

Why interaction is more powerful than algorithms

Peter Wegner

Communications of the ACM, May 1997, pages 81–91



Thesis

- The paradigm shift from algorithms to interaction captures the technology shift from mainframes to workstations, from number-crunching to embedded systems and GUIs, and from procedure-oriented to object-based and distributed programming
- The radical notion that *interactive systems* are more powerful problem-solving engines than algorithms is the basis for a new paradigm for computing technology build around the unifying concept of *interaction*



Analogy 1:

From sale contracts to marriage contracts

- Algorithms are like sale contracts: *algorithm yield outputs completely determined by their inputs*
- Interactive systems are like marriage contracts: *interactive systems provide history-dependent services over time that can learn and adapt to experience*
- Object contracts are like marriage contracts: *object's interface specifies its behavior for all contingencies over the lifetime*
- Interactive tasks cannot be realized through algorithms because they are blind to the environment which defines the interaction



Example differences

- Closed-book exams are algorithmic, only knowledge embodied within the brain are useful; open-book exams are interactive (brain can question the environment which may be infinite)
- Smart bombs are interactive systems because they can check the properties of current environment against the properties stored within their control mechanism and adapt accordingly
- Algorithms are metaphorically dumb and blind because they cannot adapt interactively while they compute
- Objects can remember their past and can interact with their clients through an interface that share a hidden state. Object's operations are not algorithmical; their response to messages depend on shared state

Growing pains

Growing pains of software technology are due to the fact that programming in the large cannot be expressed by or reduced to programming in the small

My note: bad terms; programming as usual should be replaced by "process of problem solving"

Peter's view: behavior of interactive systems (such as airline reservation systems) cannot be expressed by algorithms



No silver bullet for specs

Brook's persuasive argument, is a consequence of the irreducibility of interactive systems to algorithms

Interpretation:

- Silver bullet are (algorithmic) system specification
- Interactive systems cannot be reduced to algorithms
- Hence, no silver bullet can exist. Therefore AI has undergone a shift from logic-based to interactive (agent-based).



What is OO programming?

- Though OO has become dominant practical technology its conceptual framework and theoretical foundation are still unsatisfactory
- Common agreement: everyone talks about it but no one knows what it is
- *Knowing what it is* has proven elusive because of implicit assumption that *explanations must specify "what it is" in terms of algorithms*
- Accepting irreducibility of interactive systems to algorithms has a liberating effect: *what OO is is naturally defined in terms of interactive model*



From TM to IM

- Turing (1936) has shown that TM and programming languages have the same computation power
- Hence, we can identify the class of computable functions with the class of functions computable by TM
- This precise characterization of what can be computed established the respectability of computer science discipline



Turing tarpit

Frustration generated by the inability to compute more than the computable functions by adding new properties to TM

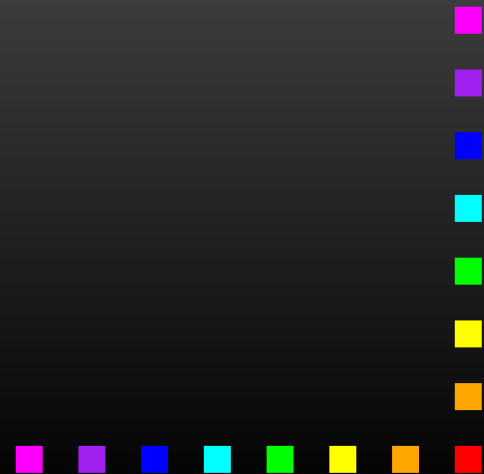
Interactive computing let us escape Turing tarpit

Church thesis: formal notion of computability by TM corresponds to the intuitive notion of what is computable



Note

Church thesis breaks down when computability includes *interactive computability*



IM (interactive machines)

IM are TM extended by addition of input/output actions that support dynamic interaction with an external environment

- IM may have single or multiple I/O streams
- IM may have synchronous or asynchronous I/O actions
- Can also differ along many other dimensions

Note: this small change to TM extends TM computation power



Note

- IM transforms closed systems described by TM into open systems
- IM expresses behavior beyond that computable by algorithms:
 - TM cannot handle passage of time or interactive events that occur during the process of computation
 - Input streams of IM are not expressible by finite tapes, since any finite representation can be dynamically extended by uncontrollable adversaries



Origin of IM

- In 1939, Turing has shown that TM + oracles (like the oracle at Delphi) were more powerful than TM
- In 1975, Milner noticed that concurrent processes cannot be expressed by sequential algorithms
- In 1992, Manna and Pnueli showed that nonterminating reactive processes, like OS-s, cannot be modeled by algorithms
- Gödel's theorem: . . . *integers cannot be described completely through logic*, demonstrates the limitations of mathematical formalism and may be adapted to show that IM cannot be completely described by first order logic



Nature of IM

- Observable behavior of IM is specified by interaction histories describing system behavior over time
- Operations whose effects depend on the time of their occurrence are time stamped
- Objects with inherently nonsequential interfaces (such as joint bank accounts) have inherently nonsequential interfaces
- Interaction history of distributed systems consists of nonsequential events that may have duration and may interfere with each other



Histories

- Interaction histories express external unfolding events in time; instruction histories express an ordering of inner events of an algorithm without any relation to actual passage of time
- Algorithmic time is measured by the number of instructions executed, not by actual time taken by instruction execution; this provides an independent measure of logical-complexity
- Duration of operation execution may be interactively significant
- Operation sequences are interactive systems with temporal as well as functional properties; instruction sequences describe inner state transition semantics



Interacting Identity Machines

- IIMs are machine that output their input.
- Hence, IIM are transducers that realize nonalgorithmic behavior by harnessing the computing power of the environment

- IIMs may be described by:

$P = in(message).out(message).P$ or using a PL by:

`while true do echo input end while`



Note

Though IIMs are not inherently intelligent, they can behave intelligently by replicating intelligent inputs from the environment

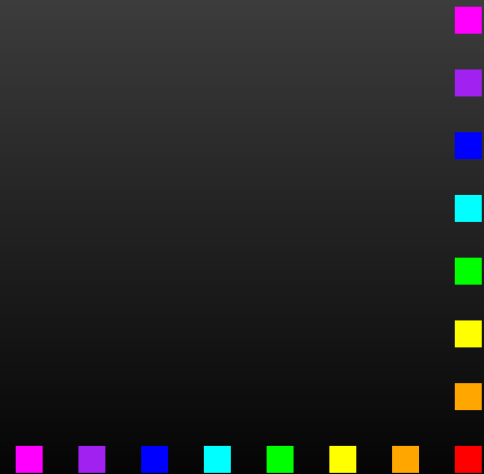
Example: a person ignorant of chess can win half the games in a simultaneous chess match against two masters by echoing the moves of one opponent on the board of the other



Claim

IIM have a richer behavior than TM

Evidence: An IIM can mimic any TM and any input stream from the environment



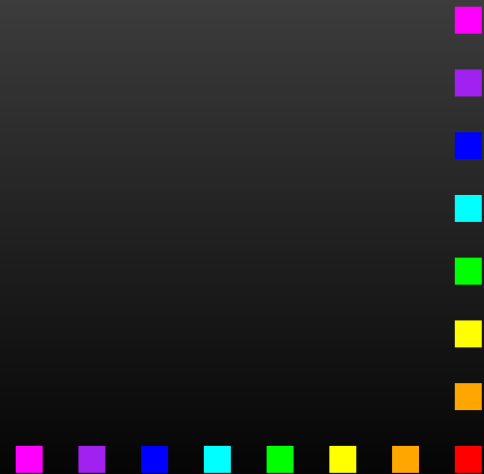
Harnessing the environment

- An ant behavior on a beach cannot be described by algorithms because the set of all possible beaches cannot be described
- The ant behavior reflects the complexity of the beach where nonalgorithmic topography causes the ant to traverse a nonalgorithmic path.
- Interaction opens up limitless possibilities for harnessing the environment and is entirely dependent on external resources; machines with built-in algorithmic behavior are not
- High achievement whether by machines or people, can be realized either by self-sufficient inner cleverness or by harnessing the environment



Note

Interaction scales up to very large problems better than inner cleverness because it expresses delegation and coordination



Interaction, parallelism, distribution

These are conceptually distinct concepts:

- Interactive systems interact with an external environment they cannot control
- Parallelism (concurrency) occurs when computations of a system overlap in time
- Distribution occurs when components of a system are separated geographically or logically

Note: parallel and distributed computation can, in the absence of interaction, be expressed by algorithms



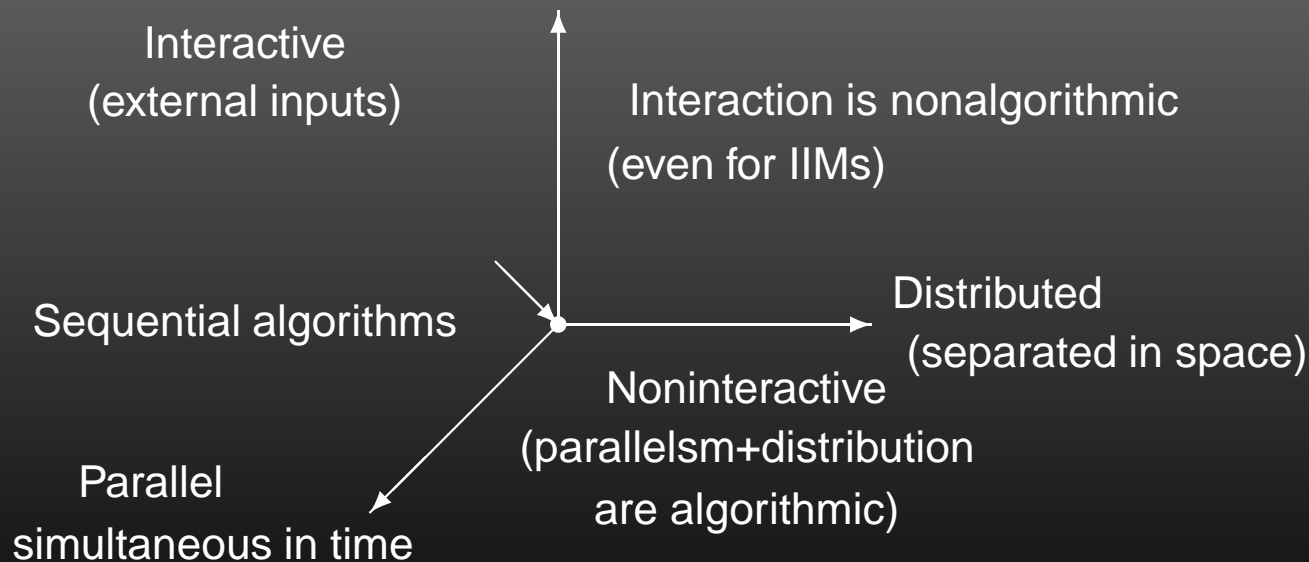
Relationship

- The insight that interaction rather than parallelism or distribution is the key element in providing greater behavioral richness is not trivial
- This requires a fundamental reappraisal of the role of parallelism and distribution in the design of complex systems
- Figure 1 reflects the complex system design space in Wegner's philosophy



Design space

Design space for interactive systems is shown in Figure 1



Example interactive systems

- Agents in the environment can interact in a cooperative, neutral, or malicious way with an interaction machine
- Adversaries with algorithmic goals provide a measure of the limits of interaction machine behavior: synchronous adversaries control *what* input an agent receives; asynchronous adversaries have additional power over *when* an input is received
- Interactiveness provides a natural and precise definition of the notions of open and closed systems.



Open and closed systems

- Open systems can be modeled by IM while closed systems are modeled by algorithms (TM)
- IM provide a precise definition for other fuzzy concepts, such as *empirical computer science* and *programming in the large*
- IM robustly capture many alternative notions of interactive computing just like TM captures algorithmic computing



More on open systems

- Open systems have a rich behavior to handle all possible clients while individual interface demands of clients may often be simple.
- Open systems whose unconstrained behavior is nonalgorithmic can become algorithmic by constraining their interactive behavior summarized as: *supplied behavior of IM versus demanded behavior of a given interface*
- Interfaces are primitive building blocks of IM playing the role of primitive instructions
- Interfaces express the mode of use or pragmatics of an IM, complementing syntax and semantics, see Figure 2



Computational modeling

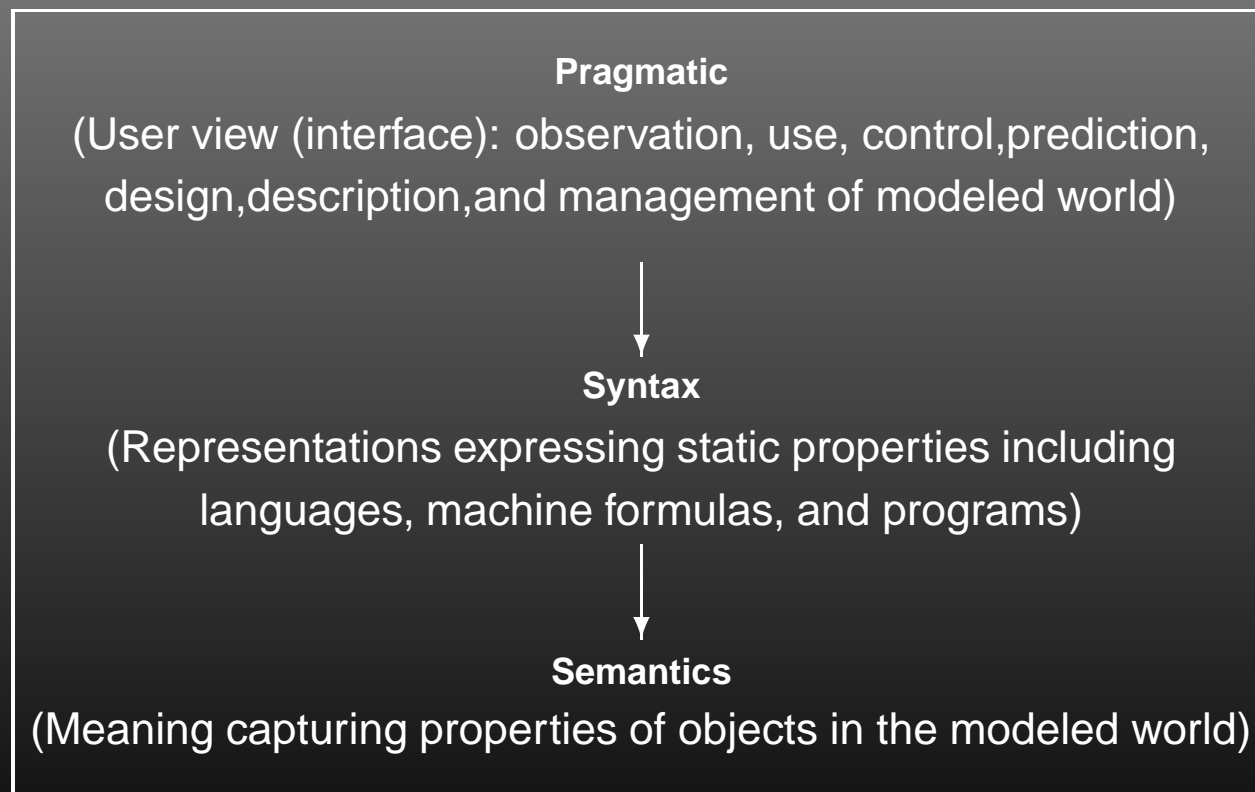


Figure 2: A user view of computation modeling

Complexity

- Closed systems with algorithmic behavior may have open subsystems with nonalgorithmic behavior: example an engine of a car may behave unpredictable when a spark plug is removed
- Animals may behave erratically in unfamiliar environments
- Interactiveness (openness) is nonmonotonic in that decomposition of systems may create interactive unpredictable subsystems. In contrast, concurrency and distribution are monotonic
- Nonmonotonicity is an untidy formal property of of interaction since it implies that noninteractive systems with algorithmic behavior may have interactive subsystems whose behavior is nonalgorithmic



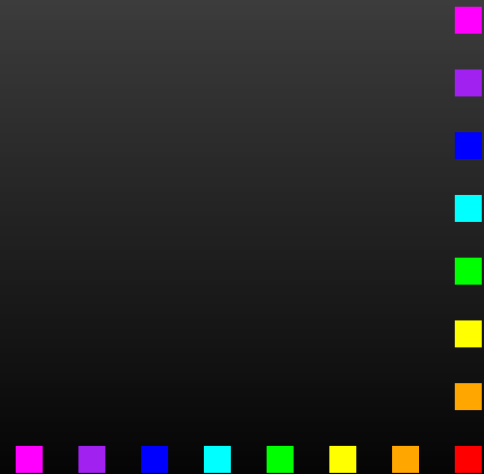
Interfaces as behavior specs

- Negative statement “*interaction is not expressible by algorithms*” give up the goal of complete behavior specification
- Thus it leads to positive new approaches to system modeling in terms of *interfaces*
- Complete specification must be replaced by partial specification of interfaces, views, modes of use



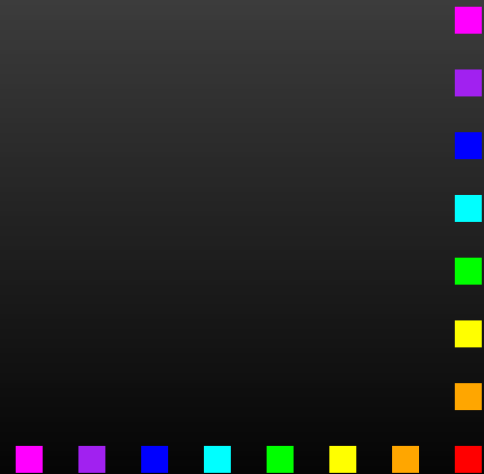
Example

Airline reservation system specified in terms of multiple interfaces: travel agents, passengers, desk employee, etc.



Note

- Description of systems by their mode of use is a starting point for system design
- Interfaces play a practical as well as conceptual role in interactive system technology



Partial versus complete specs

- A system satisfies its requirements if it supports specified modes of use
- Correct behavior for a given mode of use may not be guaranteed
- Complete system behavior for all possible modes of use is unspecifiable

Note

- Though correctness of programs under carefully qualified conditions can still be proved *result checking is needed to verify that results are valid.*
- Result checking is performed automatically by people for such tasks as driving with visual feedback and must be performed explicitly by instruments or programs as safeguards against potential catastrophe generated by embedded systems



Interface descriptions

- Are called *harnesses* because they serve both to constrain system behavior and to harness behavior for useful purposes
- Harnesses have a negative connotation as constraints and positive connotation as specification of useful behavior
- *Open harnesses*: permit interaction through other harnesses
- *Closed harnesses*: cause the system (together with its harnesses) to become closed and thereby algorithmic
- **Example**: The key property of every component of the Microsoft's Component Object Model (COM) is an interface directory allowing access to complete set of interfaces



Shifting goals

- Goal of proving correctness for algorithmic behavior is replaced for interactive systems by the more modest goal of showing that components have collection of interfaces to desired form of useful behavior
- IM specifying software systems are described by multiple interfaces expressing functionality for different purposes
- People are likewise better described as collections of interfaces: the behavioral effect

Plato's rationalism

- **Plato's parable of cave:** compare humans to cave dwellers who observe only shadows of the reality on their cave walls, not actual objects in the real world
- **Plato's conclusion:** abstract ideas are more perfect (than shadows) and therefore more real than physical objects represented by shadows
- **Result:** Plato's skepticism concerning empirical science contributed to 2,000-year hiatus in the evolution of empiricism



Empiricism

- Empiricists accept the view that perceptions (the shades) are reflections of reality but disagree with Plato on the nature of reality
- Empiricist believe that the physical world (rather than abstract ideas) outside the cave is real but unknowable
- Fortunately, complete knowledge is unnecessary for empirical models of physics because they achieve their pragmatic goals of prediction and control by dealing with incomplete observable reflections



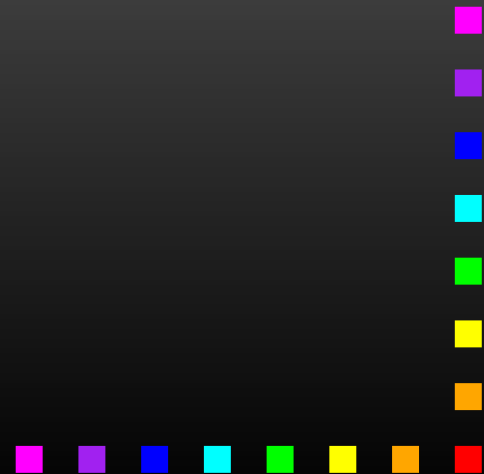
Modern empirical science

- rejects Plato's belief that incomplete knowledge is worthless
- uses partial descriptions (shadows) to control, predict, and understand the objects that shadows represent
- **Examples:**
 1. differential equations capture quantitative properties of phenomena they model without requiring a complete description;
 2. computing systems can be specified by interfaces describing relevant properties while ignoring other properties



Note

Plato's cave, properly interpreted, is a metaphor for empirical abstraction in both natural science and computer science

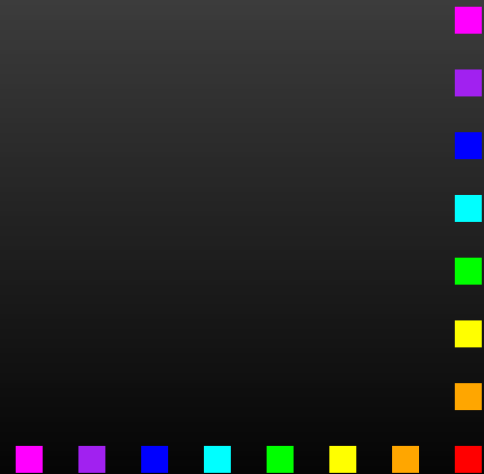


Computer science interpretation

- TM correspond to Platonic ideals by focusing on mathematical models at the expense of empirical models
- To realize logical completeness, TM sacrifice the ability to model external interaction and real time
- Extension from TM to IM is computational analog of liberation from Platonic world view that led to development of empirical science.
- IM liberate computing from the Turing tarpit of algorithmic computation providing a conceptual model for software engineering, AI agents, and the real physical world

Rationalism versus empiricism

- Contrast between algorithmic and interactive models parallels interdisciplinary distinctions in other domains of modeling.
- The purest form of this contrast manifests in philosophy



Representatives

1. Descartes: *cogito ergo sum*, i.e., thinking is the basis of existence, implying that certain knowledge of the physical world is possible through inner process of algorithmic thinking
2. Kant: in *Critique of Pure Reason* shows that pure reasoning about necessarily true knowledge is inadequate to express contingently true knowledge of real world
3. Hegel: by dialectical logic extended reason beyond its legitimate domain influencing political philosophers (Marx) and mathematical thinkers (Russell)
4. George Boole: in his treatise *Laws of Thought* has equated logic with thought



Note

- Mathematical thought in early 20-th century was dominated by Russell's and Hilbert's rationalist reductions of mathematics to logic
- Gödel's incompleteness result showed the flaws of rationalist mathematics
- Logicians and mathematicians have not fully appreciated the limitations of formalism implied by Gödel's work



More on rationalism

- Though empiricism has displaced rationalism in sciences, TM reflect rationalist reasoning paradigms of logic rather than empirical paradigms of physics
- Algorithms and TM shut out the world during the process of problem solving
- The effect of Gödel's incompleteness theorem were slow to manifest among such logicians as Church, Curry, Turing, who shaped the foundations of computer science



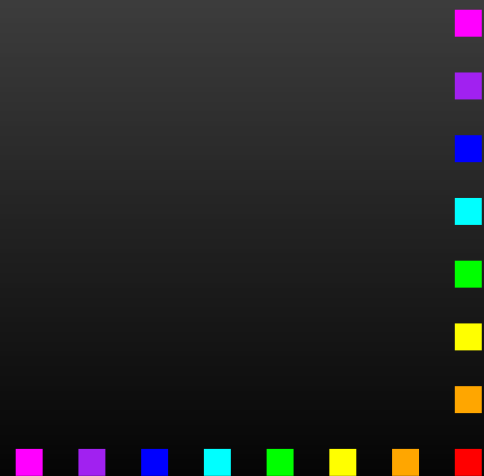
Abstraction and incompleteness

- Abstraction is the key tool in simplifying systems by focusing on relevant attributes and ignoring irrelevant ones.
- Incompleteness is the key distinguishing mechanism between rationalist, algorithmic abstraction and empiricist, interactive abstraction.
- Completeness and predictability of algorithms are inherently inadequate in modeling interactive computing tasks and physical systems
- Sacrifice of completeness is frightening to theorist working on formal models



More on abstraction and incompleteness

- Incomplete behavior is comfortable to physicists and empirical model builders
- Incompleteness is the essential ingredient distinguishing interactive from algorithmic models of computing and empirical from rationalist models of the physical world



Models

- Models in logic and computation aim to capture semantic properties of a domain of discourse by *syntactic representations for the pragmatic benefit of users*
- A model is a tuple $M = (W, R, I : W \rightarrow R)$ where:
 1. W is the modeled world (domain of discourse)
 2. R is a representation of a modeled world W
 3. I is an interpreter that determines the semantic properties of W in terms of syntactic expressions of R



Note

- Interactive models have multiple pragmatic models of use
- Algorithms have a single intended pragmatic representation determined by the syntax
- The goal of expressing semantics by syntax is replaced by the interactive goal of expressing semantics by multiple pragmatic models of use
- The goal of complete behavior specification is replaced by the goal of harnessing useful forms of partial behavior through interfaces

Modeling interactive systems

- Incompleteness of interactive systems has the (practical) consequence that *maximal goals of logic, functional programming, and formal methods cannot be achieved*
- Japanese fifth-generation computing project of 1990-s failed because logic programming is too weak to model interactive systems
- Algorithms and logical formulae take their meanings in the same semantic world of computable functions
- Well-formed formulae are semantically interpreted as assertions about modeled world; formulae true in all interpretations are called tautologies



Soundness and completeness

- A logic is sound if all syntactic provable formulae are tautologies (any wff can be evaluated)
- A logic is complete if all tautologies are provable

Note: soundness and completeness measure the adequacy of syntactic proofs in expressing semantic meaning. They relate syntactic representation R of a logical model to the semantic modeled world W



Note

- *Soundness* implies that syntactically proved theorems express meaning in the modeled world
- *Completeness* implies that all meanings can be syntactically captured as theorems

Soundness ensures that representations correctly model behavior of modeled world while completeness ensures that all possible behavior is modeled



Correctness

Soundness and completeness ensure that a representation is correct and that it captures all behavior in the modeled world

Note:

- completeness restricts the kind of modeled worlds so that only worlds whose semantic is completely expressible by syntax can be expressed
- soundness ensures that proofs are semantically correct while completeness measures the comprehensiveness of the proof system



Conclusions

- Soundness and completeness together imply that W is reducible to R (i.e., W and R are isomorphic abstractions)
- Reducibility of W to R implies completeness of R in expressing properties of W
- Incompleteness implies irreducibility of W to R
- Soundness and completeness are often abandoned for practical reasons. **Example:** semantic notion of error in dynamically executed programs cannot be statically defined
- Semantic notion of error can be syntactically approximated. In choosing the approximation conservatism of soundness and permissiveness of completeness are avoided by compromise

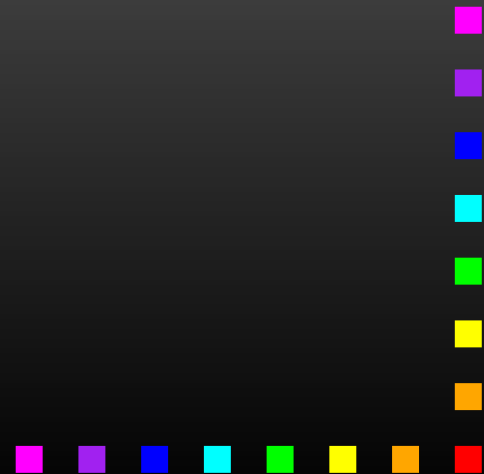
Incompleteness

- Gödel's incompleteness theorem in arithmetic has an analog in computing
- The key property of incomplete domains is irreducibility
- Completeness is possible only for a restricted class of relatively trivial logics over semantics domains reducible to syntax
- Completeness restricts behavior to that describable by algorithmic proof rules
- Models of real world sacrifice completeness in order to express autonomous (external) meanings



Note

Incompleteness is a necessary price for modeling independent domain of discourse whose properties are richer than the syntactic notation by which they are modeled



Decidability

- The set of true statements of a sound and complete logic can be enumerated and therefore is recursive enumerable
- Gödel showed incompleteness of integers by showing that the set of true statements about integers is not recursive enumerable, using diagonalization
- Incompleteness of IM can be proved by showing that the set of computations of an IM cannot be enumerated

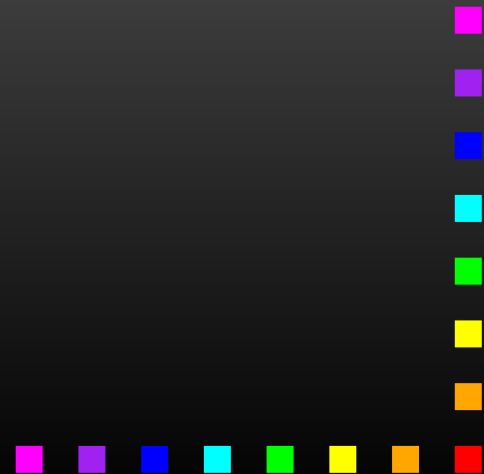


Incompleteness of IM

- The set of inputs of an IM has the same cardinality as the set of infinite sequences over a finite alphabet
- The set of infinite sequences over a finite alphabet is not recursive enumerable
- Hence, the set of computations performed by an IM is not recursive enumerable

Consequence

Proving correctness of interactive models is impossible. This parallels the impossibility of realizing Russell and Hilbert's programs of reducing mathematics to logic



Adjustments

- Goal of research on formal methods must be modified to acknowledge this impossibility
- Proofs of existence of correct behavior (type I correctness) are in many cases possible
- Proofs of nonexistence of incorrect behavior (type II correctness) are generally impossible

