Models of Interaction

Peter Wegner, Brown University
pw@cs.brown.edu
Dina Goldin, Umass Boston
dqg@cs.umb.edu

ECOOP '99 Tutorial
June 1999

What is Object-Oriented Programming?

There is no silver bullet for software engineering
Fred Brooks, early 1970s

Everyone is talking about OOP, no one knows what it is?
Tim Rentsch, 1979

Reactive systems cannot be modeled by algorithms
Zohar Manna and Amir Pnueli, 1980s

Concurrent systems cannot be modeled by algorithms
Robin Milner, 1992

Many signs that Turing machines (TMs) cannot model OOP
knowing what OO is: not possible by algorithmic models
OO can be defined only by interactive models of computation

Goal:
egative: show OOP is not expressible by, reducible to TMs
positive: unifying model for OOP, AI, networks, graphics, HCI

The Evolution of Programming Paradigms

Paradigm shift from the 1970s to the 1990s:
From mainframes to workstations and networks
From number crunching to embedded systems and GUIs
From procedures to objects and distributed systems
Fundamental shift from algorithms to interactive computing

1950s - 1960s: Machine language -> procedure-oriented
change in the granularity of actions (scale)
1970s - 1990s: Procedure-oriented -> object-based
fundamental change in modeling power (quality)

From Algorithms to Interaction

A procedure transforms arguments (inputs) to values (output)
sequence of steps of procedure is an algorithm
behavior is specified as a function
An algorithm is a computable function from integers to integers
f: X -> Y, where x∈X is completely defined prior to start

Time-independent spec: indep of start time and execution time complexity:
depends on number of instrs but not on time

Two modes of computing:
algorithms: computable functions, transformation semantics
sales contracts: given an input provide an output
interaction: services over time, observation semantics
marriage contracts, not expressible by sales contracts
formalize distinction between marriage and sales contracts
Algorithms (verbs) are less expressive than objects (nouns).
Nonalgorithmic Object Behavior

An object has operations that share a state embedded system, service over time for unpredictable clients.

Two kinds of interaction for an object’s operations:
- interaction with input streams
- interaction with a shared state

op1 is not a fixed function, action changes between activations
if parallel access to op1, op2, then op2 can interfere with op1
op2 can change effect of op1 while it is executing

Sharing of state causes nonfunctional behavior of operations
shared object variables are essential, but formally harmful
Sharing is harmful to formalization, but increases expressiveness trade-off between formalization and expressiveness
Go tos were considered harmful to formalization, Dijkstra allowing go tos merely increases flexibility, not expressiveness
Interaction is more fundamentally harmful to formalization
interaction increases expressiveness, solves more problems

Programming in the Large (PIL) Is Interactive

What is “large” in Programming in the Large?
largeness does not mean a large number of instructions
algorithms with a million instructions are not necessarily PIL
PIL programs are, however necessarily interactive
PIL = interactive programming
this definition of PIL is crisp and natural
replaces fuzzy largeness by a precise testable criterion

PIL and PIS are qualitatively (not quantitatively) different
differ in expressive power, not just in size

Fred Brooks’ “no silver bullet for PIL” translates into
“PIL is inherently nonalgorithmic, nonformalizable”
no silver bullet = no algorithmic model
Greater expressiveness of interactive over algorithmic systems explains why SE is not primarily about algorithms
explains difficulty of extrapolating from PIS to PIL
explains diff between proc-oriented and OO programming

Many Alternative Forms of Interaction

Quote from Christos Papadimitriou
In computer science, important concepts usually come in many alternative characterizations

Christos was referring to many forms of NP-completeness
also many alternative models for computability, interaction

Computability: TMs, λ-calculus, recursively enumerable sets
interaction can also be characterized in many different ways

Parallel Extensions from Algorithms to Interaction

Interactive Models (machines, sets, algebras)

Interactive Models (machines, sets, algebras)
Turing machines -> interaction machines
recursively enumerable sets -> non-well-founded sets
λ-calculus -> coalgebras
inductive reasoning -> circular reasoning (coinduction)

Multiple Models (Projections) of Shared Domain

Computer Science has multiple models of computation:
machines: rules of computation, a mechanism for realizing them
grammars: sets of strings, generating rules
logics: relate syntactic inference to semantic modeling

Same underlying semantics expressed by different models
interactive models likewise have many alternative forms

The real world has multiple models with weak interaction:
physics: matter and motion; for example, the solar system
chemistry: atoms and molecules; chemical reactions
biology: living organisms; cells, protein folding, evolution
economics: production and distribution of goods and services
literature: literary interpretation of texts, deconstruction
art: visually represented modeled worlds, human models

Multiple models of real world is basis for science
chemistry, biology; different projections of the real world
models are also central to economics, literature, art
Multiple Models of Object-Based Design

The object modeling technique OMT has a three-level model three levels of analysis of interactive software systems

**object model:** relation among interactive components (nouns) nonformalizable static description of interaction patterns

**dynamic model:** specific sequential interaction histories
dynamic partial inter-object behavior for sequential steam

**functional model:** transformation behavior of specific functions
dynamic intra-object behavior at the level of algorithms

---

**Interactivity Models**

Parallel levels of interactive and algorithmic abstraction
static model: object model \( \rightarrow \) flow diagram
dynamic model: interaction history \( \leftrightarrow \) execution history

Greater gap between static and dynamic structure

two levels of dynamic modeling: outer events, inner rules
interactive operation sequences composed of instr sequences
interactive events are second order, built from first-order events

---

**Interaction Machines**

IMs extend TMs: incremental input streams from environment
I/O stream \((i_1, o_1), (i_2, o_2), \ldots, (i_{k+1})\) can depend on output \(o_k\)

Sequential IMs (SIMs): interact with single I/O stream
Multi-stream IMs (MIMs): interact with k streams, \(k > 1\)

A SIM is a machine \(M = (S, I, F)\)
S is an enumerable set of states
I is an enumerable set of input strings
F: \(S \times I \rightarrow S \times O\) is a computable function

*computation step is a complete TM computation*

The transition from \(s_k\) to \(s_k\): atomic I/O pair \((i_k, o_k)\)

Input nondeterminism: unpredictable dynamic inputs \(i_k\)
Output determinism: \(o_k\) is determined by \(i_k\)
can easily be extended so that output is nondeterministic

SIMs: single-user workstations and databases, Markov processes
MIMs: ATMs, distributed systems, networks, the Internet

---

**Properties of Interaction Machines**

Finite computing agent:
the computing agent is finitely specifiable

Dynamic (late, lazy, incremental) binding of inputs
\(i_{k+1}\) may depend on \(o_k\) and on external events

Persistence of state:
initial state for each interaction is the previous final state

History dependence:
output can depend on previous I/O tokens via the state

Hidden information:
state is unknown to the observer, circular reasoning

No halting state:
computation is a continuing process, not a transformation

Behavior of IMs is modeled by I/O streams
IMs (streams) express nonenumerable possible behaviors

Streams are mathematically modeled by non-well-founded sets
axiomatic set theory provides a model for streams

SIMs interact with a single stream, MIMs with multiple streams
MIMs are more expressive than SIMs
Interactive Identity Machines (IIMs)

Pure interaction without transformation is nonalgorithmic interactive identity machines, transduce without transforming

```
loop "interactive-identity-machine"
  input(message); output(message);
end loop
```

Interactive chess machine M:

- use intelligence of player A against player B

Ant on a beach finding way home to ant colony

Set of all beaches cannot be described algorithmically nonalgorithmic behavior, complexity determined by beach

Management paradigm: harness behavior of environment managers are as powerful as their workers interaction machines can harness power of the environment potentially more powerful than Turing machines at the cost of heavy dependence on the environment

---

Concurrency, Distribution, and Interaction

Parallel computations are overlapping (simultaneous) in time

Distributed computations are separated in space

Interactive computations have inputs distributed in time

Parallel noninteractive computations are algorithmic textbooks, graduate courses, practical applications

Distributed noninteractive computations are also algorithmic the design space is algorithmic in the horizontal base plane

Focus on interaction instead of concurrency changes perspective transactions: concurrency control to interaction control processes: concurrent composition -> interactive composition distinguish interactive from execution concurrency

---

Multi-Stream Interaction Machines (MIMs)

Finite agents that interact with multiple streams

ATM systems, distributed databases, collaborative computing MIMs express n-agent interaction, n>2

SIMs model 2-body problem, MIMs n-body problem, n>2

physics: 2-body problem is tractable, 3-body problem is not

MIMs precisely define distributed systems a system is distributed if its interaction is inherently concurrent

Claim: MIMs are more expressive than SIMs interactive expressiveness: observation (problem-solving) power two technical arguments for greater expressiveness:

1. nondeterministic behavior of multiple autonomous observers

MIM observers: hidden interfaces, secondary observers observed objects are connected to, modified by environment

MIM observers can observe only partial system behavior observed objects interact with hidden secondary observers

MIM observers cause ND, eg random noise observer perceives subjective ND, but objectively deterministic compatible with Einstein’s belief that God does not play dice Einstein’s intuition was correct, but his example was wrong observed objects in real world are MIMs, not SIMs

Testable hypothesis: extend Aspect experiment to 3-bodies need 3 bodies (2 streams) for ND by secondary observers

Computationally this model is the multiple writers problem multiple-writers cause nondeterminism for databases airline reservation systems are nondeterministic in this way

MIM models: collaboration, coordination, higher management

---

Does God Play Dice?

Einstein: quantum nondeterminism (ND) due to hidden variables partial knowledge causes quantum ND, God does not play dice Bohr, Bell, Aspect proved hidden variable model inadequate concluded ND is inherent part of reality, God plays dice

MIM observers: hidden interfaces, secondary observers observed objects are connected to, modified by environment

Secondary observers cause ND, eg random noise observer perceives subjective ND, but objectively deterministic compatible with Einstein’s belief that God does not play dice Einstein’s intuition was correct, but his example was wrong observed objects in real world are MIMs, not SIMs

Testable hypothesis: extend Aspect experiment to 3-bodies need 3 bodies (2 streams) for ND by secondary observers

Computationally this model is the multiple writers problem multiple-writers cause nondeterminism for databases airline reservation systems are nondeterministic in this way

MIM models: collaboration, coordination, higher management
The Turing Test

**Turing test**: experiment to test “Can machines think?”
depends on the meaning of “machine”, “think”, “can”

Machine simulates human responses to broad range of questions
Turing: machines think if their behavior is indistinguishable

Turing hypothesis in 1950: machines can in principle think
they will in time be able to pass the Turing test

Details: what kind of a machine?
Turing naturally assumed that machines were Turing machines
allowed machines to delay answer to simulate human slowness
but no provision for inherently slower tasks, scene recognition

Critics (Penrose and Searle):
- **extensional skeptics**: machines cannot behave like thinkers
- **intensional skeptics**: even if they can, they lack awareness

Two kinds of machines: TMs, IMs
Can Turing machines think? - no, thinking includes interaction
Can interaction machines think? - maybe, plausibly yes

The Interactive Turing Test

Turing’s two main contributions: TMs and the Turing test
part of a comprehensive unified model of computation
computational limitations of TMs limit the Turing test
does not handle SIM or MIM models of computation

Extend Turing test to interactive behavior (interactive thinking):
sequential thinking: history-dependent behavior of SIMs
sequential Turing test: SIMs have more realistic behavior
distributed thinking: multi-agent behavior of MIMs
distributed Turing test: MIMs have even more realistic behavior

SIMs remember answers, questioners ask follow-up questions
Starr can learn more about Clinton with follow-up questions

MIMs can delegate to experts hidden from the questioners
encyclopedia Britannica, human or chess expert
SIMs can do better than TMs, though not as well as MIMs

Searle was right that TMs are an inadequate model of thought
but wrong that behavior cannot inherently model thought
SIMs and MIMs can model intentionality, though TMs cannot
Searle was right for the wrong reasons
Turing, though wrong, was right in his experimental method

Penrose’s Model of Physics and Thought

Penrose: physical world is deterministic but noncomputable
all computable models of physics are nondeterministic, but
∃ noncomputable deterministic models of physics and thought

Two interpretations of Penrose’s view:
1. Penrose is wrong: IMs are computable deterministic models
IMs model extensional behavior of physics and thought
2. Penrose is nearly right: IMs are non-TM computable
∃ non-TM computable det. models of physics and thought

Penrose viewed noncomputability as mystical, undefined
however non-TM computable is defined by IM models
non-TM models express physics and thought

If we replace “noncomputable” by “non TM computable”
then MIMs have the properties Penrose postulates
MIMs are ND for TM and SIM observers
objectively deterministic for an omniscient observer (God)

Computational reasoning provides insight into physics
resolves Einstein-Bohr controversy, Penrose dilemma
physics provides a computational model of the real world
interactive models in turn provide a framework for physics

Correctness and Testing Specifications

A specification Sp defines behavior for observation set O.
Algorithms: elements of O are single I/O observations.
SIM specifications extend O to streams of I/O observations.

**Correctness** requires O to account for all system behavior
**Testing**: subset of O that determines partial system behavior

Incompleteness: complete behavior O not formally specifiable
Algorithms have an absolute correctness spec for all behavior
even though correctness may be undecidable, unverifiable

Interactive systems have only partial specifications of behavior
relative correctness: no absolute correctness even in principle
correctness is definable only relative to a class of observations
Correctness can be approximated by progressively finer OEs
sets of all k-step observations for SIMs
but k-step OEs do not capture complete behavior for any k
Interactive correctness cannot be formally specified by fol
only testing specifications are possible

Dijkstra: correctness shows only presence of bugs, not absence
used as an argument in favor of correctness proofs
but for interactive systems testing is the only game in town
**Specification**

Specification language: class of things that may be specified

First-order logic: specifies algorithms, computable functions

Non-well-founded set theory: specifies sequential interaction

Consistency: the property that a system satisfies a specification

Specification $Sp$ defined by a class $O_{Sp}$ of observations

System $S$ defined by class $O_S$ of observable behaviors

System $S$ is consistent with $Sp$ if $O_{Sp} \subseteq O_S$, complete if $O_{Sp}=O_S$

System $S$ consistent with $Sp$ can be viewed as possible worlds

Software Specification and the Associated Class $C_{Sp}$ of Consistent Systems (Possible Worlds)

Compare fol and nwfst as specification languages:

- fol specifications: $C_{Sp}$ is a class of TMs
- nwfst specifications: $C_{Sp}$ is a class of SIMs, stronger formalism
- MIMs require an even stronger specification formalism

**Interactive Expressiveness of Machines**

Expressiveness of algorithms: transformation power definable by classes of sets, regular, context-free, etc recursively enumerable sets express computable functions

Expressiveness of objects, agents: observation power ability to make distinctions in the environment

Environment: set of strings for TMs, streams for SIMs

SIMs can make finer environment distinctions than TMs

MIMs can make finer environment distinctions than SIMs

Classes of specifications of sets, equivalence classes of an equivalence relation

M$_1$, M$_2$ can be distinguished in $E$ if $d_e \in DS$ exists
distinguishable machines have a finite distinguishability certificates

M$_1$ is more expressive on $E$ than M$_2$ if $B_E(M_1) < B_E(M_2)$

M$_1$ can make finer distinctions on $E$ than M$_2$

**Observation Power of Finite Computing Agents**

The number of equivalence classes of an ER is called its index

Lemma for finite automata (Myhill, Nerode)

ERs induced on tapes (strings) by FAs have a finite index

strings are equivalent for state $s$ if they compute same final state

strings equivalent for all states $s$ are indistinguishable

Extension of this lemma to TMs and IMs

ERs induced on strings by TMs have an enumerable index

ERs induced on environs by IMs have nonenumerable index

Interactive finite agents can make nonenumerable distinctions about their environments (surprising result)

Proof: I/O streams are nonenumerable (cardinality of reals) streams are modeled by non-well-founded sets

This “proves” that IMs are more expressive than TMs

grounds interactive expressiveness in classical automata theory

relates observation-based expressiveness to automata theory

IMs model the real world by the real numbers

For FAs and TMs, domain = enumerable environment $X$, $f:X \rightarrow Y$

FAs make enumerable distinctions, TMs nonenumerable ones for IMs, domain is nonenumerable streams

SIMs partition ENV into nonenumerable classes (streams)

MIMs require more complex definition of observation

**Classes of Agents and Environments**

Classes of agents (machines) with behaviors: TMs, SIMs, MIMs

Classes of environments:

- Turing machines TM (interaction specified by I/O pairs)
- SIMs (interaction specified by I/O streams)
- MIMs (interaction specified by multiple I/O streams)
- the physical world $W$ (at least as demanding as MIMs)

Sequential interaction: producer/consumer

agent (machine) consumes behavior produced by environment

actual behavior: intersection of agent, environment behavior

Real world $W$ is the strongest environment considered

$B_W(M)$ is a measure of the expressiveness of $M$

Expressiveness result: $B_W(TMs) < B_W(SIMs) < B_W(MIMs)$

But if the environment is constrained to be a TM:

$B_{TM}(TMs) = B_{TM}(SIMs) = B_{TM}(MIMs)$

Or if the agent is constrained to be a TM:

$B_{TM}(TMs) = B_{SIM}(TMs) = B_{MIM}(TMs) = B_{W}(TMs)$

Infinite expressiveness hierarchy for SIMs:

Let SIM$_k$ be class of SIMs with $k$ interactions

$B_W(SIM_k) < B_W(SIM_{k+1})$, for all $k \geq 0$

$B_W(TM) = B_W(SIM_1)$, TMs are at the bottom of the hierarchy
High-Level Questions about Computer Science

What is a computational problem?
What are the basic models of computation?
What is a good intuitive notion of computation?
What is expressiveness, how should it be modeled?
What are the goals of computer science?

Traditional answers:
problem: algorithm, traveling salesman, sorting
models: Turing machines, strings, grammars, logic
intuition: Church’s thesis, intuitive computing = TM computing
expressiveness: transformation power, recognition power
goals: complexity, quantitative analysis of algorithms

Broader view:
problem: services over time, objects, agents, embedded systems
models: interaction machines, streams, well-founded sets
intuition: beyond Church’s thesis, manage interaction over time
expressiveness: external modeling power, environment capture
goals: modeling reactive, embedded, interactive applications

Evolution:
solve a larger set of problems
more ambitious set of technological goals
different foundational models
from reasoning to modeling
parallel extensions: machines, set theory, algebra

Declarative, Imperative, and Interactive Models

Computing: lingua franca for modeling many different domains
predicate calculus: models domain-independent notion of truth
declarative: functional and logic programming, relations
imperative: algorithms, computable functions, TMs
interactive paradigm: services over time, objects, agents

Declarative, imperative, interactive models of computation

What is a Model?
A model represents, describes, implements something
it abstracts “relevant” properties, ignores “irrelevant” properties
Visual models: partial description, model airplane, car, building
behavior models (simulate, implement behavior)
differential equation, airline reservation models

A model expresses semantic properties of a modeled world W
by syntactic properties of a representation R

Goal of logic models: to capture semantics syntactically
empirical models have different goal: express external reality

Soundness and completeness: measure adequacy of R for W
soundness: R reliably expresses specific properties of W
completeness: all properties of W can be expressed by R

Godel: first-order logic is incomplete for arithmetic over integers
semantic domains of mathematics are too rich for fol models
can only enumerate, inductively specified domains
fol is incomplete for interaction domains for the same reasons
fol is too weak to model mathematics or computation

Declarative Models

Syntax: representation R
Semantics: world W

Declarative models: first-order logic
well-formed formulae
model theory domain
computer functions
transition rules

Imperative models:
lambda calculus
computable functions
transition rules

Interactive models:
procedural paradigm
computable functions
transition rules

Pragmatics: Partial Description Relative to Specific Mode of Use
Mathematical model: logic, algorithms:
modeled world of functions and predicates
automatic derivation by rules, noninteractive interpreters
prescribed single pragmatics identified with syntax

Empirical model: physics, objects, components, interaction
modeled world is the real world or an artificial world
derivation by interaction with external events
multiple pragmatics sharing a common syntax and semantics
grounded in an external reality that imposes external semantics

Pragmatics relates modeled worlds (and objects) to modes of use
uses of simple objects, algorithms, systems may be complex
empirical models describe patterns of observation, simulation
multiple pragmatics enhance abstraction and description power

ECOOP Tutorial 25/36
Plato's Cave: Incomplete Physical Models

People observe only shadows of reality on the walls of a cave projection of light on our retina, incomplete cues of reality

Plato: abstract ideas are more real than empirical reality ideal table in heaven more "real" than physical tables reality is mathematical, physics cannot give certain knowledge denial of validity of empirical observation fundamental mistake, harmful consequences obstacle to the development of science for 2000 years

We agree reality is unknowable, knowledge is incomplete disagree with Plato that partial knowledge is worthless fortunately understanding, prediction, control are possible coping with incompleteness: the basis for empiricism

Dwelling in a cave is not so bad, reflects reality we deal with inherently incomplete knowledge all the time when we talk to each other, no knowledge of inner thoughts coinductive models express interacting computers and people

History of Modeling

Models aim to formalize or mechanize intuitions about domains math models formalize semantics by syntax physics models describe, predict, control physical behavior

History of modeling dates back to the Greeks Presocratic models: Thales, Heraclitus, Parmenides, Democritus Plato’s cave: observe only shadows (reflections) of reality Euclidean geometry: geometric model of reality

Rationalism (Descartes), Empiricism (Locke, Berkeley, Hume) “Cogito ergo sum”, thinking implies existence certain knowledge only through inner algorithmic thinking Hume: inductive inference and causality are not deductive Kant: Critique of Pure Reason, logic is inadequate for modeling Rationalist revivals in 19th and 20th century Hegel, Marx, extend reason beyond its legitimate domain Russell, Hilbert, formalize mathematics by logic Godel, logic cannot completely model mathematics overturns Hilbert’s formalism Church-Turing models arise in 1930s, just after Godel Church’s thesis: computational analog of Hilbert formalism Godel incompleteness applies also Church’s thesis

Godel Incompleteness

Logics prove theorems from axioms by rules of inference theorems provable from axioms are recursively enumerable

Soundness (S) and completeness (C) relate syntax to semantics syntactic proofs model truth in a semantic domain sound if all theorems are true, complete if all truths are thms

Completeness restricts the richness of modeled domains can model only model domains where all truths are theorems # or true properties of domain cannot exceed # of theorems

Proposition: S+C logics can model only RE # of properties

Proof: cardinality of properties = cardinality of theorems = RE

Corollary: Domains with non RE # of properties are incomplete

This result is part of folklore of logic, easy to prove

Godel: arithmetic over integers has non RE # of properties harder to prove: diagonalization shows non RE # of properties

General method: show incompleteness by proving non RE Program equivalence is not RE, not formalizable by S+C logic IMs: non RE # of computations, cardinality of infinite bit strings behavior of IMs cannot be completely described by S+C logic

In contrast algorithms have an RE # of computations behavior is describable by S+C logic

Implications of Incompleteness

Logic is a weak modeling tool, can model only limited domains limitation: domains must be noninteractive, closed, monotonic

Logic is a weak specification tool, cannot specify IMs cannot specify complete behavior, only partial behavior correctness is not definable, testing specifications are

Specification: observation equivalence for class of observations refine specification by enlarging set of observations interactive specs cannot be formally defined - more later

Completeness requires semantics to be isomorphic to syntax incomplete systems can express semantics beyond syntax

General setting for Godel: non RE iff incomplete incompleteness of maths -> incompleteness of computing -> descriptive incompleteness of behavior specifications

Incompleteness is stronger than undecidability noncomputability of fixed points -> nonexistence of fixed points

Connection between logical and descriptive incompleteness incomplete systems have no complete description, specification
Church’s Thesis

Church’s thesis: conjecture about intuitive notion of computing
the intuitive notion X corresponds to the formal notion Y
X = algorithmic computability, Y = TMs, lambda calculus

Stronger thesis for sequential interaction:
X = single-stream interaction, Y = SIMs, PTMs

Still stronger thesis for multi-stream interaction:
X = general interactive computing, Y = MIMs, coinduction

Church-style theses are completeness theorems:
TMs completely specify algorithmic computing
SIMs completely specify sequential interactive computing
MIMs completely specify multi-stream interactive computing

Incompleteness theorems:
TMs are incomplete for SIM computations and behavior
SIMs are incomplete for MIM computations and behavior

Church’s Thesis and Quantum Computers

Intuitive notion of physically realizable computation:
physically realizable (PR): polynomial time and space resources

Physical Church Thesis (PCT):
X = PR computations, Y = TMs with polynomial resources
intuitive characterization for subclass of TM computations
PR-computable functions is subclass of TM computable functions

Quantum TMs: larger class of PR-computable functions than TMs
quantum PR prime factorization, no known PR TM algorithm
quantum TMs appear to violate physical Church thesis

But quantum TMs are noninteractive, only single observations
PCT relates intuition to formal sub-TM models
interactive CT relates intuition to formal super-TM models

Church’s thesis relates intuition to formalism
was influenced by Brouwer’s intuitionism (1920s)
debate between Brouwer (intuitionism) and Hilbert (formalism)

Debates in the 1920s influenced both Church and Turing
Godel incompleteness (1930) provoked a crisis
to which both Church and Turing reacted

Logicism, Intuitionism, Formalism, Realism

Intense debates in the 1920s on the nature of mathematics
Logicism (Russell): mathematics is reducible to logic
Intuitionism (Brouwer): formalism does not capture intuition
Formalism (Hilbert): mathematics can be consistently formalized
Realism (Cantor, Finsler): Consistency implies existence (C -> E)
consistent theories determine possible worlds

Russell, Brouwer, Hilbert: limited logic to inductive reasoning
Hilbert believed C->E, but restricted logic to inductive methods
Godel believed coinduction inconsistent, set theory paradoxes
proved incompleteness of inductive logics for mathematics
but could have proved completeness for coinductive logics

Godel overturned Hilbert’s program of inductive formalism
Hilbert’s program was inadequate due to inductive restrictions
Hilbert’s program might be possible for coinductive formalism

Three principles of metamathematics:
(1) C->E (2) inductive formalism (3) objective mathematics

Claim: all three principles cannot be simultaneously true
Hilbert: accepted C->E, inductive formalism
Godel: accepted objective mathematics, inductive formalism
rejected C -> E, puzzling aspect of Godel’s philosophy
Cantor, realism: accept C->E, objective math
reject inductive formalism, accept coinductive formalism

Significance for OOP

Unifying framework for OOP, software engineering, and AI
provides a better basis than TMs for OOP, OMT, UML etc
answers “What is OOP?” in a satisfactory way

Interactive models unify subdisciplines of CS
also provides framework for unifying CS, mathematics, physics
new answers to questions of nondeterminism, quantum theory
empirical CS, the Turing test, Church’s thesis

What is the practical impact of interactive models on OOP?
framework for development of new foundations for OOP
will change the way textbooks are written
the way we think about problems and problem solving
will unify, simplify models of OOP, SE, AI, HCI

Impact on practice is explored in:
Interactive Software Technology, Handbook of CS&E, 1996
available at www.cs.brown.edu/people/pw
application to:
design patterns, interoperability, coordination
AI agents, control theory, virtual reality, databases

These ideas have many alternative formulations
interactive technology -> new paradigms, new models, for CS