Taming the Power of Nondeterminism

Marjan Sirjani

Abstract

Modern computing systems operate in environments characterized by concurrency, interaction, uncertainty, and time constraints. Traditional deterministic abstractions are increasingly inadequate for modeling such systems, particularly in the presence of distributed execution, cyber-physical dynamics, and human–AI interaction. In this paper, we argue that nondeterminism—combined with time—constitutes a practically sufficient foundation for modeling contemporary computing systems. We propose that all relevant sources of uncertainty, including concurrency, environmental interaction, and human or AI behavior, can be uniformly captured as nondeterministic evolution over time. Rather than treating nondeterminism as an artifact to be eliminated, we establish it as a central organizing principle for system semantics. We further argue that engineering robust systems is not about removing nondeterminism, but about structuring, constraining, and reasoning about it.

1 Introduction

Modern computing systems are no longer isolated, sequential artifacts. They are distributed, interactive, and deeply embedded in dynamic environments. From cloud platforms and autonomous systems to human–AI collaborative workflows, system behavior emerges from multiple interacting components operating concurrently under partial information and timing constraints.

Despite this shift, much of software engineering and formal reasoning remain rooted in deterministic abstractions. Nondeterminism is often treated as an undesirable artifact—introduced temporarily during specification and later refined away during implementation. This view is increasingly untenable.

In this paper, we argue that nondeterminism is not incidental, but fundamental. More precisely, we propose the following thesis:

> All relevant sources of uncertainty in contemporary computing systems, concurrency, environmental interaction, and human/AI behavior, can be uniformly captured as nondeterministic evolution over time.

From this standpoint, nondeterminism is not a weakness of models, but their strength: it provides a unifying abstraction for representing incomplete knowledge, interaction, and variability. Engineering, then, becomes the task of taming nondeterminism through structure and constraint to build a predictable behavior.

2 Sources of Nondeterminism in Modern Systems

Nondeterminism arises naturally in multiple dimensions of modern computing. While these sources are often treated separately, they share a common semantic core: the presence of unresolved or externally determined choices.

2.1 Concurrency and Interaction

Concurrency can be modeled by nondeterminism through interleaving. When multiple processes execute simultaneously, the order of operations is not fixed. Even in the absence of explicit randomness, the scheduling of threads, message delivery, and synchronization events may produce multiple possible executions.

Interaction further amplifies this effect. Components communicate asynchronously, often without global coordination. The resulting behavior depends on timing, ordering, and availability of information.

The classical abstraction of concurrency as nondeterministic choice remains sufficient. The interleaving semantics traditionally used to model concurrent execution captures its observable effects without requiring additional primitive constructs. This observation extends beyond classical concurrency: even in modern systems involving timing constraints, environmental interaction, and human or AI components, behavior can still be understood as nondeterministic evolution over time.

2.2 Environment and Partial Information

Modern systems operate in open environments. Inputs arrive from sensors, networks, or users, and cannot be fully predicted or controlled. This introduces nondeterminism as a representation of uncertainty about future inputs.

Partial observability compounds the issue: systems must act without complete knowledge of the global state. Nondeterminism naturally captures this lack of information by representing multiple possible worlds consistent with current observations.

2.3 Human and AI Behavior

Human users and AI components are increasingly integral to system behavior. Both exhibit variability that is difficult to model deterministically.

Rather than attempting to model internal cognition or probabilistic mechanisms in full detail, it is often both sound and practical to abstract such agents as nondeterministic components. Their outputs can be treated as choices within a set of possible actions, constrained by context but not fixed in advance.

3 Time as a First-Class Semantic Dimension

Nondeterminism captures the space of possible behaviors, while time provides the structure along which these behaviors unfold. Modern computing systems are not only characterized by uncertainty, but also by temporal organization. Actions do not merely occur—they occur in time, under constraints, delays, and ordering relationships that shape system behavior.

Timing influences when events happen, how components interact, and whether system-level objectives are satisfied. Deadlines, timeouts, scheduling policies, and communication delays all introduce variability that cannot be ignored or abstracted away without loss of fidelity. In distributed and real-time systems, correctness often depends not only on what happens, but on when it happens.

From this perspective, system behavior can be understood as a set of possible executions evolving over time. Nondeterminism defines the alternatives at each step, while time orders these choices into coherent trajectories. This view naturally accommodates asynchronous communication, variable delays, and partial synchronization.

Importantly, time does not reduce nondeterminism; it structures it. By treating time as a first-class semantic dimension alongside nondeterminism, we obtain a unified framework capable of expressing both logical variability and temporal constraints. We may even consider introducing time as one of the ways to tame nondeterminism.

4 Nondeterminism as a Unifying Abstraction

One of our key claims is that diverse forms of uncertainty can be reduced to nondeterminism at the appropriate level of abstraction:

- concurrent sequential processes $\rightarrow$ nondeterministic interleavings of sequences of operations,
- environmental inputs and human/AI actions $\rightarrow$ nondeterministic choices,
- timing variability $\rightarrow$ nondeterministic delays,

Even probabilistic behavior can be viewed as a refinement of nondeterminism, where choices are assigned distributions rather than treated as unconstrained. Similarly, adversarial models interpret nondeterminism as worst-case choice.

This unification enables the use of common formal tools, such as temporal logic, model checking, and game-theoretic reasoning, across a wide range of systems.

Formal Reasoning Under Nondeterminism. Formal methods are explicitly designed to reason about nondeterministic systems. Properties are evaluated over sets of possible executions rather than over a single trace.

Typical examples include:

- *Safety*: something bad never happens.
- *Liveness*: something good eventually happens.
- *Refinement*: reducing nondeterminism through design while preserving correctness.

Nondeterminism is thus not a source of imprecision, but a structured representation of uncertainty that supports rigorous analysis.

5 Taming Nondeterminism by Refinement

If nondeterminism is fundamental, the goal of engineering shifts from elimination to structure and control. Specifications define a space of admissible behaviors; implementations are obtained by progressively constraining that space.

Conjecture: For every behaviorally specified system S, there exists a timed nondeterministic model M_0 such that every implementation I of S is a refinement of M_0:

$$I \sqsubseteq M_0.$$

Moreover, design can be understood as constructing a chain

$$I \sqsubseteq \cdots \sqsubseteq M_1 \sqsubseteq M_0$$

that progressively reduces nondeterminism while preserving correctness.

This refinement view gives engineering a precise role: not to remove nondeterminism altogether, but to shape it. Scheduling policies, contracts, control laws, and interface constraints all act by reducing admissible choices while maintaining required properties.

6 Conclusion

We have argued that nondeterminism, combined with time, provides a practically sufficient foundation for modeling modern computing systems. By treating nondeterminism as a first-class concept, we obtain a unifying framework that captures concurrency, interaction, environmental uncertainty, and human/AI behavior.

The established abstraction of concurrency as nondeterministic choice therefore remains sufficient even in the broader setting of modern systems. What changes is not the primitive itself, but the scope of what it is asked to model. Time clarifies and structures nondeterministic possibility into observable execution, while refinement provides the mechanism by which such possibility is tamed in engineering practice.

Future work includes sharpening the equivalence notions underlying behavioral completeness, relating this view to probabilistic and hybrid models, and developing tool support for integrated nondeterministic and temporal modeling.