. Weihl, A. Fekete. *Atomic Transactions.* Morgan Kaufmann, 1994.

[Mi1] Robin Milner. Elements of Interaction. *Communications of the ACM*, January 1993.

[Mi2] Robin Milner. Operational and Algebraic Semantics of Concurrent Processes. *Handbook of Theoretical Computer Science*, J. van Leeuwen editor, Elsevier, 1990.

[MP] Zohar Manna, Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer Verlag, 1992.

[Pa] Christos Papadimitriou. *Computational Complexity.* Addison Wesley, 1993.

[Pn] Amir Pnueli. Linear and Branching Structures in the Semantics and Logics of Reactive Systems, in W. Brauer, editor: *Proceedings 12th ICALP*, Nafplion

[Tu] Alan Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. of the London Mathematical Society*, 2:42, pp. 230–265, 1936.

[We1] Peter Wegner. Interactive Foundations of Computing. *Theoretical Computer Science*, Feb. 1998.

[We2] Peter Wegner. Towards Empirical Computer Science, *The Monist*, special issue on the Philosophy of Computer Science, Jan. 1999