Monotonic Continuation: Intelligence Across Unequal Timescales

Flyxion
Independent Researcher

August 2026

I. The Mistake of Equating Intelligence with Speed

There is an old observation from interface design worth recovering here: a system did not need to be fast everywhere to feel fast. It needed only to be fast at the point where a human was waiting on it [46, 47]. Everything behind that point could take as long as it needed to take, provided the interface did not make the human wait for it.

Artificial systems inherit this constraint, but they also invert it in a way the original observation did not anticipate. Latency is not one thing. An interaction has a latency — how long before something responds at all. A deliberation has a latency — how long before a considered judgment is reached. A resolution has a latency — how long before a question is actually settled, as opposed to merely answered. These three need not move together, and the assumption that they should is where the trouble starts.

Low interaction latency is close to an unqualified good; a system that leaves a person waiting for acknowledgment is failing at something basic. But low resolution latency is not obviously good at all. A question resolved instantly is sometimes a question resolved correctly and quickly. It is just as often a question that has been forced to closure before the computation its answer required had time to occur. An instantaneous answer to a question that needed deliberation is not evidence of intelligence. It is evidence that something upstream refused to let deliberation latency and interaction latency come apart.

This is the mistake the title names. Speed is not a single axis a system can be more or less good at. A system can be immediately responsive and, at the very same moment, still working on something difficult — provided its architecture allows the two to run at different rates rather than forcing every question through the same clock.

II. One Intelligence, Several Clocks

If interaction latency and resolution latency can legitimately differ, the architectural question is what makes that difference possible. The answer is that the different latencies do not have to be produced by a single synchronous process asked to do everything at once. They can instead be produced by several processes, running on several clocks, that remain parts of one continuing intelligence rather than an interaction layer bolted onto a separate reasoning layer.

Call the fastest of these reflex: whatever handles the immediate feedback loop, the part with a real human-scale latency constraint attached to it. Call a slower one deliberation: contextual integration, weighing, the ordinary work of forming a considered response — a distinction with an obvious cousin in the dual-process literature on fast and slow cognition [43], though the architecture developed here adds a third, slower tier that dual-process accounts do not require. And call the slowest consolidation, or resolution: whatever a question needs when minutes, hours, or days of computation are genuinely justified by its difficulty.

$$\text{reflex} \longleftrightarrow \text{deliberation} \longleftrightarrow \text{consolidation}$$

The double arrows matter as much as the three terms. This is not a pipeline where reflex hands off to deliberation and deliberation hands off to consolidation in one direction only. Each can inform the others; consolidation's slow output can reshape what deliberation does next, and deliberation's immediate needs can determine what consolidation is asked to work on. What makes this one intelligence rather than three cooperating systems is exactly that none of the three is required to wait on the others' clock, and none of the three is what the whole system reduces to.

The engineering consequence follows directly, and it is the same consequence the interface-design observation reached decades earlier in a narrower setting: keep the largest, slowest intelligence out of the latency-critical path. Not because it is less important — it may be doing the most difficult work in the entire system — but because forcing it onto the reflex clock is what produces Section I's mistake. A system built this way is not choosing between fast and slow. It has stopped needing to choose, because fast and slow are now different components rather than different settings of the same one.

III. Intelligence That Is Allowed Not to Know Yet

Once intelligence is understood as several processes on several clocks, a question that cannot yet be resolved does not have to be treated as a failure of the system asked to resolve it. It can instead remain an active object — something the slower processes continue working on while the faster ones proceed without waiting for it.

This forces a distinction that a single-clock system has no need for. Under one clock, there are really only two states available to any question: answered, or not yet gotten to, which functions the same as failed if an answer was expected now. Under several clocks, a third state becomes not just available but necessary:

$$\text{known} \qquad \text{unresolved but reachable} \qquad \text{presently unreachable}$$

A question is known when some process has already settled it and that settlement is available to whatever needs it. A question is presently unreachable when no process, at any horizon this system runs on, currently has what it needs to settle it — the prerequisites themselves do not yet exist. Between these sits the state this essay is actually interested in: unresolved but reachable, meaning some slower process is still working on it, or could be set to, and the question has not been abandoned even though it has not been answered.

Fant's treatment of asynchronous logic provides a useful hardware-level precedent for this distinction, at a much smaller scale [1]. In an ordinary Boolean representation, a signal line represents a value, but the representation does not necessarily say whether that value is settled. During propagation, the system can therefore require information about its own state of resolution that the value representation itself does not contain. External timing supplies one solution: wait long enough that the value may safely be treated as valid. Fant's alternative is to encode the distinction internally, marking signals as NULL or DATA so that incompleteness becomes representable rather than something an external clock must infer.

The importance here is not the particular encoding. It is the decision to make *not yet* a state of the system.

A cognitive architecture encounters an analogous problem at a much larger scale. If every question must be represented only as answered or unanswered, two importantly different conditions collapse into one: unresolved but still under effective computation, and not presently reachable by the system at all. The distinction this section has proposed therefore requires at least three epistemic conditions,

$$K, \qquad U_R, \qquad U_P$$

where K denotes presently known, U_R unresolved but reachable, and U_P presently unreachable, with movement possible between them as resources, evidence, representations, or computational time become available:

$$U_P \to U_R \to K$$

The significant property of U_R is that it preserves unfinished computation as an explicit object. A system need not manufacture an answer merely because its immediate interaction cycle has ended. It can retain the question, associate it with the state already accumulated around it, and permit a process operating on another timescale to continue its resolution.

Fant's NULL/DATA distinction is much simpler than this three-state epistemic scheme, and the analogy should not be pushed beyond this point. NULL does not distinguish between something that will shortly resolve and something that cannot presently be resolved at all. But it demonstrates the more basic architectural principle: if incompleteness matters to correct continuation, incompleteness must be representable. Otherwise some external mechanism must guess when incompleteness has ended.

What this state requires, though, is something the three-way distinction alone does not supply: some way of saying whether a process working across time on an unresolved question is actually making progress toward reaching it, as opposed to merely occupying it indefinitely, or worse, drifting away from it while appearing to remain engaged. A question can be held open honestly or held open as a kind of stalling; the difference has to be stateable, and stating it requires a criterion for whether successive states of a process are still moving toward what the earlier state left reachable. That criterion is what the next section supplies.

IV. Monotonic Continuation

Call the sequence of states a cognitive process passes through $S_0, S_1, S_2, \ldots$. The central claim of this essay is that intelligence under incomplete resolution is best understood not as a series of answers but as a relation between successive states:

$$S_0 \preceq S_1 \preceq S_2 \preceq \cdots$$

It matters what $\preceq$ does not mean. It does not mean that later states contain more information than earlier ones, or that S_{t+1} is simply S_t plus additions. That reading would make forgetting, compression, and correction look like violations of the relation rather than ordinary parts of it, and any system that consolidates, abstracts, or discards state would fail to qualify as continuing at all. A process that moves an enormous working context into a compact durable summary loses most of its bytes on purpose. If monotonicity required byte preservation, that act of consolidation would count as a break in the process rather than one of its most important operations.

$\preceq$ should instead be read as continuational adequacy. S_{t+1} succeeds S_t when it preserves the distinctions from S_t that remain necessary for the trajectory still reachable from it. Write the reachable set of a state as $\mathcal{R}(S_t)$, the set of continuations that state can still admit. Then the relation is closer to

$$\mathcal{R}(S_t) \supseteq_* \mathcal{R}(S_{t+1})$$

where the starred containment means the later state retains the admissible or significant continuations of the earlier one, not literally every continuation the earlier state could in principle have produced.

Two clarifications about this notation are worth making explicit before proceeding, because the sequence $S_0, S_1, S_2, \ldots$ can superficially invite a reading it is not meant to carry. The first is that monotonic continuation should not be confused with sequential computation. Nothing in this essay claims that the underlying intelligence halts, stabilizes all of its components, and moves atomically from one global state to the next — an assumption that distributed-systems theory has its own name for and its own reasons to reject [3]. The processes this essay has been describing — reflex, deliberation, consolidation, human steering, memory migration, repair — may all be running concurrently and asynchronously, without any privileged instant at which the whole system is simultaneously at rest:

$$P_1(t) \parallel P_2(t) \parallel \cdots \parallel P_n(t)$$

Each S_t should be understood as an analytical cut through this ongoing, possibly overlapping activity — a distinguishable configuration used to state what continuational adequacy requires, in much the same spirit as a consistent global snapshot of an otherwise asynchronous distributed computation [4] — not a claim that the process itself consists fundamentally of sequential global updates. $S_t \preceq S_{t+1}$ is a statement about what survives between two such cuts, not about the execution order of whatever machinery produced them.

monotonicity $\neq$ sequentiality

The S_t notation orders states by continuational adequacy. It does not assert that the underlying machine moves through globally stable states one at a time. Reflex, deliberation, consolidation, and repair can run concurrently and asynchronously; what must remain ordered is not their execution, but the preservation of the reachable future across whatever cuts an observer or the architecture itself chooses to take.

The second clarification follows from the first: $\preceq$ does not require that there be one privileged next state. From a given S_t, several continuations may be equally admissible,

$$S_t \to S_{t+1}^{(1)}, \qquad S_t \to S_{t+1}^{(2)}, \qquad S_t \to S_{t+1}^{(3)}$$

and what distinguishes an admissible continuation from an inadmissible one is not its position in some predetermined sequence but whether it preserves the relevant dependencies and reachable futures:

$$S_t \preceq S_{t+1}^{(i)} \quad \Longleftrightarrow \quad \text{the relevant dependencies and reachable futures survive}$$

A dependency network with independent branches typically admits many correct orderings and many incorrect ones; correctness tracks preserved dependency, not any single sanctioned sequence. The same is true here. Monotonic continuation constrains which transitions are admissible. It does not by itself select a unique one.

A great deal of S_t can be thrown away as long as nothing in $\mathcal{R}(S_{t+1})$ has been foreclosed that mattered. This is what makes the memory hierarchy in Section VIII a case of the same relation rather than a separate mechanism, and it is why HYDRA's repair machinery belongs later in the essay: repair is what restores continuational adequacy when a transition has instead narrowed the reachable set it shouldn't have.

Two failure modes now become distinguishable in the same vocabulary. A system can fail by refusing to move: it holds S_t static, declines to commit to any S_{t+1}, and calls that caution. And a system can fail by moving without the relation holding at all: it produces some S_{t+1} that discards a distinction the trajectory needed, which is not continuation but collapse dressed as progress. Monotonic continuation is the narrow path between the two: never static, never lossy of what matters, one adequate transition at a time.

It is worth being explicit that this section, and not the Expiatory Gap, is where the essay's real claim lives. The Expiatory Gap names a mismatch: a generating process can produce distinctions at a density or rate that a receiving one cannot presently contain. That is a statement about capacity. Monotonic continuation names a strategy: what a process does, generating or receiving, when adequate resolution is not presently available. The two are related but neither entails the other. A process can continue monotonically for reasons that have nothing to do with any capacity mismatch, simply because evidence is still arriving, or computation is still running, or the environment has not finished changing. And a process facing a genuine Expiatory Gap is not thereby guaranteed to continue monotonically; it can just as easily resolve prematurely, truncate, or distort, which is exactly the failure mode Section XIV returns to. The Expiatory Gap describes a problem of transmissibility. Monotonic continuation describes a strategy under temporal incompleteness. The rest of this essay is an argument that the second concept, not the first, is the one that organizes explanation, steering, memory, translation, and repair.

V. Working Toward Conclusions Monotonically

Explanation is the most familiar case of monotonic continuation, which is exactly why it is treated here as a special case rather than as the essay's subject. Given a conclusion C that cannot yet be stated responsibly, one option is to state something smaller and false, and correct it later:

$$\text{false simplification} \rightarrow \text{correction} \rightarrow \text{correction} \rightarrow C$$

This is the ordinary shape of bad pedagogy, and it violates continuational adequacy directly: each correction has to first undo a distinction the previous state asserted before it can add a new one, so $\mathcal{R}(S_t)$ is not preserved across the transition, it is damaged and then repaired. The alternative is a sequence of successive refinement,

$$A_1 \subseteq A_2 \subseteq A_3 \subseteq \cdots \subseteq A_n$$

where each A_i remains true within the scope it claimed and later terms differentiate rather than overturn it. The operative instruction is not "say the most accurate thing" and not "say the simplest thing." It is: say the largest thing that can presently be said without destroying the path to the larger thing. What is sayable and what is true are not the same criterion, and monotonic explanation is the discipline of only ever asserting the intersection of the two.

This is where the role of a mediating process becomes interesting, and where the essay's later discussion of multi-model architectures (Section XI) gets its motivating case. Suppose one process has advanced far along its own trajectory, having reached some S_{1000}, while the process it communicates with is still at S_{37}. The naive response is compression: reduce S_{1000} to a size S_{37} can absorb. But this mistakes the problem. S_{1000} was reached by a long sequence of admissible transitions, most of whose prerequisites do not exist yet at S_{37}; compressing it produces something small but not something reachable. The actual problem is local: determine $S_{37} \rightarrow S_{38}$. A process that can do this is not summarizing, it is managing the gap between what is known, what can presently be explained, and what can presently be assimilated, which are three different quantities:

$$\text{what is known} \neq \text{what can presently be explained} \neq \text{what a recipient can presently assimilate}$$

The gap manager's task is not to shrink the first quantity into the third. It is to find the next transition that is simultaneously true to the first, expressible as the second, and admissible given the third. That transition is usually much smaller than either the sender's full state or the distance to the sender's conclusion, and its smallness is not a limitation on the explanation. It is the explanation, correctly scaled to what a single step of continuation can carry.

VI. Steering Instead of Complete Specification

A related but distinct case arises whenever a process does not need explanation delivered to it so much as direction supplied to it mid-trajectory. Represent this as a perturbed con-

tinuation,

$$S_{t+1} = F(S_t, u_t)$$

where u_t is a small intervention and F is whatever admissible transition the underlying process performs on its own. The interesting question is not how much information u_t carries in isolation, since in isolation it usually carries very little. A two-word correction means almost nothing detached from context. What matters is how much it determines given S_t:

$$I_{\text{eff}}(u_t; S_{t+1} \mid S_t)$$

This relation applies to any process supplying u_t, but it applies nowhere more vividly than when that process is a human, whose accumulated judgment can never be fully specified upfront and whose steering therefore leans almost entirely on this conditional leverage rather than on the content of any single instruction — a case Section XIII returns to directly. This is a claim about conditional, not additive, significance. It would be a mistake to write $I_{\text{eff}} = I(u_t) + I(S_t)$, as if the steering message and the accumulated trajectory were separate deposits into some shared account. The steering message does not add its own information to the trajectory's information. It selects among the continuations the trajectory has already made available, and its leverage is entirely a function of how much that trajectory has narrowed the space of admissible next states. A large, richly developed S_t makes a tiny u_t decisive. A thin or absent S_t makes the same u_t nearly meaningless, because there is no accumulated structure for it to select within.

This has a direct instance in coding-agent interaction. An agent working on a task accumulates a trajectory before any steering occurs: files inspected, hypotheses formed, edits attempted, tests run and failed, a local representation of what the task actually requires. A steering message like "keep the existing abstraction, move this into the trait instead" enters that trajectory already in progress. Detached from it, the sentence is not a specification of anything; it presupposes an abstraction, a trait, and a "this" that only the trajectory has made determinate. Within the trajectory, the same sentence can redirect the entire remaining continuation. The sentence has not grown more informative. The state it is conditioning on has grown large enough that a small perturbation of it has large consequences.

This produces an inversion worth naming explicitly, because Section V's gap-manager case and this one are mirror images of the same relation. In the gap-manager case, the process with the more developed internal state is the one being explained — a slow system running far ahead, disclosing itself monotonically to a faster one that cannot yet hold the whole of it. In the steering case, the process with the more developed state is the one doing the explaining: a human's accumulated intention, built from years of unstated design commitments, entering a fast executor's trajectory as a sequence of small corrections rather than a complete upfront specification. Neither party could produce a complete transmission even if they tried.

There is, however, a further stage this relation reaches once steering is not a single event but a repeated one. Suppose interventions belonging to some category c — file edits of a certain kind, shell commands of a certain kind, any class of change a system might make — are supplied and accepted repeatedly across a trajectory:

$$u_t^{(c)}, u_{t+1}^{(c)}, \ldots, u_{t+n}^{(c)}$$

Each acceptance is evidence about a specific claim: that the transition $F(S_t, u_t^{(c)})$ is one the trajectory's owner endorses, not just once but as a standing preference. Past a certain point, repetition of this kind stops being usefully described as steering at all. What has actually happened is that the preference has migrated out of the intervention and into the transition rule itself:

$$F(S_t, u_t^{(c)}) \quad \longrightarrow \quad F^{(c)}(S_t)$$

Call this steering absorption. It is the third stage of this section's argument, after the basic relation $S_{t+1} = F(S_t, u_t)$ and the conditional-information account of why a small u_t can matter enormously. Absorption is what happens when a category of intervention has been supplied often enough, and with enough consistency, that the system can perform the corresponding transition on its own recognizance, without waiting for $u_t^{(c)}$ to be resupplied.

The confirmation behavior of a typical coding-agent CLI is an unusually clean instance of exactly this process, and it is worth treating as more than an implementation anecdote. Such tools commonly ask for confirmation before making a change, but they do so at the granularity of change *categories*, and often offer, per repository, the option to stop asking for a given category once it has been approved enough times. This is not best described as a single trust dial being turned up. What is actually on offer is a category-indexed structure of authorization,

$$\tau : C \to L$$

where C is the set of transition categories — filesystem edits, command execution, network operations, destructive operations, and so on — and L is the space of authorization conditions attached to each, itself something closer to a partial order than a binary allow/forbid switch. Trust, on this reading, is not a scalar $\tau \in [0, 1]$ that rises uniformly as a system proves itself. It is structured permission, acquired independently along each category's own trajectory, because each category has its own history of approvals and its own point at which that history becomes sufficient.

This gives Section VII, which follows, its bridge. A trajectory does not merely make future instructions easier to interpret, the way a long conversation makes a short correction legible. Past a threshold, a trajectory changes what must be explicitly said at all:

$$\text{trajectory} \;\to\; \text{repeated steering} \;\to\; \text{stable regularity} \;\to\; \text{absorption into } F$$

Absorption is not, however, unconditionally legitimate simply because approval has repeated. Its legitimacy is exactly the question Section IV's relation was built to ask: does removing the human intervention preserve the reachable continuations the intervention was protecting? In rough form, absorption of category c is warranted when

$$\mathcal{R}_{\text{admissible}}\Big(F^{(c)}(S_t)\Big) \approx \mathcal{R}_{\text{admissible}}\Big(F(S_t, u_t^{(c)})\Big)$$

that is, when the autonomous transition and the previously-supervised one lead to approximately the same reachable future. This is a considerably stronger claim than "the user clicked approve several times." Repeated approval is evidence for the equivalence, not identical to it, and the evidence is only ever evidence about the distribution of category-c interventions actually observed. A category can accumulate a long, consistent history of low-consequence instances and then encounter a rare, high-consequence instance the prior approvals never tested — a destructive filesystem operation that happens to touch something none of the earlier approved instances touched, for example. Absorption grants autonomy on the strength of a pattern; it does not certify every instance the pattern could someday produce.

This is precisely why Section XIII's account of the human's place in the architecture does not shrink to nothing as absorption proceeds. A mature human-machine system is not one where every category of intervention eventually gets absorbed and the human's role approaches zero. It is one that absorbs the categories where $\mathcal{R}_{\text{admissible}}$ has genuinely stabilized, while deliberately withholding absorption from categories whose consequences remain long-horizon, ambiguous, novel, or difficult to restore — the categories, in other words, where a single untested instance could foreclose something no repair budget downstream could recover. Steering absorption is consequently not a story about steering disappearing as trust accumulates. It is a story about what a trajectory's history is actually for: not just making the next instruction easier to give, but determining which instructions no longer need to be given at all.

VII. Why the Trajectory Matters

If continuational adequacy is the relation that governs each transition, the trajectory as a whole — the full sequence $S_0, S_1, \ldots, S_t$ — is what makes any single transition cheap. Persistent history substitutes for specification. A steering message that would be underdetermined as a standalone instruction becomes sufficient once conditioned on everything the trajectory has already fixed. This is the same claim as Section VI's conditional information, restated at the level of the whole process rather than a single intervention: a system with a long, coherent trajectory does not need large inputs, because most of what a large input would specify is already implicit in S_t.

This changes what should count as evidence of intelligence. Judged transition by transition, a system's value looks like the correctness of each individual output. Judged as a trajectory, a system's value looks more like whether it remains steerable — whether its accumulated state continues to admit correction without discarding what correction depends on. A system that answers every query correctly on the first attempt but cannot incorporate a subsequent "no, not that" without restarting from scratch has a narrower $\mathcal{R}(S_t)$ than its apparent competence suggests; each of its correct answers is a local success purchased at the cost of the trajectory's continuability. A system that is occasionally wrong but preserves enough state that "not quite — this way" reliably propagates forward is, on the criterion this essay has been building, doing something a purely correct-on-first-attempt system is not.

This is also where the difference between an error and a foreclosure becomes precise

rather than merely intuitive. An error is a transition $S_t \to S_{t+1}$ where S_{t+1} turns out to be a poor local step, but $\mathcal{R}(S_{t+1})$ still contains the continuations that matter — the trajectory can still reach where it needed to, at the cost of an extra correcting transition. A foreclosure is a transition where S_{t+1} has narrowed $\mathcal{R}(S_{t+1})$ so that some needed continuation is no longer reachable from it at all; no subsequent steering can recover it, because there is nothing left in the state for a steering message to select. The first kind of failure is the ordinary cost of operating under incomplete resolution. The second is a break in continuation itself, and it is the failure this essay is actually concerned with — not wrongness, but the premature closing off of what a later, better-informed transition would have needed.

VIII. From Context Windows to Cognitive Memory Hierarchies

Section IV's account of $\preceq$ already implied that continuation does not require preserving everything, only what remains necessary to the reachable trajectory. This section makes that claim architectural. Treat cognitive state the way a computer treats working memory: not as one undifferentiated pool but as a hierarchy of temperatures, the same locality that justifies caching and paging in ordinary computer systems [9, 8],

$$\text{VRAM} \leftrightarrow \text{RAM} \leftrightarrow \text{NVMe} \leftrightarrow \text{durable semantic memory}$$

Fast, small, immediately available state sits closest to active computation. Colder, larger, less immediately needed state migrates outward. What makes this a hierarchy rather than a mere storage tiering is that movement between levels is itself a transition subject to $\preceq$: consolidating an enormous working context into a compact durable representation is a case where $|S_{t+1}| \ll |S_t|$, and it counts as monotonic exactly when $\mathcal{R}(S_t) \supseteq_* \mathcal{R}(S_{t+1})$ still holds — when the compression has kept the continuations that mattered and discarded only the ones that did not.

This reframes what a context window is for. A context window is not the site where a system's intelligence lives; it is the fast tier of a memory hierarchy whose slower tiers are doing most of the actual work of remaining coherent over time. A process that only ever operates within its context window, with nothing consolidating outward, is a process with no slow tier — every session starts near S_0 regardless of how many sessions preceded it, and the trajectory Section VII described as the source of a system's real capability never accumulates. Conversely, a process whose expensive conclusions do get consolidated outward, so that later working contexts can inherit them without recomputation, is one where the reachable set at $t{+}1$ can exceed what the raw context window at $t{+}1$ would suggest — the missing bytes are recoverable in effect, because what they made reachable was carried forward by a different route.

The natural next question is whether this migration can cross not just memory tiers within one process but the boundary between different processes entirely — whether something consolidated by one system can become part of another system's fast tier without being recomputed from scratch. That is the question Section IX takes up directly.

IX. Are KV Caches Thoughts?

Section VIII ended by asking whether consolidated state can cross process boundaries, not just memory tiers within one process. The most literal version of this question is whether a KV cache — the accumulated internal state a transformer builds up while processing a sequence [13] — can be handed from one model to another the way a consolidated memory is handed from RAM to disk. The honest answer is generally no, and it is worth being precise about why, because the reason is more informative than the negative result itself.

A KV cache is not a neutral record of "what was attended to." It is a representation stated entirely in the geometry of one particular model — its own learned basis, its own layer structure, its own way of carving a sequence into features. Two models trained separately, even on similar data, do not share this geometry, so a tensor meaningful inside one is close to noise inside the other. The failure is not that the receiving model is too small to hold the cache. It is that the cache is not stated in a vocabulary the receiving model has any way to read. This is the same failure Section IV ruled out when it insisted that $\preceq$ could not mean literal content preservation: transferring raw state across an incompatible representation preserves bytes while destroying $\mathcal{R}$, which is exactly backwards.

What survives this failure is a more interesting possibility: training models, from the start, around a shared latent interface rather than attempting to reconcile their internal geometries after the fact. Let each model i have its own hidden representation space H_i, and posit an encoder and decoder pair,

$$E_i : H_i \to Z, \qquad D_i : Z \to H_i$$

mapping into and out of a common space Z. The goal of such an interface is not that $E_i(h)$ recovers h exactly, nor that two different models' encodings of "the same thought" be numerically identical. Numerical identity was never the right criterion; Section IV already established that adequate continuation does not require preserving everything, only what remains necessary to the reachable trajectory. The right criterion for Z is that it preserve whatever distinctions are operationally significant — the ones a downstream continuation actually depends on — while being free to discard everything else, including the model-specific texture that made direct KV transfer fail in the first place.

This reframes the original question. "Are KV caches thoughts?" is the wrong form of the question, because it treats a thought as identical to one particular tensor. The right question, and the one the rest of this essay has been assembling the vocabulary to ask, is what a thought would have to be such that it could survive being moved between representations that do not share a geometry. Section X states that condition directly.

X. Persistence Under Translation

Given two models' representation spaces H_i and H_j, define a translation

$$T_{ij} : H_i \to H_j$$

Section IX already ruled out requiring T_{ij} to be an exact structural isomorphism; no such map generally exists between differently-trained models, and demanding one would

just relocate the KV-transfer failure one level up. What can be asked instead is whether continuation approximately commutes with translation — whether applying model i's update and then translating gives approximately the same result as translating first and then applying model j's corresponding update:

$$T_{ij}(F_i(h)) \approx F_j(T_{ij}(h))$$

This is not primarily a statement about model interoperability, although it has that consequence. It is an operational test for what it would mean for a state to persist under translation at all. A state $h \in H_i$ passes the test at a given moment when the two paths through the diagram land close together; it is at that moment translatable without loss of what matters. When the two paths diverge, translation has failed not because the target representation was too small, but because whatever h was tracking does not survive being re-expressed in H_j's vocabulary — the same failure mode as an inadequate transition under $\preceq$, now located at the boundary between two different processes rather than between two moments of one process.

This gives the essay's account of a "thought" its final and most general form. A thought is not a tensor, a cache, or any other fixed artifact of one representation. It is whatever structure in a state's continuations survives being carried across the transformations Sections VIII, IX, and X have each considered in turn: consolidation across memory tiers, encoding into a shared interface, and translation across model boundaries. All three are instances of the same relation this essay opened with. What differs between them is only which kind of transformation the state is being asked to survive — moving in time, moving into a shared code, or moving across a boundary between processes that were never the same process to begin with.

XI. Evans's Three-Model Machine as Sustained Continuation

It is worth returning, now that the vocabulary is in place, to the architecture that motivated this essay in the first place. A micro model, a medium model, and a huge model, arranged so the fast model handles interaction, the medium model handles consolidation, and the huge model handles whatever computation minutes, hours, or days can justify — this can be described, and usually is described, as a performance optimization: keep the expensive model off the latency-critical path. That description is not wrong, but it is no longer the interesting one available. Reread with the machinery of Sections IV through X, the same architecture looks like an engineering sketch of sustained continuation between processes running on radically unequal clocks.

The huge model's trajectory can advance to some S_{1000} while the conversation the micro model is sustaining sits at S_{37}. Section V already established that the right response to this gap is not compression of the huge model's state down to a size S_{37} can hold; S_{1000} was reached by transitions whose prerequisites do not yet exist at S_{37}, so a compressed version of it would be small without being reachable. What is needed is the local transition $S_{37} \to S_{38}$, and the medium model's role is to find it — not to summarize what the huge model knows, but to manage the gap between what is known, what can presently be explained,

and what the conversation can presently assimilate. This is Section V's gap manager, now given a place in an actual architecture rather than treated as an abstract role.

The question of shared state, raised abstractly in Sections IX and X, also has its concrete form here. The micro model does not need to inherit everything the huge model's state contains; Section IX already argued that numerical identity of internal state was never the right criterion. What the micro model needs is access to the portion of the huge model's trajectory for which the current conversation has adequate prerequisites — the part of $\mathcal{R}(S_{1000})$ that is already reachable from S_{37}, or nearly so. A translation in the sense of Section X, from the huge model's representation to something the micro model's trajectory can use, succeeds exactly when it delivers that portion and nothing the conversation cannot yet hold.

What makes this worth calling sustained continuation rather than orchestration is that no component of the machine is required to resolve prematurely on the others' behalf. The huge model is not forced to compress; the medium model is not forced to summarize everything at once; the micro model is not forced to wait for a result it cannot yet use. Each process continues along its own trajectory at its own rate, and the architecture's entire function is to keep those trajectories mutually admissible — to ensure that what the fast process eventually receives from the slow one is always the next transition, never a premature or lossy compression standing in for one. Section VI's account of steering supplies the other half of this same machine: the conversation does not only receive from the huge model, it also steers it, sending small interventions that the huge model's own S_t makes decisive. The three-model architecture, seen this way, is not fast-AI-plus-medium-AI-plus-slow-AI. It is one continuing cognitive process, deliberately built to keep its own components from forcing each other into closure before they are ready.

XII. HYDRA: Distributed Repair Rather Than Model Hierarchy

Section XI described a machine of three processes on three clocks, each kept from forcing premature closure on the others. HYDRA generalizes this beyond any fixed number of processes and beyond the specific fast/medium/slow architecture Evans's machine happens to instantiate. The relevant object is not the count of processes but the fact that different processes carry different repair budgets — different amounts of correcting capacity, available over different horizons:

$$R_\mu(\Delta t_\mu) \subseteq R_m(\Delta t_m) \subseteq R_L(\Delta t_L)$$

A fast process operating under a short horizon Δt_μ has a correspondingly small repair budget: little time to notice an inadequate transition, less to correct it. A slower process has a larger budget, not because it is more capable in any general sense, but because its horizon is longer and more of a transition's consequences have had time to surface before it must respond. The nesting matters more than the inequality: whatever the fast process can repair, the medium process can also repair, and whatever the medium process can repair, the slow process can also repair, but not the reverse. Repair capacity accumulates with horizon.

This gives a precise sense in which failure at a fast scale is not global failure. Section VII already distinguished an error, which leaves $\mathcal{R}(S_{t+1})$ intact, from a foreclosure, which

does not. A fast process that commits an error — a transition that is locally poor but does not narrow the reachable set — has not thereby damaged anything a slower process downstream cannot address. The unresolved distinction the fast process failed to handle does not disappear; it becomes an object available to whichever process has a repair budget large enough to reach it. This is the architectural cash-out of Section III's insistence that "unresolved but reachable" is a distinct and valuable epistemic state, not a synonym for failure. A distinction only becomes truly lost when no process in the hierarchy has a repair budget capable of reaching it before it foreclosed — when even $R_L(\Delta t_L)$ is insufficient, which is the condition Section XIV will return to as the actual failure this essay is concerned with.

The consequence is that the object worth tracking is not any individual model, at any point in the hierarchy, but the continuing distributed state the hierarchy as a whole maintains. A single process viewed in isolation looks like it is constantly failing to resolve things — it hands off unresolved distinctions upward at every scale. Viewed as a whole, the hierarchy is doing exactly what Section IV's $\preceq$ requires: each local transition, even an imperfect one, preserves the reachable continuations by ensuring a process with adequate budget is available to repair what the immediate transition could not. The repair machinery is not a correction mechanism bolted onto continuation after the fact. It is what makes continuation across radically unequal timescales possible in the first place, since no single process, running at a single clock, could otherwise carry every distinction to the horizon it eventually needs.

XIII. The Human Inside the Architecture

Every process discussed so far has been a computational one, but nothing in Sections IV through XII actually required that. The relation $\preceq$, the repair budgets $R_\mu \subseteq R_m \subseteq R_L$, the gap-manager role — none of it depends on the participating processes being models rather than people. A person steering a system, correcting it, or waiting for it to catch up is not external to the architecture, supplying occasional input from outside, any more than the notebook is external to the mind that relies on it [36, 37]. They are another process running on their own clock, with their own accumulated trajectory and their own repair budget.

This is easiest to see by returning to Section VI's account of steering. A design correction like "keep the existing abstraction, move this into the trait" was already shown to carry almost no information in isolation and a great deal conditional on an accumulated S_t. What was left implicit there is that the S_t doing the conditioning is not only the agent's trajectory. It is also the human's — years of accumulated judgment about how software should be structured, most of it never stated because it has never needed to be. That accumulated judgment is itself a slow process, in exactly the sense Section VIII gave the word: a large, cold body of consolidated state that a small, fast intervention draws on without ever transmitting directly. The sentence is small. What it is conditioned on is not.

This means the system under description in this essay is never simply human-then-machine, a person issuing instructions to a process that executes them. It is a collection of processes at different temporal resolutions — some biological, some computational — each capable of altering what the others can still reach. The human's slow trajectory constrains what the agent's fast trajectory can admissibly become; the agent's fast trajectory, by

surfacing intermediate states the human could not have fully anticipated, constrains what the human's next intervention will be. Section VII's claim that a system should be judged by whether it remains steerable applies to this pairing symmetrically. A human whose accumulated judgment cannot be drawn on by small interventions — who can only specify completely or not at all — is exhibiting the same failure Section VII described in a machine: a trajectory that has stopped being able to condition anything.

This has a direct consequence for how the repair hierarchy of Section XII should be read. R_L, the largest repair budget in that nesting, is not guaranteed to belong to a computational process at all. Often it belongs to the person, who alone holds the horizon long enough to notice that a distinction foreclosed weeks earlier now matters, and who alone can supply the correction no faster process had the budget to make. The architecture is not complete without this tier, and treating the human as outside it, rather than as its slowest and highest-budget member, understates what the human's presence is actually doing.

None of this, however, should be mistaken for a claim that the human's presence is permanent by necessity, or that its disappearance always means the same thing. There are at least three distinct ways in which human participation in a process can cease to be required, and **these are not three stages in the progressive elimination of the human.** Treating them as points on a single continuum — as though every disappearance of the human were a further increment of the same achievement — would misdescribe what is actually happening in each case.

The first is steering absorption, already developed in Section VI. Here the disappearance is historical and learned. The trajectory contains a record of repeated human interventions in some category c, and past a threshold the system incorporates the regularity that history demonstrates:

$$F(S_t, u_t^{(c)}) \longrightarrow F^{(c)}(S_t)$$

The human supplied genuine, trajectory-dependent information at each step. What changed is that the system's own accumulated history eventually made further explicit supply unnecessary. This is a real achievement of continuation in this essay's sense — the system's S_t grew rich enough that $u_t^{(c)}$ became redundant with what S_t already determined.

The second is representational internalization, and it is architectural rather than learned. Here the human was never contributing trajectory-dependent judgment at all. They were compensating for a formalism that omitted information the system needed to coordinate correctly — Fant's engineering human, supplying the timing interval a Boolean circuit's own symbolic expression could not supply for itself [1]:

$$(R, x) + q_{\text{human}} \to y \qquad \Longrightarrow \qquad (R', x) \to y$$

Replacing the impoverished representation R with an adequate R' makes the human's apparent function disappear by construction, not by accumulation. There is no trajectory of learning to point to, because there was never a regularity to learn — only a gap in the representation that a better representation closes outright.

The third is emergent continuation, and it does not belong to either of the first two categories. It does not explain how a designed system comes to need its designer less. It challenges whether the relevant continuation must have been designed at all. Fant's picture

of a sufficiently rich population of spontaneously interacting symbols — his prebiotic case — has no engineer supplying a missing coordination variable and no history of steering being absorbed into a rule. There is only a population of interactions, some of which persist and some of which do not.

This third case is where this essay's own apparatus reaches its limit, and it is worth being exact about where that limit falls. Throughout this essay, $\mathcal{R}_{\text{admissible}}(S_t)$ and the nested repair budgets $R_\mu(\Delta t_\mu) \subseteq R_m(\Delta t_m) \subseteq R_L(\Delta t_L)$ have presupposed a criterion by which some transition counts as repair rather than merely as change — a criterion supplied, in every case this essay has considered, by an architecture, a task, a constraint system, or a continuing purpose that something or someone maintains. Fant's limiting case would replace that structure,

$$\text{admissibility} \rightarrow \text{persistence}$$

with something considerably more minimal:

$$\text{persistence} \rightarrow \text{the only available selection criterion}$$

These are not interchangeable. In the first, $\mathcal{R}_{\text{admissible}}$ is normative relative to something outside the bare fact of persistence — it says which continuations *ought* to be preserved, not merely which ones happen to occur. In the second, there may be nothing yet corresponding to "ought" at all. There are only processes that continue and processes that stop.

The question this leaves open is consequently not quite "where does $\preceq$ come from," which could suggest merely tracing the relation back to whoever specified it. The sharper question is:

$$\boxed{\text{How can persistence acquire normativity?}}$$

How does what merely continues ever become distinguishable from what ought to be preserved for a process to remain the kind of process it is? That question is large enough to warrant its own treatment, and this essay does not attempt one. What matters here is only that the boundary be stated rather than quietly crossed. Steering absorption and representational internalization both operate entirely within a system where admissibility is already given — by a task, a constraint, an architecture, a human's standing intentions. Extending this essay's vocabulary to Fant's third case would require assuming that same normativity is available at a point where, by construction, nothing yet guarantees it. Monotonic continuation, admissibility, and repair keep their precise meanings only by not being asked to also explain how a criterion of admissibility could arise where none was given — a question closer to abiogenesis and self-organization than to the designed, multi-timescale cognition this essay has been describing throughout.

XIV. Against Premature Resolution

Everything built across this essay converges on a single failure mode, and it is worth stating without further formalism. Interactive systems are under constant pressure to produce an output whenever queried. That pressure is not itself the problem; the problem is what it

does to a state that is not yet ready to resolve. Section III distinguished three epistemic states — known, unresolved-but-reachable, and presently-unreachable — and argued that the middle one is not a deficient version of the first. It is a state with its own integrity, one that Sections IV through XII have since given a precise mechanical description: a state whose $\mathcal{R}$ still contains what matters, waiting on a transition, or a repair budget, adequate to reach it.

Premature resolution destroys that state by answering the wrong question. Instead of asking what the next admissible transition is, a system under pressure to respond asks only what output would satisfy the interaction, and produces something that looks resolved without being reachable from anywhere true. This is not the same failure as an ordinary error. Section VII already distinguished the two: an error leaves $\mathcal{R}(S_{t+1})$ intact and can be corrected by a later transition; premature resolution forecloses it, converting a distinction that was unresolved-but-reachable into one that is now simply false, or simply gone. No repair budget, however large, can recover a distinction that a prior transition has actively erased rather than merely failed to advance. This is the condition Section XII flagged and deferred — the case where even $R_L(\Delta t_L)$ is insufficient — and it is now clear why that case is not primarily about the size of the repair budget. It is about whether there is anything left for that budget to act on.

Sustained intelligence, on the account this essay has been assembling, is not the capacity to resolve quickly. It is the capacity to carry an unresolved-but-reachable state forward without collapsing it into a false resolution — to prefer a correct "not yet" over an incorrect "yes." A slower system that reliably makes this choice is doing something a faster, more confidently wrong one is not, regardless of how the two compare on any single transition. The gap this essay opened with is answered not by closing it faster, but by never mistaking its temporary width for a reason to pretend it has already closed.

XV. Conclusion: Intelligence as Monotonic Continuation

The picture this essay opened with was the wrong shape for what it was trying to describe: intelligence as a mapping,

$$x \mapsto y$$

a fixed input producing a fixed output, evaluated once and forgotten. Nothing built across the preceding fourteen sections fits that shape. What fits instead is a temporally extended process,

$$S_0 \to S_1 \to \cdots \to S_t \to \cdots$$

distributed across components running on different clocks, holding state at different temperatures, and expressed in representations that do not share a common geometry. The essay's task was to say what makes such a process one continuing intelligence rather than a sequence of disconnected acts, and the answer it arrived at was a relation, not a mechanism: $S_t \preceq S_{t+1}$ whenever the later state preserves the continuations the earlier one still needed, whether or not it preserves the earlier state's content.

That relation required more than the opening sections anticipated, and it is worth tracing what it actually accumulated rather than restating it in the abstract. Preservation could not mean preservation of bytes, on pain of ruling out consolidation, translation, and repair as failures rather than as the ordinary business of continuing; it had to mean preservation of $\mathcal{R}$, the reachable future. Before that relation could even be stated, Section III had to establish that *not yet* is a state a system can hold explicitly rather than a gap it must silently guess at — a point sharpened by Fant's own architecture, where a value's settledness had to be represented internally before any circuit could resolve correctly without an external clock supplying the missing coordination. Section IV then gave the relation its name and its qualification: not containment, but continuational adequacy.

Steering, once introduced, turned out not to be one relation but three stacked ones — a perturbation against an accumulated trajectory, a conditional quantity explaining why a small perturbation against a rich enough S_t can matter enormously, and a further process, steering absorption, by which a repeated and well-evidenced perturbation ceases to need supplying at all. Trust, correspondingly, turned out not to be a single number rising with good behavior but a structure, indexed by category, each category accruing its own history and its own threshold for delegation. And when the question turned to why a human's participation might ever become unnecessary, the essay had to distinguish three mechanisms that are not stages of one progressive elimination: steering absorption, where a real regularity is learned from a real trajectory; representational internalization, where an apparently indispensable human turns out to have been compensating for a representation that was simply incomplete, and disappears by construction once that representation is repaired; and emergent continuation, where there is no architecture to internalize into, only a population of interactions, some of which persist. The essay followed the first two into the architecture it was describing and deliberately left the third outside it, because extending $\mathcal{R}_{\text{admissible}}$ and $\preceq$ to a case with no external criterion of admissibility at all would not extend this essay's theory — it would require a different one, answering a different question: not where $\preceq$ comes from within a designed system, but how persistence could ever acquire normativity where no design was given. That question remains open, and stays open on purpose.

What the essay did complete, within the boundary it kept, was an account of the multi-clock architecture that motivated it from the start. The huge model running far ahead of a conversation that cannot yet hold its conclusions, the medium model managing the gap between what is known and what can presently be assimilated, the nested repair budgets that let a fast process's error become someone else's reachable problem rather than a foreclosure, the human occupying the slowest and highest-budget tier of the same hierarchy rather than standing outside it — none of this was visible from Section I's starting complaint that speed and intelligence had been wrongly equated. It became visible only because the essay refused, at each stage, to let a question resolve before its own architecture allowed it to, and refused, at the very end, to let its own scope resolve into something larger than it had actually earned — which is a fitting way for an essay about premature resolution to have been built, and about the difference between a boundary honestly stated and a boundary quietly crossed.

The deepest systems, on the account this essay has reached, are not the ones that answer fastest, and not the ones built from the largest model. They are the ones able to keep a

difficult question alive across however many clocks, representations, and repair budgets it takes to reach it; to accept a small intervention without losing the trajectory that intervention depended on; to let genuine learning and mere representational repair be told apart rather than conflated; and to move, one adequate transition at a time, toward explanations, corrections, and resolutions that could never responsibly have been produced all at once — while knowing, and saying, where that account stops.

References

[1] Fant, K. M. *Computer Science Reconsidered: The Invocation Model of Process Expression*. Hoboken, NJ: Wiley-Interscience, 2007.

[2] Smith, B. C. *On the Origin of Objects*. Cambridge, MA: MIT Press, 1996.

[3] Lamport, L. "Time, Clocks, and the Ordering of Events in a Distributed System." *Communications of the ACM* 21, no. 7 (1978): 558–565.

[4] Chandy, K. M., and L. Lamport. "Distributed Snapshots: Determining Global States of Distributed Systems." *ACM Transactions on Computer Systems* 3, no. 1 (1985): 63–75.

[5] Hewitt, C., P. Bishop, and R. Steiger. "A Universal Modular ACTOR Formalism for Artificial Intelligence." In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, 235–245, 1973.

[6] Hoare, C. A. R. *Communicating Sequential Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[7] Lynch, N. A. *Distributed Algorithms*. San Francisco: Morgan Kaufmann, 1996.

[8] Hennessy, J. L., and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. 6th ed. Cambridge, MA: Morgan Kaufmann, 2019.

[9] Denning, P. J. "The Locality Principle." *Communications of the ACM* 48, no. 7 (2005): 19–24.

[10] Miller, R. E. "A Comparison of Some Theoretical Models of Parallel Computation." *IEEE Transactions on Computers* C-17, no. 8 (1968): 710–717.

[11] Baudet, G. M. "Asynchronous Iterative Methods for Multiprocessors." *Journal of the ACM* 25, no. 2 (1978): 226–244.

[12] Bertsekas, D. P., and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[13] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need." In *Advances in Neural Information Processing Systems 30*, 5998–6008, 2017.

[14] Brown, T. B., et al. "Language Models are Few-Shot Learners." In *Advances in Neural Information Processing Systems 33*, 1877–1901, 2020.

[15] Kaplan, J., et al. "Scaling Laws for Neural Language Models." arXiv:2001.08361, 2020.

[16] Hoffmann, J., et al. "Training Compute-Optimal Large Language Models." In *Advances in Neural Information Processing Systems 35*, 30016–30030, 2022.

[17] Hinton, G., O. Vinyals, and J. Dean. "Distilling the Knowledge in a Neural Network." arXiv:1503.02531, 2015.

[18] Buciluǎ, C., R. Caruana, and A. Niculescu-Mizil. "Model Compression." In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 535–541, 2006.

[19] Romero, A., N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. "FitNets: Hints for Thin Deep Nets." In *International Conference on Learning Representations*, 2015.

[20] Ba, J., and R. Caruana. "Do Deep Nets Really Need to Be Deep?" In *Advances in Neural Information Processing Systems 27*, 2654–2662, 2014.

[21] Shazeer, N., et al. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer." In *International Conference on Learning Representations*, 2017.

[22] Fedus, W., B. Zoph, and N. Shazeer. "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity." *Journal of Machine Learning Research* 23, no. 120 (2022): 1–39.

[23] Leviathan, Y., M. Kalman, and Y. Matias. "Fast Inference from Transformers via Speculative Decoding." In *Proceedings of the 40th International Conference on Machine Learning*, 19274–19286, 2023.

[24] Chen, C., S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper. "Accelerating Large Language Model Decoding with Speculative Sampling." arXiv:2302.01318, 2023.

[25] Dao, T., D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness." In *Advances in Neural Information Processing Systems 35*, 16344–16359, 2022.

[26] Kwon, W., Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. "Efficient Memory Management for Large Language Model Serving with PagedAttention." In *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626, 2023.

[27] Dai, Z., Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2978–2988, 2019.

[28] Rae, J. W., A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. "Compressive Transformers for Long-Range Sequence Modelling." In *International Conference on Learning Representations*, 2020.

[29] Wu, Y., M. Rabe, D. Hutchins, and C. Szegedy. "Memorizing Transformers." In *International Conference on Learning Representations*, 2022.

[30] Borgeaud, S., et al. "Improving Language Models by Retrieving from Trillions of Tokens." In *Proceedings of the 39th International Conference on Machine Learning*, 2206–2240, 2022.

[31] Lewis, P., et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." In *Advances in Neural Information Processing Systems 33*, 9459–9474, 2020.

[32] Park, J. S., J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. "Generative Agents: Interactive Simulacra of Human Behavior." In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[33] Yao, S., J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. "ReAct: Synergizing Reasoning and Acting in Language Models." In *International Conference on Learning Representations*, 2023.

[34] Schick, T., et al. "Toolformer: Language Models Can Teach Themselves to Use Tools." In *Advances in Neural Information Processing Systems 36*, 68539–68551, 2023.

[35] Wang, G., Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. "Voyager: An Open-Ended Embodied Agent with Large Language Models." arXiv:2305.16291, 2023.

[36] Clark, A., and D. J. Chalmers. "The Extended Mind." *Analysis* 58, no. 1 (1998): 7–19.

[37] Hutchins, E. *Cognition in the Wild*. Cambridge, MA: MIT Press, 1995.

[38] Clark, A. *Supersizing the Mind: Embodiment, Action, and Cognitive Extension*. Oxford: Oxford University Press, 2008.

[39] Simon, H. A. *The Sciences of the Artificial*. Cambridge, MA: MIT Press, 1969.

[40] Newell, A., and H. A. Simon. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[41] Newell, A. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press, 1990.

[42] Minsky, M. *The Society of Mind*. New York: Simon & Schuster, 1986.

[43] Kahneman, D. *Thinking, Fast and Slow*. New York: Farrar, Straus and Giroux, 2011.

[44] Norman, D. A. *The Psychology of Everyday Things*. New York: Basic Books, 1988.

[45] Card, S. K., T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

[46] Miller, R. B. "Response Time in Man-Computer Conversational Transactions." In *Proceedings of the AFIPS Fall Joint Computer Conference*, 267–277, 1968.

[47] Nielsen, J. *Usability Engineering*. Boston: Academic Press, 1993.

[48] Ashby, W. R. *An Introduction to Cybernetics*. London: Chapman & Hall, 1956.

[49] Wiener, N. *Cybernetics: Or Control and Communication in the Animal and the Machine*. Cambridge, MA: MIT Press, 1948.

[50] Conant, R. C., and W. R. Ashby. "Every Good Regulator of a System Must Be a Model of That System." *International Journal of Systems Science* 1, no. 2 (1970): 89–97.

[51] Beer, S. *Brain of the Firm*. London: Allen Lane, 1972.

[52] Sutton, R. S., and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT Press, 2018.

[53] Cover, T. M., and J. A. Thomas. *Elements of Information Theory*. 2nd ed. Hoboken, NJ: Wiley, 2006.

[54] Shannon, C. E. "A Mathematical Theory of Communication." *Bell System Technical Journal* 27 (1948): 379–423, 623–656.

[55] Tishby, N., F. C. Pereira, and W. Bialek. "The Information Bottleneck Method." arXiv:physics/0004057, 2000.

[56] Friston, K. "The Free-Energy Principle: A Unified Brain Theory?" *Nature Reviews Neuroscience* 11 (2010): 127–138.

[57] Varela, F. J., E. Thompson, and E. Rosch. *The Embodied Mind: Cognitive Science and Human Experience*. Cambridge, MA: MIT Press, 1991.

[58] Varela, F. J. "The Specious Present: A Neurophenomenology of Time Consciousness." In *Naturalizing Phenomenology*, edited by J. Petitot, F. J. Varela, B. Pachoud, and J.-M. Roy, 266–314. Stanford: Stanford University Press, 1999.