Interaction as a Framework for Modeling

Peter Wegner[1] and Dina Goldin[2]

[1] Brown University
Providence, RI
pw@cs.brown.edu
[2] U.Mass - Boston
Boston, MA
dqg@cs.umb.edu

Abstract. The irreducibility of interactive to algorithmic computing requires fundamental questions concerning models of computation to be reexamined. This paper reviews single-stream and multiple-stream interaction machines, extensions of set theory and algebra for models of sequential interaction, and interactive extensions of the Turing test. It motivates the use of interactive models as a basis for applications to computer architecture, software engineering, and artificial intelligence.

1 Concepts of Modeling

Interactive models of computation provide a unifying conceptual framework that captures changes in computer architecture, software engineering, and AI in the last quarter of the 20th century [We1]. The evolution of computer architecture from mainframes to personal computers and networks, of software engineering from procedure-oriented to object-oriented and component-based systems, and of AI from logic-based to agent-oriented and distributed systems has followed parallel paths. The evolution of programming can, as a first approximation, be described as follows:

1950s: machine language programming, assemblers, hardware-defined action sequences
1960s: procedure-oriented programming, compilers, programmer-defined action sequences
1970s: structured programming, composition of action sequences, algorithm architecture
1980s: object-based programming, personal computers, sequential interaction architecture
1990s: structured object-based programming, networks, distributed interaction architecture

Whereas the shift from machine to procedure-oriented programming involves a quantitative change in the granularity of actions, the shift from procedure-oriented to object-based programming is more fundamental, involving a qualitative paradigm shift from algorithms to interactive computing. The shift from sequential interaction with a single agent to distributed interaction among multiple agents is a further fundamental paradigm shift in underlying models of computation. Turing machines (TMs) provide a robust model of computation for algorithmic (action-oriented) programming. Our goal is to develop interactive models of computation whose role is comparable to that of TMs for algorithms.

Database models have evolved separately from programming models, from hierarchical to relational and entity-relation models. Conceptual modeling [Ch] is rooted in the database tradition, flowered in the decade 1975-1985, and led to the development of object-oriented databases [ZM]. But this area failed to develop underlying models of computation comparable to those of Turing machines for automatic computing. It could not shake off the notion that database modeling should be rooted in algorithmic modeling, so that even books published in the mid 1990s [KR] appeal primarily to the tradition of Turing and Dijkstra in establishing a conceptual foundation for information modeling.

Turing's proof that algorithms, TMs, and the lambda calculus are equally expressive [Tu1], suggested that the question of expressiveness of computing had been settled once and for all, and channeled research to questions of design, performance, and complexity for a fixed notion of computability. Church's thesis that computable functions capture the intuitive notion of computability contributed to the belief that computing could be identified with algorithmic computation. The claim that interactive computation is more expressive than algorithms opens up a research area that had been considered closed, requiring fundamental questions concerning models of computation to be reexamined.

Our work over the last five years has shown that interactive models are not reducible to or expressible by algorithmic models. This paper reviews concepts and models of interactive computation discussed in [We1,We2,We3,We4,We5,WG]. Readers interested in an introduction to this work should read [We1], which is easily accessible and written for a general audience. The fundamental model is described in depth in [WG], which builds on recent extensions to set theory and algebra [Ac,BM,Ru1] in developing a model for interactive computing.

The irreducibility of interactive systems to algorithms was noticed by [MP] for reactive systems, by [Mi] in the context of process models, and by many other researchers. Our approach differs from related work in focusing on models of interaction and notions of expressiveness that are language-independent as well as domain-independent. A unifying language-independent model of computation for a variety of levels of expressiveness is developed, and is related to mathematical models of set theory and algebra [Ac,BM,Ru1]. This work has led to the development of persistent Turing machines (PTMs) as a canonical model of sequential computation, of an expressiveness hierarchy for sequential computation, and of the result that multi-stream interaction machines (MIMs) are more expressive than sequential interaction machines (SIMs).

Turing's seminal paper [Tu1] was not intended to establish TMs as a comprehensive model for computing but on the contrary to show undecidability and other limitations of TMs. Turing actually mentions irreducibility to TMs of interactive choice machines (*c-machines*) as a presumably well-known fact [Tu1].

TMs model "automatic" computation, while IMs extend the notion of what is computable to nonautomatic (interactive) computation. IMs can model TMs with oracles, which Turing showed to be more powerful than TMs in his Ph.D. thesis [Tu2].

The robustness of TMs as a model of computation led to Church's thesis that the intuitive notion of computability corresponded to TM computation of computable functions. Interactive computation has a richer notion of computability corresponding to computation of sequences and patterns of functions (algorithms). The richer notion of computation is captured mathematically by extending the input domain of functions from predetermined strings to interactively generated steams that cannot be represented as functions from integers to integers. Greater computation power is due not to greater transformation power but to richer domains of transformable inputs, that can be specified by non-well-founded sets.

The idea that interaction is not expressible by or reducible to algorithms was first proposed in 1992 at the closing conference of the fifth-generation computing project in the context of logic programming [We4]. Reactiveness of logic programs, realized by commitment to a course of action, was shown to be incompatible with logical completeness, realized by backtracking. The 5th-generation project's failure of achieving its maximal objective of reducing computation to first-order logic was attributed to theoretical impossibility rather than to lack of cleverness of researchers. The implication that success could not have been achieved by a twenty year extension or substantial further research shows that impossibility results may have practical impact in justifying strategic research decisions.

The irreducibility of interaction to algorithms and of computation to first-order logic implies that tools of algorithm analysis and formal methods cannot, by themselves, solve the software crisis. Software tools for design patterns, life-cycle models, embedded systems, and collaborative planning are not modeled by algorithms and their behavior cannot inherently be expressed by first-order logic. Fred Brooks' claim that there is no silver bullet for systems can be proved if we define "silver bullet" to mean "formal or algorithmic specification". New classes of models are needed to express the technology of interaction, since software technology has outstripped algorithm-based models of computation.

2 Models of Interactive Computing

Finite agents that perform actions, transforming inputs into outputs, are modeled by Turing machines (TMs), while finite agents that provide services over time, like objects, components, and databases, are modeled by interaction machines (IMs). TMs transform predetermined input strings into output strings, shutting out the world during the process of computation, while interaction machines (IMs) are transducers of interactively generated streams, solving a larger class of problems that TMs by interaction during the process of computation.

The shift from algorithms to interaction may be described as a shift from models of string transformation to models of stream transduction.

Expressiveness of finite computing agents is captured by the notion of observation equivalence, that measures the ability to make observational distinctions. Expressiveness depends both on the inherent computation power of observed systems and on the power of observers to observe or cause system behavior. The observed expressiveness of systems cannot be greater than that of observers who interact with them. When observers (testers) are limited to TMs that perform single interactions (ask single questions) then observed behavior is limited to TM behavior even for systems capable of interactive behavior [WG].

TMs are limited to single interactions that transform inputs to outputs, while IMs permit a greater range of interactive behavior, thereby extending the range of behavior of finite agents. Two distinct levels of interactive expressiveness corresponding to single-stream sequential interaction and multi-stream distributed interaction are identified:

single-stream (sequential) interaction: realized by sequential interaction machines (SIMs)
multi-stream (distributed) interaction: realized by multi-stream interaction machines (MIMs)

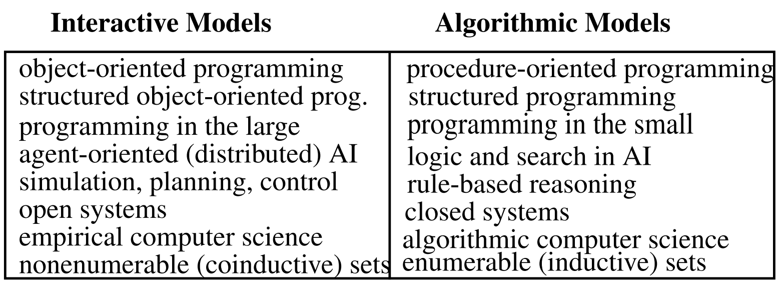
SIMs define a domain-independent model of sequential interaction that expresses sequential interaction between two objects in question-answering and control applications and more generally the interaction of sequential object-oriented languages and databases. MIMs provide a more expressive model for multi-stream (distributed) interaction that allows finite agents to interact with multiple autonomous agents, like an airline reservation systems that interacts with multiple travel agents or a distributed database that provides services to multiple autonomous clients. The greater expressiveness of MIMs over SIMs captures the greater problem-solving power of senior managers and CEOs over workers (finite agents) that perform sequences of interactively assigned tasks. Collaboration and coordination is shown to be more expressive than and not reducible to sequential interaction, contrasting with the fact that multitape TMs are no more expressive than single-tape TMs.

Adding autonomous streams (observation channels) to a finite agent increases expressiveness, while adding noninteractive tapes simply increases the structural complexity of predetermined inputs, and does not increase expressive power. Finite agents with autonomous multiple streams model distributed inputs by a form of interactive concurrency that is dual to algorithmic concurrency of process execution. MIMs support the behavior of nonserializable transactions and true concurrency [Pr], while SIMs support only serializable transactions and interleaving concurrency.

Sequential interaction is modeled mathematically by non-well-founded set theory [WG] and is captured by PTMs, which provide a canonical model for SIMs by a minimal extension of TMs to multitape machines with persistence. We show that PTMs with k+1 interactions are more expressive than PTMs with k interactions for all k, and that TMs have the expressiveness of PTMs with k=1. This hierarchy of expressiveness implies that interactive questioners can elicit more information about a questioned subject by k+1 sequences of follow-up questions than by k questions.

Observed behavior of finite agents can be meaningfully specified only relative to an observer, both for physical and computational systems. TM observers can observe only behavior for single I/O pairs, while SIM observers can observe the behavior of interactively supplied sequences of inputs that can depend on dynamically occurring events in the environment. For MIMs, finite agents can observe only sequential behavior of single streams. Complete behavior at all interfaces of a distributed system, such as an airline reservation system or nuclear reactor, cannot be observed by finite agents, though it could possibly be observed by physically unrealizable distributed observers (God). Since there are forms of behavior that can be specified but not observed, there is a distinction between specifiable behavior and observable behavior; in particular, the complete behavior of MIMs is specifiable but not observable.

The impact of models of observation on observed behavior has a simple form for SIMs, where stronger distinguishing power yields more expressive behavior, and has a subtler form for MIMs, where limitations of sequential observers in observing complete system behavior is related to quantum nondeterminism. Parallels between computational and physical models explored in [We3,WG] establish substantive connections between empirical computer science and models of observation in relativity and quantum theory. In particular, interactive models provide a stronger formulation of Einstein's argument that nondeterminism is due to incomplete observability, based on hidden interfaces rather than hidden variables. Einstein's intuition that God does not play dice and that observed nondeterminism is due to weaknesses of our model of observation rather than inherent in nature is modeled by primary observers who have no control over "random noise" due to secondary observers.



Interactive Models	Algorithmic Models
object-oriented programming	procedure-oriented programming
structured object-oriented prog.	structured programming
programming in the large	programming in the small
agent-oriented (distributed) AI	logic and search in AI
simulation, planning, control	rule-based reasoning
open systems	closed systems
empirical computer science	algorithmic computer science
nonenumerable (coinductive) sets	enumerable (inductive) sets

Fig. 1. Parallel Robustness of Interactive and Algorithmic Models of Computation

The parallelism between algorithmic concepts and corresponding interactive concepts is shown in Figure 1. Each algorithmic concept in the right-hand column is paralleled by a more expressive interactive concept in the left hand column. Object-oriented programs are more expressive than procedure-oriented programs in providing clients with continuing services over time: they are specified by marriage contracts that cannot be expressed by procedure-oriented sales contracts [We1]. Structured programming for actions (verbs) can be formally defined by compositional function specifications, while structured object-oriented programming is not compositional: composite structures of interactive components (nouns) have emergent behavior, such that the whole is greater than the sum of its parts. Programming in the large (PIL) is not determined by size, since a program with a million addition instruction is not PIL while an embedded system with only hundreds of lines of code is PIL. PIL can plausibly be defined as interactive programming and differs qualitatively from programming in the small (PIS) in the same way that interactive programs differ from algorithms.

Agent-oriented and distributed AI differs from logic-based AI in the same way that PIL differs from PIS. Simulation, planning, and control for interactive applications are not expressible by rule-based reasoning. Open systems cannot be expressed by closed algorithms that compute outputs noninteractively from their inputs. Interaction distinguishes robustly and precisely between empirical computer science and algorithmic models of computation. Each of the entries on the left hand column of Figure 1 has both sequential applications modeled by SIMs and more expressive distributed applications modeled by MIMs.

These parallel extensions from algorithms to interaction are captured by a mathematical paradigm shift that provides a foundation for sequential models of interaction, and requires new modes of mathematical thinking that will play an increasingly important role in undergraduate and graduate education. A key element in this paradigm shift is the extension from inductive definition and reasoning about enumerable sets to coinductive definition and reasoning about nonenumerable sets.

3 Mathematics of Interactive Modeling

Though impossibility results are useful in avoiding resource expenditures on unsolvable problems and in explaining the low profile of formal and algorithmic techniques in software engineering, they are not a sufficient basis for developing a technology of interactive computing. The impossibility of modeling interaction by mathematical models of first-order logic, recursive function theory, or traditional set theory at first suggested a sharp limitation of mathematics in modeling interaction and a tradeoff between formal algorithmic models and unformalizable interactive models. Fortunately a mathematical framework for interactive modeling has emerged [Ac,BM] that makes such a choice unnecessary.

The paradigm shift from algorithms to interaction is captured by a mathematical paradigm shift from inductive principles of definition and reasoning to stronger coinductive principles. Zermelo-Frankel set theory is extended by replacing the inductive *foundation axiom* by the coinductive *anti-foundation axiom*, which asserts that set equations of a certain form have a unique solution and provides a basis for non-well-founded set theory. This method of augmenting classes of mathematical objects is similar to algebraic methods of introducing rationals as solutions of linear equations with integer coefficients. Non-well-founded sets provide a consistent formal model for sequential interaction by formalizing stream behavior, and a denotational semantics for interactive models of sequential computation. Coalgebras provide a coinductive operational semantics for non-well-founded set theory [Ru1].

Induction is a technique for definition and reasoning over enumerable linear or partially ordered structures such as strings and trees. Coinduction is a technique for definition and reasoning over nonenumerable circular structures such as graphs and streams. TMs are inductively defined string processing machines that specify enumerable computations, while interaction machines are coinductively defined stream processing machines that specify nonenumerable computations. Inductively defined well-founded sets have enumerable elements, while coinductively defined non-well-founded sets may have nonenumerable elements.

Induction is specified by an *initiality condition*, an *iteration condition* that allows new elements to be derived from initial elements, and a *minimality condition* that only elements so derived can be considered. Coinduction eliminates initiality and is specified by an *iteration condition* and a *maximality condition* that models observers who consider all "possible worlds" not ruled out by observation.

> **Inductive definition**: initiality condition, iteration condition, minimality condition.
> **Coinductive definition**: iteration condition (circularity condition), maximality condition.

Initiality is a closed-system requirement whose elimination makes it possible to model open systems. There is a close relation between elimination of inductive initiality in mathematical models and shedding of the initial-state requirement of TMs for SIMs and PTMs. Initiality embodies preconceptions about the modeled world, and its elimination allows modeling of systems without preconceptions about the nature of their environments. In particular, predetermined static environments required by initiality can be generalized to incrementally supplied dynamic environments.

The minimality condition "everything is forbidden that is not allowed" specifies a smaller class of things than the more permissive maximality condition "everything is allowed that is not forbidden". Minimality is a property of constructive paradigms for definition and reasoning that characterize both algorithmic computation and constructive mathematics, while maximality is a property of empirical observation paradigms for describing observed behavior in an already constructed (existing) world. Minimality is modeled by least fixed points while maximality is modeled by greatest fixed points.

The distinction between minimality and maximality is a ubiquitous principle of system organization that extends from mathematical to social and political systems:

> *totalitarianism embodies the minimality principle: everything is forbidden that is not allowed*

democracy supports the maximality principle: everything is allowed that is not forbidden

This association of algorithms with rule-governed restrictive forms of organization and of interactive models with permissive flexible forms of organization is a central theme in philosophy and the history of ideas, expressed by Isaiah Berlin in his parable of the hedgehog (who knows one big thing) and the fox (who knows many little things) [Be]. Hedgehogs embody minimalist depth at the expense of flexibility, while foxes are maximalist opportunists. Algorithmic models are hedgehog-like, focusing on functional correctness, while interactive models are fox-like, allowing dynamic flexibility of goals and forms of behavior that are to be modeled. Maximality and elimination of initiality extend mathematics so it can model fox-like systems that arise in computing, physics, and social organization.

Minimality of behavior is associated with maximality of constraints on behavior, while maximality of behavior is associated with minimality of constraints. This contravariance of constraints and behavior is illustrated by Berlin's essay "Two Concepts of Liberty" [Be], that contrasts negative freedom from interference by others with positive freedom to determine one's course of action. Berlin advocates negative freedom associated with minimal constraints on behavior and maximal fixed points, and is deeply suspicious of positive freedoms advocated by socialism or even liberalism, because they are realized by constraints designed to improve social conditions by constraining the freedom of individuals.

Maximal fixed points provide a mathematical framework for the empirical paradigm that any behavior (possible world) consistent with observation is admissible. Specifications that admit any possible world consistent with a specification are maximal fixed points. The reluctance of mathematicians to consider coinductive maximal fixed-point models stems in part from an inadequate analysis of the notion of circular reasoning [BM]. Russell and others failed to distinguish between inconsistent forms of circular reasoning that led to the paradoxes of set theory and consistent forms of circular reasoning.

However, a deeper reason was the strong influence on logic of Brouwer's intuitionism and Hilbert's formalism, which caused logicians to focus on properties of formalisms (proof theory) and to exclude consideration of independent concepts that formalisms aim to model. Constructive mathematics accords existence only to things that can be constructed, and even nonconstructive formalists restrict existence to finitely formalizable enumerable specifications. Both exclude concepts like the real numbers or the real world specified by nonenumerable elements or situations, which are unformalizable by enumerable inductive models, but can be formalized by coinductive (circular reasoning) models.

Godel's incompleteness result showed the impossibility of Hilbert's program of reducing mathematics to logic and by implication the impossibility of reducing computing to logic. In spite of this, Turing's model of computation is in the formalist tradition, implicitly accepting the Godel's limitation of incompleteness. We show in section 5.2 that interactive models are formally incomplete in the sense of Godel.

Finsler's remarkable work on set theory in the 1920s [Fi] showed the consistency of non-well-founded set theory and anticipated Godel's incompleteness result. His work was largely ignored because it did not conform to the prevailing formalist tradition of Russell, Hilbert, and Tarski, resting instead on the Platonic tradition (espoused by Cantor) that concepts existed independently of the formalism in which they were expressed. Whereas formalist accepted "existence implies consistency", Finsler accepted the stronger ontological commitment "consistency implies existence". Finsler argued that since non-well-founded set theory was consistent it existed conceptually, independently of whether it had any useful models.

The discovery sixty years later that non-well-founded set theory models interactive computing provides an incentive for reevaluating Finsler's work, and validates the Platonic approach of according existence to conceptually consistent possible worlds independently of their formalizability or constructibility. Tensor analysis is another example of an abstract mathematical technique that found uses in relativity theory and a variety of other areas of applied mathematics many years after it was initially developed.

Godel's constructive demonstration of incompleteness of first-order logic by arithmetization of metamathematics (Godel numbering) was hailed as a seminal result by formalist logicians, while Finsler's prior result that formalisms are incomplete in expressing concepts, because we can conceive nonenumerable sets while formalisms express only enumerable sets, were not considered significant. Interactive models show the limitations of formalist inductive mathematics and have contributed to a mathematical paradigm shift that reestablishes the importance of independently given conceptual domains in mathematical modeling. Though philosophical questions concerning the foundations of mathematics are outside the scope of this paper, they are relevant to understanding the relation between formally specified finite interactive agents and their nonformalized environment.

4 Behaviorist Conceptual Modeling

The Turing test, which aims to express cognitive and conceptual inner properties of systems by their behavior, was criticized by Searle [Se] on the grounds that behavior cannot capture intentionality, and by Penrose [Pen] on the grounds that TM behavior is too weak to model the extensional behavior of humans or physical systems:

intentional skepticism: TMs cannot model inner (intentional) qualities of thinking
extensional skepticism: TM behavior is extensionally too weak to model thinking

We show [We1,WG] that replacing TMs by IMs in the Turing test (Figure 2) allows us to model not only stronger extensional behavior but also stronger intentional behavior. Turing not unnaturally assumed that the computing agent with

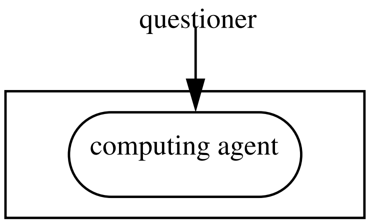


Fig. 2. Interactive Turing Test

which a questioner communicated was a TM, and his examples of Turing-test dialogs assumed that sequences of questions were unrelated, following a non-interactive, predetermined script [Tu3]. The interactive Turing test preserves Turing's behaviorist assumption that thinking is specifiable by behavior, but extends modes of questioning and responding to be interactive. Analysis of interactive question answering yields behaviorist models of "thinking" that are qualitatively different from the traditional Turing test model.

algorithmic thinking: TMs that answer single questions by algorithmic computation
algorithmic Turing test: single questions with algorithmically computable answers

sequential thinking: PTMs (SIMs) that answer sequences of history-dependent questions
sequential Turing test: dialog where questioner can ask sequences of follow-up questions

distributed thinking: MIMs that react to multiple autonomous streams of requests
multi-agent Turing test: coordination and collaboration for multiple autonomous streams

Algorithmic, sequential, and distributed thinking are progressively stronger forms of behavioral approximation to human thinking, defined by progressively more stringent empirical tests of behavior. Sequential thinking is realized by SIMs and observed by observers that ask follow-up questions (make sequential observations). Distributed thinking is realized by MIMs and requires multi-agent observers that can observe how the system responds to multiple potentially conflicting asynchronous requests.

Testing for multi-agent behavior extends the Turing test beyond SIMs to interaction with external resources like the Library of Congress, the Web, or human experts. Resources accessible to an IM are hidden from the questioner, who cannot distinguish whether the tested agent is an IM or a TM. The interactive Turing test extends machines without changing the form of the Turing test. Turing would certainly have accepted the sequential and multi-agent Turing tests as legitimate, and would have approved of the behaviorist notions of sequential and distributed thinking as conforming to the spirit of his notion of thinking.

Current IQ tests that focus on algorithmic intelligence by asking single questions have been shown to be poor predictors of later human success because they measure only limited forms of human problem solving. Sequential thinking can be tested by comparing the response of systems to sequences of related questions with human responses. Distributed thinking is tested by comparing system responses to autonomous streams with human responses. Sequential and distributed thinking require radical revision of methods of intelligence testing. Devising "intelligence" tests for sequential and distributed thinking is a challenging task that raises interesting research issues and could correlate significantly better than current IQ tests with later human success. Studies have shown that tests that measure human ability to delay gratification, such as the marshmallow test [Go], are better predictors of later success than IQ tests.

The arguments of both intentional and extensional skeptics lose much of their force when the algorithmic Turing test is replaced by the sequential and multi-agent Turing tests. The weak extensional behavior of TMs causes them to be ineffective in modeling intensional behavior. SIMs and MIMs can express stronger forms of intentionality because of their stronger extensional behavior. Searle's argument that machines cannot express intensional behavior is partly neutralized by SIMs that express intentionality through history dependence, and further weakened by MIMs that express the strong intentionality of multi-agent collaboration. The translators of Chinese stories in Searle's Chinese-room experiment must do more than follow transliteration rules in responding to interactive questions of sequential or multi-agent Turing tests. SIMs and MIMs have many, though not all, the properties that Searle calls intensional.

Penrose's arguments that computers cannot express extensional behaviors of physical object [Pen] are valid for TMs and possibly for SIMs but certainly not for MIMs [We3]. We agree with Penrose that TM models of computation are extensionally too weak to model physics. However, physics is computable according to the broader notion of MIM computability: Penrose's arguments about the noncomputability of physics by TMs become inapplicable when computing is broadened to include interaction.

Though Searle and Penrose are right that TMs are too weak to model thinking, the generalization that thinking cannot be modeled by behavior is an erroneous overreaction. The Turing test is deficient because TMs are too weak as a model of behavior, not because, as claimed by Searle and Penrose, computational behavior is an inadequate model of thought. Searle and Penrose were right about the inadequacy of the Turing test, but for the wrong reasons. By contrast, Turing had the right experimental design but applied it to the wrong model of computation. The interactive Turing test not only augments the extensional behavior that machines can exhibit, but also their ability to express intentional qualities.

Penrose argues in [Pen] that the nondeterminism of physics may be illusory because physical phenomena may be deterministic but noncomputable. He suggest that nondeterminism arises because we limit our physical model to being computable and that a noncomputable model of physics could potentially resolve nondeterminism. Though this argument is wrong in rhe sense that physical phenomena are computable by MIMs, it is nearly right. If Penrose's term "noncomputable" is replaced by the term "not-TM-computable", MIMs can be viewed as the "not-TM-computable" models that Penrose needs.

A MIM is a tool for studying the interaction of k+1 entities, where one plays a special role as an observed system and the remaining entities are observers, modeled by streams. For interacting SIMs, k = 1 and the observer with the observed system form a closed system. When the observed system is a MIM, k is greater than 1 and a primary observer representing the experimenter is distinguished from k-1 secondary observers. MIMs appear nondeterministic from the viewpoint of a primary observer who cannot predict or control the effects of secondary observers and may be unaware of their presence. Primary observer perceive MIMs as subjectively nondeterministic even when, viewed by an omniscient multi-agent observer (God), they are objectively deterministic (Figure 3).

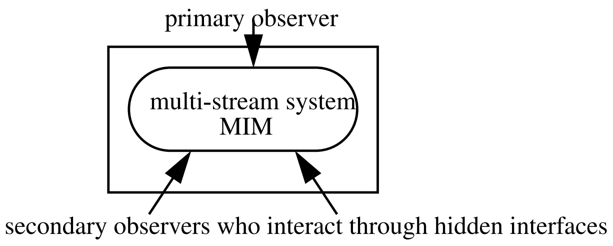


Fig. 3. Interactive Turing Test

In retrospect, Penrose's insight that nondeterminism of physics is due to a weakness of the model rather than an inherent feature of the physical world may turn out to be remarkably close to the truth. His mistake was to accept TM computability as the most powerful form of computing when in fact there are stronger models. By reformulating Penrose's argument in terms of non-TM computability and introducing MIMs as a stronger form of computation, we not only validate Penrose's argument but also demystify it by providing a concrete mechanism and model for the stronger form of computation. Whereas Penrose's hypothesis is not testable, we expect that the hypothesis that MIMs provide a model for quantum theory can be tested.

Penrose's view that computers cannot model physics is based on a misconception of the nature of computing shared by Church and Turing that has its historical roots in the rationalism of Plato and Descartes [We3]. Penrose's identification of computation with Turing computability and Turing's identification of thinking with Turing computability are based on rationalist presuppositions remarkably similar to Plato's model of observation in his parable of the cave [We3].

Plato's cave metaphor, which compares humans to cave dwellers who can observe only shadows (projections) of reality on the walls of their cave and not actual objects in the outside world, is a model of relationships between observers and observed systems that has played an important historical role and is still relevant today. We extend Plato's cave metaphor to cave dwellers who interact with the external world, and show that such interactive observers have the power of IMs and express Platonic empiricism.

Empiricists accept Plato's argument that people perceive incomplete projections P(W) of an external world W on their sensory perceptors. However, they disagree with Plato's conclusion that empirical objects are therefore not worth modeling and that ideal (Platonic) tables are more real than empirically observable tables. Plato's belief that incompletely specifiable objects are not worthy of study, based on a faulty analysis of the notion of abstraction, contributed to a 2000-year delay in the emergence of empirical science.

Empiricism repudiates the requirement of completeness, accepting that complete knowledge is unnecessary because the goals of prediction and control can be achieved entirely through incomplete observable shadows (projections) that are observational abstractions of inherently unobservable reality. The mathematical incompleteness of interactive models parallels the descriptive incompleteness of observed physical objects, while completely specifiable algorithms are the computational analog of Platonic objects. Empirical computer science provides a concrete framework for studying observational abstractions in a context where "inner reality" can be specified and systematically varied. Computational abstractions provide insight into realist models of physics, where inner reality is unobservable and uncontrollable [We3].

Plato understates the information that can be gained by shadows on the walls of a cave. Humans can obtain a variety of different images of their environment through multiple senses (vision, sound, touch). They can obtain multiple simultaneous spatial images through two eyes and two ears to obtain a stereo effect. Multiple temporal images can be obtained by sampling the environment at multiple points of time, creating stereo-spatio temporal images that provide richer cues for image reconstruction than geometric two-dimensional images on the walls of a Platonic cave. The richness of images created in this way has been recognized by "caves" in virtual reality.

We not only can select the images we wish to perceive, but can act to modify the external world and observe the effect of our actions. The process of observing how shadows that represent the state of the world change as a result of our actions can be thought of as "shadow boxing": you can observe, predict, and control the way shadows change when you "punch" them. Shadow boxing may be interpreted as interaction, and occupants of Platonic caves who can shadow-box correspond to full-fledged empiricists.

By showing that interaction machines express empirical computer science, we show that arguments of Penrose, along with those of Church, Turing, and Descartes, are rooted in a common rationalist fallacy concerning the role of non-interactive algorithms in modeling the real world. Penrose's instinct that Turing machines cannot model physics is correct, but his belief that noncomputable models are the key to a unified theory of physical and mental phenomena rests on a limited definition of computability.

Though Searle's and Penrose's criticisms of algorithmic thinking become much weaker when applied to sequential and distributed thinking, the anti-behaviorist argument that inner processes cannot be unambiguously described by behavior remains valid, since behavior is an abstraction of system structure. Since behavior specifications have nonenumerable system implementations, behavior is a weak determinant of system structure. However, the status of behaviorist models is completely changed when thought is modeled by interactive behavior that is extensionally and intentionally richer than algorithmic behavior.

References

[Ac] Peter Aczel, *Non Well-Founded Sets*, CSLI Lecture Notes #14, Stanford, 1988.

[Be] Isiah Berlin, *The Proper Study of Mankind*, anthology edited by Henry Hardy and Roger Hausheer, Farrar, Straus, and Giroux, 1997.

[BM] Jon Barwise and Lawrence Moss, *Vicious Circles*, CSLI Lecture Notes #60, Cambridge University Press, 1996.

[Ch] Peter Chen, The Entity Relationship Model: Towards a Unified View of Data, *Transactions on Database Systems*, 1976.

[Fi] Paul Finsler, *Finsler Set Theory: Platonism and Circularity*, Ed David Booth and Renatus Ziegler, Birkhauser, 1996.

[Go] Daniel Goleman, *Emotional Intelligence*, Bantam paperback, 1997.

[KR] Haim Kilov and James Ross, *Information Modeling: An Object-Oriented Approach*, Prentice Hall, 1994.

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

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

[Pen] Roger Penrose, *The Emperor's New Mind*, Oxford, 1989.

[Pr] Vaughan Pratt, Chu Spaces and their Interpretation as Concurrent Objects, in *Computer Science Today: Recent Trends and Developments*, Ed. Jan van Leeuwen, LNCS #1000, 1995.

[Ru1] Jan Rutten, A Tutorial on Coalgebras and Coinduction, *EATCS Bulletin 62*, 1997.

[Se] John Searle, *Minds, Brains, and Programs*, The Behavioral and Brain Sciences, 1980.

[Tu1] Alan Turing, On Computable Numbers with an Application to the Entscheidungsproblem, *Proc. London Math Soc.*, 2:42, pp. 230-265, 1936.

[Tu2] Alan Turing, Systems of Logic Based on Ordinals, *Proc. London Math. Soc.*, 1939.

[Tu3] Alan Turing, Computing Machinery and Intelligence, *Mind*, 1950.

[We1] Peter Wegner, Why Interaction is More Powerful Than Algorithms, *CACM*, May 1997.

[We2] Peter Wegner, Interactive Foundations of Computing, *Theoretical Computer Science*, Feb. 1998

[We3] Peter Wegner, Towards Empirical Computer Science, *The Monist*, Issue on the Philosophy of Computation, Spring 1999, available at `www.cs.brown.edu/people/pw`.

[We4] Peter Wegner, Interactive Software Technology, *Handbook of Computer Science and Engineering*, CRC Press, December 1996.

[We5] Peter Wegner, Tradeoffs Between Reasoning and Modeling, in *Research Directions in Concurrent Object-Oriented Programming*, Eds. Agha, Wegner, Yonezawa, MIT Press 1993.

[WG] Peter Wegner and Dina Goldin, Mathematical Models of Interactive Computing, `www.cs.brown.edu/people/pw`, December 1998.

[ZM] Stanley Zdonik and David Maier, *Readings in Object-Oriented Database Systems*, Morgan Kaufmann, 1990