An Algebra of Agents

Synthesis of Parts I to V

Matthew Long
The YonedaAI Collaboration, YonedaAI Research Collective
Chicago, IL
matthew@yonedaai.com

September 2026

Abstract

Agent systems are assembled from a small vocabulary of loops, handoffs, retries, supervisors, approval steps and policies, and each item in that vocabulary has a mature theory in a different field. The five Parts of this series give all of them one mathematical object and prove, for each construction, which laws it obeys and at which relation. This synthesis reads the five Parts as one argument. Part I fixes an agent as an effectful Mealy coalgebra with typed termination and failure as a terminal outcome, over one of three commutative effect monads, and orders the relations the series uses. Part II proves the composition laws, Part III the wiring laws, Part IV the execution laws, and Part V reads policies, plans, options, shields and controllers as terms and proves properties of their closed loops. We assemble the sixteen-field projection table, giving for each field the named mechanism, its translation into a term, and the adequacy result actually proved, which is a soundness theorem, a one-way simulation or a proved limitation. We derive five properties that no single Part states, each from results of two or more Parts, and we contribute one theorem: restricted to marker-free composition terms under the lockstep scheduler with no crashes, the wiring and execution layers coincide with the composition semantics, so two such terms are bisimilar in the extension exactly when they are bisimilar in Part II. The theorem rests on its proof alone. The executable checks that accompany the series are finite-model checks, not verification.

1 Introduction

A system called agentic is a loop that observes and acts, a handoff from one worker to the next, a retry with a budget, a supervisor that restarts a crashed child, a queue between two workers, a human who approves an action, and a policy that picks a tool. Actor systems, process calculi, Erlang, workflow engines, durable execution engines, control theory, planning, reinforcement learning, session types and formal methods each own one of these constructions and each has its own object, its own equality and its own theorems. The theorems do not transfer, because the objects differ. The series *An Algebra of Agents* proposes one object and shows what can and cannot be proved about the constructions once they are terms over it, under one claim, quoted here as every Part quotes it.

> Part I fixes the object: discrete time, an effect monad T drawn from the list Id, $\mathcal{P}$, $\mathcal{D}$, and agents as effectful Mealy coalgebras with typed termination and failure as a terminal outcome. Parts II to IV introduce the operators under additional assumptions stated in each Part and prove, for each operator, that it is well-defined on agents, which relation

it respects, and which laws hold at that relation: Part II for ;, $\oplus$, $\otimes$, $\triangleright$ and μ; Part III for Tr, channels, $\|$ and Gate_H; Part IV for Sup, Persist and Replay; Part V introduces no operator and interprets policies, plans, options, shields and controllers as terms of Parts I to IV, proving properties of their closed loops. For each of sixteen fields the series gives an explicit translation of one named mechanism into a term of the algebra together with an adequacy result for that translation, which is a soundness theorem, a one-way simulation, or a proved limitation; no field is claimed to reduce to the algebra.

This synthesis adds one theorem to that claim: on marker-free composition terms under the lockstep scheduler with no crashes, the wiring layer of Part III and the execution layer of Part IV are conservative over the composition algebra of Part II, so two such terms are bisimilar in the extension exactly when they are bisimilar in Part II.

The object is a Mealy machine with three changes. Its step function lands in an effect monad, so that a deterministic worker, a nondeterministic router and a sampling policy are the same kind of thing. It has typed termination: an agent starts at a value and, if it halts, hands a value on, which is what makes a handoff composable and gives the eventual category its objects. And failure is a terminal outcome rather than an effect, so that recovery is an operator available at every instance of T. Everything else in the series is a construction on this object or a statement about one.

Four relations are needed rather than one, because different laws are true at different grains. Bisimilarity $\sim$ is the finest and is what the composition and wiring operators respect. Trace equivalence $\simeq_{\mathrm{tr}}$ is what a black-box observer sees. Accepted-trace equivalence $\simeq_{\mathrm{acc}}$ forgets runs that never halt and is where the laws about divergence hold. Stuttering-closed refinement $\sqsubseteq$ is a preorder and is what a supervised or retried execution satisfies against its crash-free specification. Part I proves the inclusions among them once, and every later result names one relation.

The imports run in one direction. Part I exports the object and its relations; Part II exports the operators as constructions that respect bisimilarity, with their laws; Part III exports feedback and the wires built from it; Part IV exports the runtime under which terms and wires execute; Part V instantiates the policy component of Part I's step with the policy classes of decision theory and control. The direction has two consequences. A bisimilar component may be substituted at any layer without disturbing the layers below, because every operator of Parts II and III respects bisimilarity and the constructions of Part IV are compared at refinement over marker erasure, which bisimilarity implies. And the hypotheses accumulate: the theorems of each layer carry the assumptions of the layers below together with their own, so a result about a supervised, wired, composed policy holds only where all five sets of assumptions do.

Sixteen mechanisms of agent systems translate into terms with adequacy results of three kinds: nine soundness theorems, five one-way simulations and three proved limitations, one mechanism carrying both a simulation and a limitation. Five further properties are invisible to any single Part and follow from two or more together. A retry needs a clock to denote anything; supervision is transparent only with persistence; logged model calls replay deterministically inside a workflow; human approval is a shield whose fallback is idling; and value composes along a handoff. Each of the five is proved by citation, with the hypotheses of the cited results carried along.

The series is a unifying semantics with proved laws, not new category theory. Bisimulation, universal coalgebra, Kleene algebra with tests, concurrent Kleene algebra, the delayed trace, Kahn networks, supervision trees, durable functions, options and shields all exist in the literature and are cited where they enter. What is new is that the results hold together for one signature at named relations, and that several of the negative results fall out of that signature meeting the operators: the exchange law fails for the synchronous product even under stuttering refinement and holds for the derived interleaving only at accepted traces (Part II, Counterexample 10.3 and Part III,

Theorem 4.12); an unbounded retry denotes nothing until a delay separates the attempts (Part II, Proposition 7.3); supervision is transparent only with checkpointing at every tick (Part IV, Theorem 8.3 and Counterexample 8.5); and a human approval gate is a shield whose fallback is idling (Part V, Proposition 9.7).

2 The framework

2.1 The object and the relations

Fix a pointed input alphabet I with idle symbol ε, a pointed output alphabet Σ with idle symbol τ, a finite failure alphabet E, and an effect monad T that is one of Id, the finite nonempty powerset monad $\mathcal{P}$, and the monad $\mathcal{D}$ of finite-support rational distributions (Part I, Definitions 2.1 and 2.2). An agent of sequential type $X \Rightarrow Y$ is a tuple $(S, \mathsf{init}, \mathsf{step}, \mathsf{halt})$ with

$$\mathsf{init} : X \to S, \qquad \mathsf{step} : S \times I \to T(\Sigma \times S), \qquad \mathsf{halt} : S \to Y + E + 1,$$

subject to the stuttering requirement that a state whose outcome is not running emits τ and stays on every input (Part I, Definition 2.10). It is a pointed coalgebra of $F(S) = (Y + E + 1) \times (T(\Sigma \times S))^I$. Halting and failing are evaluated between ticks and consume no tick, and the value handed on at a halt is what the next component starts from. The interface (I, Σ) says what crosses the boundary during a tick and is fixed for a whole category of agents; the sequential type says what is handed in and handed on and is what composition types.

The semantics $[\![A]\!] : X \times I^* \to T(\Sigma^* \times (Y + E + 1))$ sends a start value and an input word to the effectful collection of output words paired with the outcome after that many ticks (Part I, Definition 3.2). Bisimilarity $\sim$ is defined by relation lifting of T on states related by the initialisers and preserving halt (Part I, Definition 4.1), with the identity lifting for Id, the Egli-Milner lifting for $\mathcal{P}$ and the Larsen-Skou lifting for $\mathcal{D}$ (Part I, Lemma 2.7). Trace equivalence is equality of $[\![\cdot]\!]$; accepted-trace equivalence compares only the runs whose outcome is a value, recorded at the first tick at which it is one (Part I, Definition 3.5); refinement $A \sqsubseteq B$ is inclusion of stuttering-closed observation sets, and of their supports at $\mathcal{D}$ (Part I, Definition 5.4), the discrete-tick form of the stuttering-insensitive refinement of Abadi and Lamport [1]; and $\sim_{\mathrm{erase}}$ is bisimilarity after the reserved marker outputs are replaced by τ (Part I, Definition 5.1). Part I orders them:

$$A \sim B \implies A \simeq_{\mathrm{tr}} B \implies A \simeq_{\mathrm{acc}} B, \qquad A \sim B \implies A \sqsubseteq B \text{ and } B \sqsubseteq A,$$

with the first implication an equivalence at Id and strict at $\mathcal{P}$ and $\mathcal{D}$ (Part I, Theorem 4.4, Corollary 4.5, Proposition 5.5, Theorem 6.2 and Proposition 4.6). A law proved at $\sim$ therefore holds at every coarser relation, and a law proved at $\simeq_{\mathrm{tr}}$ or $\simeq_{\mathrm{acc}}$ does not lift. That asymmetry is the reason the series proves congruence at $\sim$ and states the laws about divergence, about unguarded choice and about crashes at the coarser relations where they are true.

2.2 Layers and imports

Every shared object of the series is defined once, by the lowest-numbered Part that needs it, and every import runs from a lower-numbered Part to a higher one, as Figure 1 records. The discipline is what makes the cross-Part results of Section 10 well typed: a theorem of Part V about the approval gate is a theorem about Part III's gate and not about a restatement of it, and a theorem of Part IV about a supervised product is about Part II's product. No Part cites a result of a higher-numbered Part except in remarks without labels, so the dependency graph of the series is acyclic, and the synthesis, which cites all five Parts, is the only place where a statement can use every layer at once.

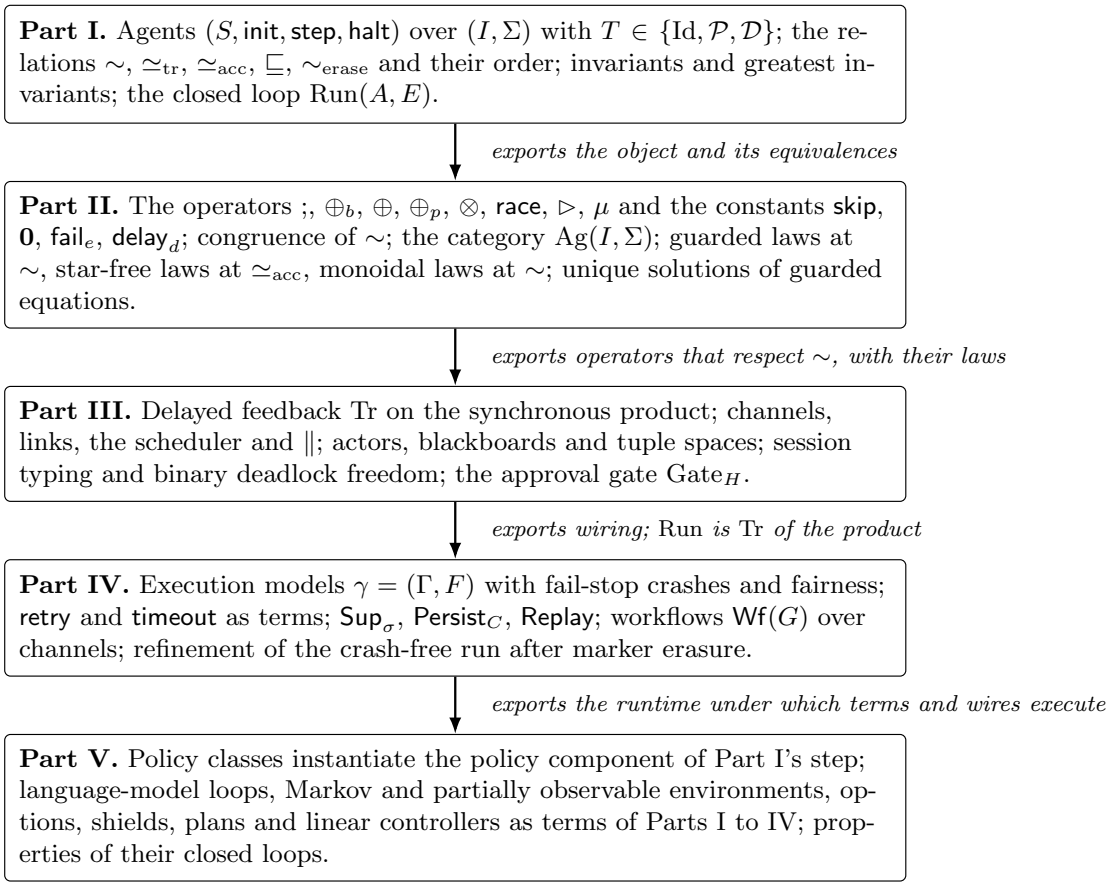

Figure 1: The composition diagram of the series. Each arrow is an import in one direction, and every shared object is defined once, by the lowest-numbered Part that needs it.

Each layer adds hypotheses that the layer below does not need, and Table 1 collects them. They are hypotheses of theorems, not properties of the object, and every result of the series carries the ones it consumes.

2.3 Prior work

The question that organises the prior work is which object deserves the name, and the answer the series gives is the one Russell and Norvig's distinction between agent function and agent program already implies [27]: the program is a coalgebra and the function is its semantics, and the two have different equalities. Universal coalgebra supplies bisimulation, coinduction and finality [28], and the probabilistic lifting is Larsen and Skou's [23]; Part I proves its lifting calculus at the three instances by hand rather than importing the general theory, so that every step can be checked instance by instance.

The equational side sits on the scale from Kleene algebra with tests [22] through its guarded fragment [31] to concurrent Kleene algebra with the exchange law [16]. Part II places the agent signature on that scale: the guarded system is sound at bisimilarity for all three monads, the star-free part of the unguarded system is sound at accepted traces on the engaged fragment for $\mathcal{P}$, and the exchange law fails for the synchronous product. Part III recovers the exchange inclusion, strictly and at accepted traces with idle ticks deleted, for the interleaving built from channels.

Table 1: The assumptions each Part adds beyond those of the Parts below it. Every result of the series carries the hypotheses it uses, and dropping one removes the object the result is about rather than weakening the result.

Part	Added assumptions
I	Time is discrete and synchronous. The effect monad is one of three commutative monads. State spaces are finite wherever a decision procedure or a Markov kernel is asserted.
II	Loop bodies consume a tick. The arguments of the unguarded and probabilistic choices are running at their initial states. The effect monad is commutative.
III	The feedback register holds its value for exactly one tick. Traced ports are hidden from the interface. Every parallel statement is made either for components coupled only by channels or for a shared object that serves one operation per tick, never for both.
IV	Crashes occur only at tick boundaries. A crash tick is isolated, so components other than those being reset receive the idle input. Supervised children are deterministic and do not fail internally. The environment is quiescent on idle output and on markers.
V	The interaction is epoch aligned. Decision models are finite with rational data and a discount in $[0, 1)$. Option durations have finite expectation. The shielded predicate is controllable and the environment is idle stationary. The plant is a discrete-time linear system with real matrices.

Feedback is the delayed trace of Sprunger and Katsumata [32] rather than the trace of Joyal, Street and Verity [19], and the gap between the two is exactly one tick.

The coordination constructs are older than the categorical ones: actors [2, 3], generative communication [13], blackboards [11], linearizability [15], session types [17, 18]. The execution constructs come from fail-stop processors [30], supervision trees [5], coordinated checkpointing [9], durable functions [7], Kahn networks [21, 24] and the series-parallel classification [34]. The decision constructs come from discounted dynamic programming [26], options [33], belief states [20], shielding [4], nondeterministic planning [10] and sampled-data control [6]. The series adds no primitive to any of these. It gives them one carrier and proves which of their theorems survive translation, at which relation, and which do not.

3 Part I: the object

Part I's object is a Mealy machine with three modifications, and each is forced by something a later Part needs. The step function lands in an effect monad (Definition 2.2), so that a deterministic worker, a nondeterministic router and a sampling policy differ only in the instance of T. Termination is typed: an agent starts at a value of type X and hands on a value of type Y when it halts (Definition 2.10), which is what makes a handoff composable and what gives the category of Part II its objects. And failure is a third outcome alongside halting and running rather than an effect, so that recovery becomes an operator available at every instance of T instead of a construct tied to an exception monad. A state that has halted or failed is required to stutter, emitting τ and staying on every input; imposing the rule on the coalgebra is what makes halting silent and free of ticks, and what makes stuttering closure a congruence on observations.

The coalgebraic apparatus is proved at the three instances by hand rather than imported. Relation lifting is the identity for Id, the Egli-Milner lifting for $\mathcal{P}$ and the Larsen-Skou coupling for $\mathcal{D}$ (Lemma 2.7), and an eight-clause calculus (Lemma 2.8) records how it interacts with unit, bind, graphs of functions and relational composition; the clause that pulls a coupling back along a function (Lemma 2.9) is the concrete content of weak-pullback preservation for $\mathcal{D}$ and the only place where a division is performed. Every congruence proof of the later Parts, and both lemmas of

Section 9, are applications of two clauses of that calculus.

Bisimilarity (Definition 4.1) is the finest of the series' relations and implies all the others. Bisimilar agents have equal decorated semantics at all three monads (Theorem 4.4), from which trace equivalence (Definition 3.3), accepted-trace equivalence (Definition 3.5 and Corollary 4.5) and mutual stuttering-closed refinement (Definition 5.4 and Proposition 5.5) follow at once. The implication reverses at Id, where the trace map is the unique morphism into a final coalgebra of causal halting stream functions and $\sim$ coincides with $\simeq_{\mathrm{tr}}$ (Theorem 6.2), and it fails at $\mathcal{P}$ and $\mathcal{D}$, where the two-branch witness of process algebra, transported to states with outcomes, is trace equivalent but not bisimilar (Proposition 4.6). The order is what lets every later law be proved once, at bisimilarity, and read at every coarser relation.

Safety questions about a single agent have exact answers. Invariants (Definition 8.2) are closed under arbitrary unions and intersections, the greatest invariant inside a predicate Φ is the greatest fixed point of $\Psi \mapsto \Phi \cap \mathsf{pre}(\Psi)$, and an invariant inside Φ exists exactly when the reachable set lies in Φ (Theorem 8.3); on a finite state space the fixed point is reached in at most $|\Phi|$ iterations. Part IV's restart budget and Part V's shields are invariants in this sense. Finite agents also have minimal quotients: the quotient by state bisimilarity is bisimilar to the agent, has no two bisimilar states, is computable by signature refinement, and is canonical once unreachable states are discarded (Theorem 7.2).

An agent is closed by an environment co-agent over the dual interface (Definition 9.1), and the closed loop $\mathrm{Run}(A, E)$ passes each side's output to the other through a register that holds it for one tick, initialised at the idle pair (Definition 9.2). At $\mathcal{D}$ with finite state spaces the closed loop is a time-homogeneous Markov chain whose kernel is the product of the component kernels, in either binding order (Proposition 9.3). The register belongs in the state space, since projecting the chain onto $S_A \times S_E$ destroys the Markov property (Remark 9.5). Every construction of Parts III and V inherits that one-tick latency, where it appears as channel latency and as the three-tick decision epoch.

Two further definitions carry weight downstream. The policy-update factorisation $\mathsf{step}(s, i) = \pi(s, i) \gg\!= \lambda(\sigma, k).\, \eta(\sigma, \delta(s, i, \sigma, k))$ (Definition 2.13) isolates the effect into a choice k: Part IV's log records the sequence of choices, which is why replay is deterministic, and Part V's policy classes are constraints on π alone. The observation policy (Definitions 5.1 and 5.2) reserves a marker subset of the output alphabet, defines erasure and $\sim_{\mathrm{erase}}$, hides traced ports, and fixes what a fair scheduler and a crash point are, so that Parts III and IV cite one convention rather than defining two. Part I adds three assumptions beyond the object: discrete synchronous time, one of three monads, and finite state spaces where decidability or a kernel is asserted.

4 Part II: the composition laws

Agents form a category, and bisimilarity is a congruence for every operator of the signature. Sequential composition $A\,;B$ replaces a halted state of A by the initial state of B at the halting value, between two ticks and silently, and propagates failure (Definition 4.6); with the identity skip_X, which halts at once with its start value (Definition 4.1), it makes $\mathrm{Ag}(I, \Sigma)$, whose objects are value sets and whose morphisms are agents modulo $\sim$, a category for every interface, failure alphabet and monad (Theorem 6.1). The associativity proof handles a cascade of two silent handoffs at one boundary, which is the finite form of what guardedness later forbids. Congruence is proved one operator at a time for ;, the three choices, the product, the race, recovery, delay and guarded recursion, using that halting and failing are state predicates (Theorem 5.1); every substitution of a bisimilar component in a later Part is an application of it.

The guarded fragment is a model of guarded Kleene algebra with tests at bisimilarity. Guarded choice $A \oplus_b B$ selects a branch from the start value before the first tick (Definition 4.10), the test agent $[b]$ is $\mathsf{skip} \oplus_b \mathbf{0}$ with $\mathbf{0}$ the agent that never halts and emits τ forever (Definitions 4.2 and 4.11), and the loop $\mathsf{while}_b(A)$ is the guarded recursion $\mu\mathsf{X}.((A\,;\mathsf{X}) \oplus_b \mathsf{skip})$ (Definition 4.22). With those readings the axioms of Smolka and coauthors [31] hold at $\sim$ for all three monads on terms whose loop bodies consume a tick (Theorem 8.1), with one exception: right annihilation $e\,;\mathbf{0} \equiv \mathbf{0}$ fails at $\sim$, because everything e emitted is still in the trace, and holds only at $\simeq_{\mathrm{acc}}$, where an agent that never halts has no accepted trace (Proposition 8.2). Left annihilation $\mathbf{0}\,; e \sim \mathbf{0}$ holds outright.

Guarded equations have unique solutions, and the condition is not decorative. A context is guarded when the silent rewriting that performs handoffs, recoveries and unfoldings terminates from every runtime term (Definition 4.19), a sufficient syntactic criterion decides it (Lemma 4.23), and $\mu\mathsf{X}.F(\mathsf{X})$ is then the agent whose states are settled runtime terms (Definition 4.20). For guarded F the equation $\mathsf{X} = F(\mathsf{X})$ has a solution unique up to $\sim$, by a bisimulation-up-to argument in the form Sangiorgi gives [29] (Theorem 7.1). The retry context $A \rhd \mathsf{X}$ is unguarded whenever A can fail without consuming a tick, because recovery re-enters a failed initial state without end, while $A \rhd (\mathsf{delay}_d\,;\mathsf{X})$ is guarded for every $d \geq 1$ (Proposition 7.3). Backoff is therefore the condition under which an unbounded retry denotes anything.

The unguarded choice is weaker than a Kleene algebra, and the weakness comes from typed termination rather than from nondeterminism. Because the halt outcome is a function of the state, an unguarded choice $A \oplus B$ over $\mathcal{P}$ must select its branch by a transition, which costs a tick (Definition 4.12), and an accepted trace is read after a tick and never before the first. Three consequences are proved. Accepted-trace equality is not a congruence for sequential composition, since skip and delay_1 have the same accepted traces while $\mathsf{skip}\,;\mathsf{delay}_1$ and $\mathsf{delay}_1\,;\mathsf{delay}_1$ do not (Counterexample 9.2). The algebra of accepted-trace sets has no multiplicative unit, because no accepted trace is empty (Proposition 9.3). And left distributivity $A\,;(B \oplus C) \sim (A\,;B) \oplus (A\,;C)$ fails at bisimilarity, because the two sides commit to a branch at different ticks (Counterexample 9.6). What survives is a star-free idempotent semiring with tests at $\simeq_{\mathrm{acc}}$ on the engaged fragment, the agents running at their initial states, with both distributive laws (Theorem 9.4); an iterate that unfolds and misses the induction axiom by one idle tick (Proposition 9.5); a one-sided structure at $\sim$ with right distributivity only (Proposition 9.7); and the skew commutativity and associativity of the probabilistic choice $\oplus_p$ at $\mathcal{D}$ (Proposition 9.9).

The synchronous product is symmetric monoidal and is not the operator of concurrent Kleene algebra. $A \otimes B$ steps both factors in lockstep on the product interface and halts when both have halted (Definition 4.14); the race $\mathsf{race}(A, B)$ has the same step, halts as soon as one factor does, and fails only when both have (Definition 4.16). Both are associative, commutative and unital up to $\sim$ for every commutative T (Theorem 10.1), and the writer monad on a free monoid, which is not commutative, breaks the commutativity law, so the hypothesis does work. The exchange law relating $(a\,;c) \otimes (b\,;d)$ to $(a \otimes b)\,;(c \otimes d)$ fails in both directions even under $\sqsubseteq$ as soon as a and b have different durations, because a synchronous product does not release its value until the slower factor halts and the waiting is visible in the other component's output (Counterexample 10.3); it holds at $\sim$ when the two left factors halt in step (Proposition 10.4). Sequencing does not distribute over the product either (Counterexample 10.5), so fan-out of a stream is a wiring question for Part III and not an operator of Part II.

Recovery is a handler-shaped operator with the laws one expects and without one that a framework might assume. $A \rhd B$ replaces a failed state of A by the initial state of B at the failure reason (Definition 4.17); it is associative, $\mathsf{skip} \rhd B \sim \mathsf{skip}$, and $\mathsf{fail}_e \rhd B$ is B started at e (Proposition 11.1). It does not distribute over sequencing: $(A\,;B) \rhd C$ replaces the whole sequence on a failure of A, while $(A \rhd C)\,;(B \rhd C)$ replaces only the failing stage and resumes (Counterexample 11.2).

Part II adds three assumptions to Part I's: loop bodies are guarded, the arguments of $\oplus$ and $\oplus_p$ are engaged, and T is commutative. Its closing observation, that commutativity of the product is a property of the semantics that an execution model need not inherit (Remark 13.1), is where Part IV begins.

5 Part III: wiring

Wiring is one operator, and it is a delayed trace rather than a trace. Part III separates the port dimension from the value dimension: a process is an agent of type $1 \Rightarrow 1$ that never halts (Definition 2.1), pipelining $A \gg B$ composes processes inside a tick (Definition 2.2), and the wiring category $\mathcal{W}_T$ of processes modulo $\sim$ with $\otimes$ as product is symmetric monoidal (Definition 2.3, Lemma 2.4). The delayed trace $\mathrm{Tr}^{C,c_0}(A)$ feeds the C-output of tick t to the C-input of tick $t+1$ through a one-tick register and hides the port (Definitions 3.1 and 3.2), and it respects $\sim$ (Lemma 3.3). It satisfies naturality in the non-feedback ports, vanishing, superposing and a delayed yanking law $\mathrm{Tr}(\mathrm{swap}) \sim \mathsf{delay}_1$, at $\sim$ and for all three monads (Theorem 3.5), and it fails ordinary yanking on any alphabet with two letters, because closing a symmetry wire on itself produces a register and not an identity (Counterexample 3.6). The structure is that of Sprunger and Katsumata [32] rather than of Joyal, Street and Verity [19], and the gap is exactly one tick. Part I's closed loop is that operator applied to the product of an agent and its environment, with the register at (ε, τ) (Proposition 3.8), so the series has one feedback construction and one latency convention.

Coupling by channels and coupling by a shared object obey different laws, and Part III never mixes them. A channel is a deterministic process with a queue (Definition 4.1), linking through an object is a trace over the object's ports (Definition 4.2), and the interleaving $\|$ for $T = \mathcal{P}$ advances exactly one component per tick, chosen by a scheduler agent that offers every index at every tick (Definitions 4.3 to 4.5). In the interleaving fragment, where components share only channels with one writer and one reader each (Definition 4.7), the accepted-trace set of $A \,\|\, B$ with idle ticks deleted is the shuffle of the component sets (Theorem 4.10), and $\|$, $;$, $\oplus$, $\mathbf{0}$ and skip form a concurrent semiring satisfying the exchange inclusion of Hoare, Möller, Struth and Wehrman [16], strictly (Theorem 4.12). The synchronous product fails that inclusion by Part II's Counterexample 10.3; the derived interleaving satisfies it because an interleaving does not wait.

In the shared-object fragment the shuffle law fails and linearizability takes its place. An object system applies at most one operation per tick and returns the result at the client's next scheduled tick (Definition 4.6). Every history of such a system, and in particular of a blackboard with knowledge sources and a control agent (Definitions 6.1 to 6.3), is linearizable with application-tick order as witness (Proposition 6.4), and a replicated blackboard with asynchronous propagation has a history that is not (Counterexample 6.5). A tuple space simulates the Linda primitives one way (Definition 6.6 and Proposition 6.8), two consumers competing for one tuple have at least two outcomes (Proposition 6.9), and a single unbounded channel between deterministic agents has exactly one outcome under fairness (Proposition 6.10). The last of these is the two-node case of Kahn determinacy and is the base case of Part IV's workflow theorem.

Actor isolation is a theorem once the mailbox is the only shared component. An actor is a message-driven behaviour linked through an unbounded mailbox at an address, and an actor system is an object system over a fixed finite family of addresses (Definitions 5.1 and 5.2). The receive, send and become rules of Agha, Mason, Smith and Talcott [3] are simulated one way by the system, with creation outside the fragment (Proposition 5.3), and an actor's state after any run is the iterated step of its behaviour on the sequence of messages it dequeued alone, so a bisimilar behaviour substitutes bisimilarly (Theorem 5.4).

Session typing is a predicate on port traces and adds no field to the agent. A session type is a coinductive tree over sends, receives, selections and branches (Definition 7.1), and $A : S$ has two clauses, fidelity, that every port trace lies in the language of S, and progress, that from every reachable non-halted state the agent performs its next conforming action after finitely many scheduled ticks and halts only at the end (Definition 7.3). Two agents typed by dual types (Definition 7.2), linked by their only channels under a fair scheduler, never reach a deadlock and halt exactly when the session ends (Theorem 7.6, with Lemma 7.5 as its invariant). The human approval gate is a session-typed link to a co-agent that answers ok, no or pending, with the wrapped agent frozen while a question is pending (Definitions 8.1 and 8.2): no action leaves the gate before an approval of that action, and under an always-approving oracle the gate and the bare agent refine each other on input words that are idle while the gate waits (Theorem 8.3); a live oracle under a fair scheduler keeps the gate deadlock-free (Corollary 8.4). Part III adds three assumptions: the register holds for one tick, traced ports are hidden, and every parallel statement lives in one fragment.

6 Part IV: execution

A crash in Part IV is an input and a marker, and an execution model is a hypothesis about the deployment. The execution interface adds a crash port to the input and a fresh marker set to the output (Definition 2.1); a fail-stop lifting makes a struck component emit the marker and stop, a reset lifting returns it to a designated state, and a failure model is a prefix-closed set of crash schedules that may be bounded and isolating (Definition 3.1). An execution model $\gamma = (\Gamma, F)$ pairs a class of Part III's scheduler agents with a failure model, and a γ-execution links the crash port to a source and hides it (Definition 3.2). Fairness is a predicate on scheduler traces (Definition 3.3), assumed rather than derived, since Theorem 1 of Fischer, Lynch and Paterson [12] rules out deriving it. Fail-stop behaviour in the sense of Schlichting and Schneider [30] follows from the typing of the alphabet: a struck component has no way to emit an ordinary output on the tick it is struck.

Timeout and retry are terms over Part II's operators and inherit their laws. The timeout races the agent against a delay and converts a win by the timer into a failure through a guarded choice (Definition 4.1); it is not running after $t + 1$ ticks for every agent and every monad (Proposition 4.2). Retry with backoff recovers into a delay and re-enters the attempt at its start value, which a task lifting keeps in the state and which is transparent up to $\sim$ (Definitions 2.2 and 4.3, Lemma 2.3); retry respects $\sim$, and its unbounded form has a unique solution exactly under Part II's guardedness condition (Proposition 4.4).

Supervision, checkpointing and replay are constructions on agents rather than primitives. A supervisor owns a vector of crash-lifted children, restarts the set its strategy names when a child is struck or fails, emits a restart marker, and dies when more than $\max_r$ restart-triggering ticks fall in a window of $\max_t$ (Definition 5.1); death is exactly budget exhaustion and is absorbing (Proposition 5.3), and the restart sets of the three strategies described by Armstrong [5] are realised one way (Proposition 5.4). Checkpointing $\mathsf{Persist}_C$ moves the restart target to the last saved state of a cut set (Definition 6.1) and commutes with ; when the cut contains the halting boundary and with $\otimes$ under synchronous cuts and synchronous crash schedules, at $\sim_{\text{erase}}$ (Proposition 6.2), both hypotheses being load-bearing (Remark 6.3). A log records inputs paired with the choice component of Part I's factorisation, and replay against it is an Id agent whose trajectory and outputs are the logged ones, for every one of the three monads; replay of a modified agent completes exactly when the log stays compatible (Definitions 7.1 and 7.2, Theorem 7.3), so a logged stochastic run is reproducible and a support-changing edit is detected on the tick where it occurs (Corollary 7.4).

The execution results are refinements over marker erasure, and each has a counterexample that

fixes its hypothesis. With deterministic, internally failure-free children checkpointed at every tick, an isolating and bounded failure model and an environment quiescent on idle output and markers (Definition 8.1), the erased observed closed system of the supervisor and the crash-free product refine each other, because a crash or restart tick is an idle position (Lemma 8.2, Theorem 8.3); the strategy is then invisible (Corollary 8.4), and a counter child without checkpointing produces a word no crash-free run produces (Counterexample 8.5). Retry of a keyed Id agent against an environment idempotent on the key refines the crash-free effect trace in both directions for every schedule with at most n crashes and halts with the same value (Definition 9.1, Theorem 9.2), while a counting environment records the duplicates (Counterexample 9.3). At-least-once delivery with a deduplicating receiver gives effectively-once effects and not exactly-once delivery.

Workflows are Kahn networks over Part III's channels, and their dependency orders exceed what the product expresses. Under the channel discipline with deterministic, blocking, productive nodes, unbounded channels, an acyclic graph and no race, retry or timeout inside a node (Definitions 10.1 and 10.2), every edge history is the same under every fair scheduler, by induction along a topological order with Part III's Proposition 6.10 as base case (Theorem 10.3). Under the early-start discipline with duration-flexible leaves, the dependency order of every $\otimes$ and ; term is series-parallel (Definition 10.5, Proposition 10.7), and the N-shaped four-node graph, whose order is the obstruction of Valdes, Tarjan and Lawler [34], is not bisimilar to any such term (Counterexample 10.9). Part IV adds four assumptions, listed in Table 1, and none of Sup, Persist and Replay is claimed to respect bisimilarity.

7 Part V: decision and control

Part V adds no operator, and a policy is not extra structure on an agent. It is the first component of the factorisation Part I already imposes, and the six policy classes, deterministic, nondeterministic, stochastic, plan, controller and language model, are constraints on π alone (Definition 3.1). The support map $\mathcal{D} \to \mathcal{P}$ is a monad morphism (Lemma 3.2), so an agent over $\mathcal{D}$ has a support abstraction over $\mathcal{P}$ that preserves positive-probability reachability (Definition 3.3, Proposition 3.4); that morphism is how a statement about a stochastic policy is derived from one about a nondeterministic policy without any law ranging over a combined model.

The one-tick register of Part I's closed loop is the finding of Part V that appears in none of the source formalisms. An agent acting at tick t acts on information the environment emitted at tick $t-1$, so a memoryless policy and a memoryless environment settle into an epoch of three ticks: the environment announces, the agent senses, the agent acts. Sensor-aligned environments, sensing leaves, emitters and actuator-aligned agents are the phase discipline (Definition 4.1) under which the epoch process is the one-step process of the textbook models, and at $\mathcal{D}$ with finite state spaces it is a Markov chain (Lemma 4.2). Misalignment is not harmless: gains that place the aligned linear loop at spectral radius one half make the loop with one extra epoch of delay diverge (Remark 11.4).

A tool loop over a language model is an agent as soon as one says what is bounded. Under the bounded-generation abstraction, a truncated context, a bounded generation length and rational weights (Definition 5.1), the model is a kernel between finite sets, the ReAct loop is the guarded recursion $\mu X.(((\mathsf{Think}\,;\mathsf{Act})\,;X) \oplus_{\mathrm{done}} \mathsf{skip})$ (Definition 5.3), and that recursion is an agent over $\mathcal{D}$ with a finite state space, bisimilar to the loop written by hand (Proposition 5.4). A logged loop replays as an Id agent by Part IV's Theorem 7.3, and replaying the log against a changed model either reproduces the run or fails on the tick where the histories diverge (Corollary 5.5). A serving stack whose sampling depends on the batch presents no kernel, and the abstraction says nothing about a change of parameters (Remark 5.2).

The value equations of decision theory are statements about closed loops of terms. Against

a finite Markov decision environment, an instance of Part I's environment (Definition 6.1), the discounted value of the closed loop of a memoryless stochastic policy agent is the policy's value, the unique solution of the evaluation equation (Definitions 6.2 and 6.3, Theorem 6.4), so evaluation and improvement are maps on agents (Corollary 6.5). An option is a guarded recursion whose exit test reads a coin drawn while sensing (Definition 7.1), and value composes along ; at an epoch-aligned halting time of finite expectation (Definition 7.2, Theorem 7.3), which is the semi-Markov equation of Sutton, Precup and Singh [33] (Remark 7.4). A finite partially observable process with rational data induces a belief agent whose state is a rational belief and whose update is Bayes' rule (Definition 8.1); the carried belief is the posterior (Lemma 8.4), the belief process is Markov on a countable space (Lemma 8.5), and the closed-loop value is the value of the belief process (Theorem 8.6).

A shield is one guarded choice, and its correctness is an invariant. With a controllable predicate, one contained in its own controllable predecessor, and a fallback that stays inside it (Definition 9.1), the shielded agent inserts a single guarded choice between sensing and emitting (Definition 9.2) and preserves the predicate for every deterministic policy against a $\mathcal{P}$ environment (Theorem 9.3), and with probability one for a stochastic policy against a $\mathcal{D}$ environment whose support is the nondeterministic successor (Proposition 9.4); a shield that never fires is invisible (Corollary 9.5). The approval gate with the safety oracle and the shield whose fallback is idling have the same marker-erased, stutter-collapsed traces and refine each other, are not bisimilar because the gate spends a tick asking, and the gate keeps the environment inside the predicate without controllability while guaranteeing no progress (Proposition 9.7, Remark 9.8).

Planning and control need no machinery beyond the closed loop. Plan terms over ;, $\oplus_b$ and guarded μ (Definition 10.1) have as executions exactly the epoch paths of their closed loops (Lemma 10.3), so the weak, strong and strong cyclic solutions of Cimatti, Pistore, Roveri and Traverso [10] are the existential-path, winning-strategy and fair-winning solutions of the reachability game, with outcome fairness (Definition 10.2) as the fairness notion, and memoryless strong cyclic tables are the guarded recursions that win on every outcome-fair path (Proposition 10.4), hence almost surely against any positive-probability resolution (Corollary 10.5). The linear loop of a plant environment and a state-feedback controller (Definition 11.1) has epoch process $x_{k+1} = (F-GK)x_k$ and converges from every start exactly when $\rho(F - GK) < 1$ (Proposition 11.2). Part V adds five assumptions, listed in Table 1, and learning, a change in the policy's parameters, is an operation on agents outside every result.

8 The projection table

Nine of the sixteen adequacy results in Table 2 are soundness theorems, five are one-way simulations, and three rows pair their positive result with a proved limitation, the tuple-space row belonging to the last two groups at once. A soundness result says the translation preserves the mechanism's stated semantics; a one-way simulation says the term simulates the mechanism and claims no converse; a proved limitation says the mechanism is not fully representable and states the obstruction. No row claims that its field reduces to the algebra. The limitations are not concessions: in each case the obstruction is what the translation makes visible, and it is proved rather than observed.

Table 2: The sixteen-field projection table. For each field the row names the mechanism translated, the term it becomes, the adequacy result proved for the translation, and the Part and result that prove it.

Field	Mechanism	Translation	Adequacy result	Where proved
Actor model	asynchronous message passing with state isolation, after Hewitt and Agha	$\mathrm{Act}_a(A)$, the behaviour A linked through the mailbox $\mathrm{Ch}_\infty(M)$ at address a	one-way simulation of the receive, send and become rules, and locality: the actor's state is a function of its initial value and the dequeued sequence	Part III, Proposition 5.3 and Theorem 5.4
Process calculi	sequential, choice and parallel composition with equational laws	$;$, $\oplus$, $\otimes$ and μ	soundness of the guarded axioms at $\sim$, of the star-free axioms at $\simeq_{\mathrm{acc}}$ on the engaged fragment, and of the monoidal laws at $\sim$	Part II, Theorems 8.1, 9.4 and 10.1
State machines	explicit labelled transitions in the sense of Mealy	the leaf step step from $S \times I$ to $T(\Sigma \times S)$	soundness: at Id the semantics is the unique morphism into the final coalgebra of causal halting stream functions, where $\sim$ and $\simeq_{\mathrm{tr}}$ coincide	Part I, Theorem 6.2
Blackboard systems	shared epistemic state with knowledge sources and a control component	the object BB, knowledge-source clients and a control agent replacing the scheduler	soundness: every history is linearizable with application-tick order as witness	Part III, Proposition 6.4
Tuple spaces	generative communication with associative matching, after Gelernter	the object TS with out, in and rd	one-way simulation of the three primitives, and a proved limitation: two consumers competing for one tuple have at least two outcomes	Part III, Propositions 6.8 and 6.9
Functional programming	pure composition with identities	the category $\mathrm{Ag}(I, \Sigma)$ with $;$ and skip	soundness of the category laws at $\sim$	Part II, Theorem 6.1
Distributed systems	fail-stop crashes, fairness and deduplicated retries	the failure model, fair scheduler traces, and $\mathsf{retry}_{n,d}$ with a deduplication key	one-way simulation: the retried run refines the crash-free effect trace in both directions after erasure, which is effectively-once and not exactly-once delivery	Part IV, Theorem 9.2
Erlang and OTP	supervision trees with restart strategies and a restart budget	Sup_σ with budget $(\max_r, \max_t)$	one-way simulation of the three restart sets, and refinement of the crash-free run in both directions under checkpointing at every tick	Part IV, Proposition 5.4 and Theorem 8.3

Field	Mechanism	Translation	Adequacy result	Where proved
Workflow and DAG systems	dependency-ordered task graphs	$\mathsf{Wf}(G)$ over the channels of Part III	soundness for the Kahn fragment, and a proved limitation: the N-shaped graph is not bisimilar to any $\otimes$ and ; term	Part IV, Theorem 10.3 and Counterexample 10.9
Durable execution	checkpointing and deterministic replay	$\mathsf{Persist}_C$, the log as a channel with a cursor, and Replay	soundness: replay determinism for all three monads, and checkpointing commutes with ; and $\otimes$ at $\sim_{\mathrm{erase}}$	Part IV, Theorem 7.3 and Proposition 6.2
Control theory	closed-loop feedback	$\mathrm{Run}(A, E)$, the delayed trace Tr, and the linear loop	soundness: the closed loop is the trace of the product, and the linear loop converges exactly when the spectral radius is below one	Part I, Definition 9.2, Part III, Proposition 3.8, and Part V, Proposition 11.2
Planning	goal-directed selection under nondeterminism	plan terms over $\oplus_b$, ; and μ	soundness: weak, strong and strong cyclic solutions are the three solution notions of the closed-loop reachability game	Part V, Proposition 10.4
RL and MDPs	stochastic policies, options and beliefs in reinforcement learning	the policy agent A_π, option terms, and belief agents	soundness: the closed-loop value equations for policies, for sequential composition at a stopping time, and for beliefs	Part V, Theorems 6.4, 7.3 and 8.6
Type theory	interface and capability constraints as session types	typing by fidelity and progress against a coinductive session type	soundness for binary sessions, and a limitation with its obstruction stated: the proof uses a single linear order of actions that a multiparty protocol lacks	Part III, Theorem 7.6 and Section 10
Formal methods	invariants, refinement and verification	invariants, $\sqsubseteq$ and shields	soundness: the greatest invariant inside a predicate is a greatest fixed point, and a shield preserves a controllable invariant for every policy	Part I, Theorem 8.3, and Part V, Theorem 9.3
Language models	probabilistic semantic transformation at inference time	Π_{LLM} over $\mathcal{D}$ with a bounded context, and the ReAct term	one-way simulation of an inference-time tool loop under the bounded-generation abstraction, with learning out of scope	Part V, Proposition 5.4

The relation at which a row holds is part of the row. The process-calculi row holds at three relations that are not interchangeable, the guarded axioms at $\sim$, the star-free axioms at $\simeq_{\mathrm{acc}}$ and only on the engaged fragment, and the monoidal laws at $\sim$; Part II's Counterexamples 9.2 and 9.6 are what prevent one relation from serving for all three. The workflow row pairs a soundness theorem with a limitation because the limitation is the reason workflow engines exist: a dependency

the product cannot express is a dependency a graph can. The language-model row is a one-way simulation under an abstraction that presents the model as a kernel on bounded contexts, and a serving stack whose sampling depends on batch composition presents no such kernel.

9 Conservativity

Parts III and IV import the operators of Part II unchanged and add constructions around terms. Theorem 9.5 says that the two additions are invisible on marker-free composition terms as long as the scheduler is lockstep and no crash occurs, so that the equational theory of Part II is neither enlarged nor shrunk by the extension. It is the one result of the series owned by the synthesis, it rests on its proof alone, and it has no finite-model check, because it is a statement about the whole equational theory of Part II terms and no finite model establishes such a statement.

9.1 Terms and the extension

Throughout, fix an interface (I, Σ), a failure alphabet E and $T \in \{\mathrm{Id}, \mathcal{P}, \mathcal{D}\}$. Let $M \subseteq \Sigma \setminus \{\tau\}$ be the reserved marker subset of Part I, Definition 5.1, and let $\mathrm{er} : \Sigma \to \Sigma$ be its erasure map.

Definition 9.1 (Part II terms)**.** A *Part II term* over a set of agent constants is a closed context in the sense of Part II, Definition 4.18: a term built from constants of the appropriate types and from skip, $\mathbf{0}$, fail_e and delay_d by $;$, $\oplus_b$, $\oplus$, $\oplus_p$, $\otimes$, race, $\rhd$ and guarded μ, respecting the type discipline of Part II, Section 4. Its *denotation* $[\![t]\!]$ is the agent Part II assigns to it by Definitions 4.1 to 4.20 there, and its *base alphabet* Σ_t is the output alphabet of $[\![t]\!]$, which Part II's operators build from the alphabets of the constants by reuse and by product. A term is *marker-free* when er is the identity on the output alphabet of every constant in it.

Part I extends erasure to a product alphabet componentwise, so er is the identity on Σ_t whenever t is marker-free. The denotation is the same coalgebra in Parts II, III and IV: Part III defines no composition operator and cites Part II for every one it uses (Part III, Section 2.1 and Table 1), and Part IV imports the same definitions under the same labels (Part IV, Section 2.1 and Table 2). What the two later Parts add is a way of *executing* a term.

Part III adds a scheduling discipline. A family of components may be composed by $\otimes$, in which every component steps at every tick, or by the interleaving $\|$ of Part III, Definition 4.5, in which a scheduler agent (Part III, Definition 4.3) supplies a run token to exactly one component per tick and the others, being quiescent, stand still. Part III's linking construction admits either discipline (Part III, Definition 4.2, where the family is composed by $\odot \in \{\otimes, \|\}$). The *lockstep scheduler* is the discipline in which every component is scheduled at every tick, which in Part IV's vocabulary is the scheduler that emits the whole component set on every tick, an admissible member of the class of Part IV, Example 3.4. Under it the family steps exactly as $\otimes$ steps it, and Part II's operators already step their components that way: $\otimes$ steps both factors, $;$ steps the active factor and hands over silently, the choices step the selected branch. So the lockstep scheduler contributes no state and no input to a Part II term, and Part III's remaining addition to a term with no channel and no shared object is the trace over a trivial port.

Part IV adds a failure model. For a finite set N of component names, the execution interface is $(I \times 2^N, \Sigma \uplus M_N)$ with marker set $M_N = \{\mathsf{crash}\} \cup \{\mathsf{restart}_J : \emptyset \neq J \subseteq N\}$ of fresh symbols (Part IV, Definition 2.1); the fail-stop lifting $A^{\perp}$ of an agent A with state space S has state space $S + \{\perp\}$,

outcome $\mathsf{halt}(\bot) = \mathrm{inm}(\mathsf{crashed})$, and step

$$\mathsf{step}^{\bot}(s,(i,\kappa)) = \begin{cases} \eta(\mathsf{crash},\bot) & \kappa \neq \emptyset \text{ and } \mathsf{halt}(s) = \mathrm{inr}(*), \\ \mathsf{step}(s,i) & \kappa = \emptyset \text{ and } \mathsf{halt}(s) = \mathrm{inr}(*), \\ \eta(\tau,s) & \text{otherwise} \end{cases}$$

(Part IV, Definition 3.1). A γ-execution links the crash port to a crash source Φ_{φ} that emits $\varphi(t)$ at tick t, through the register of Part III, Definition 3.2, initialised at the idle value $\emptyset$, and hides the port (Part IV, Definition 3.2). Every failure model contains the empty crash schedule $\varphi_{\emptyset}$, which is constantly $\emptyset$ (Part IV, Definition 3.1).

Two conventions on alphabets are needed, because the executions enlarge them. For a pointed inclusion $\Sigma \hookrightarrow \Sigma'$ of output alphabets and an agent A over (I,Σ), write $A^{\uparrow}$ for A *read over* Σ': the same state space, initialiser and outcome, with the step post-composed by T of the inclusion. Conversely, let B be an agent over (I,Σ') every reachable step of which has support inside $\Sigma \times S_B$, reachability in the sense of Part I, Definition 8.1. The *corestriction* $B|_{\Sigma}$ of B to Σ is the agent over (I,Σ) whose state space is $\mathrm{Reach}(B)$ and whose maps are those of B. It is an agent: the reachable set contains the initial states and is closed under successors, the step of a reachable state lands in $T(\Sigma \times \mathrm{Reach}(B))$ by hypothesis, and the stuttering requirement is inherited.

Definition 9.2 (Lockstep crash-free execution)**.** Let A be an agent over (I,Σ) with state space S and let N be a finite set of component names. The *lockstep crash-free execution* of A over N is the agent $\mathcal{E}(A)$ over $(I,\Sigma \uplus M_N)$ with state space $(S + \{\bot\}) \times 2^N$, initialiser $x \mapsto (\mathsf{init}(x),\emptyset)$, outcome $\mathsf{halt}(s,\kappa) = \mathsf{halt}^{\bot}(s)$, and

$$\mathsf{step}_{\mathcal{E}(A)}\big((s,\kappa),i\big) \;=\; T\big(\mathrm{id} \times (s' \mapsto (s',\emptyset))\big)\big(\mathsf{step}^{\bot}(s,(i,\kappa))\big).$$

The second component is the register through which the crash port is linked; the crash source $\Phi_{\varphi_{\emptyset}}$ is stateless and writes $\emptyset$ into it at every tick, which is why the source does not appear as a component of the state.

For a marker-free Part II term t with constants $A_1,\dots,A_n$, the *whole-term execution* is $\mathcal{E}(\llbracket t \rrbracket)$ over the singleton name set, and the *leafwise execution* $\mathcal{E}_{\ell}(t)$ is the denotation of the term obtained from t by replacing each constant A_j by $\mathcal{E}(A_j)$ over the name set $N = \{1,\dots,n\}$, with the operators of Part II interpreted over the enlarged alphabets. The output alphabet of $\mathcal{E}_{\ell}(t)$ is the *leafwise alphabet* Σ_t^{ℓ} of t, built by Part II's operators from the alphabets $\Sigma_j \uplus M_N$ of the executed leaves; the base alphabet Σ_t embeds in it by the product of the inclusions.

The two granularities are the two Part IV uses: a supervisor treats each child as a named component with its own crash port (Part IV, Definition 5.1), and a retry treats the whole attempt as one (Part IV, Definition 4.3 with Theorem 9.2). The two enlarge the alphabet differently. The whole-term execution of $A \otimes B$ is an agent over $(\Sigma_A \times \Sigma_B) \uplus M_N$, while the leafwise execution is an agent over $(\Sigma_A \uplus M_N) \times (\Sigma_B \uplus M_N)$, and the leafwise alphabet of a term therefore depends on the term's product structure, so that two terms of different shapes have leafwise executions over different alphabets and cannot be compared by bisimilarity directly. Under a nonempty crash schedule the difference is substantive: a single component emits the single marker crash where a product of two components emits the pair $(\mathsf{crash},\mathsf{crash})$, which is why Part IV's product law for checkpointing needs synchronous crash schedules (Part IV, Proposition 6.2 and Remark 6.3). Under the empty schedule no marker is ever emitted, the corestriction to the base alphabet exists, and the granularity does not matter; that is part of what Theorem 9.5 says.

The Part III addition is covered by the trivial trace. For an agent A over (I,Σ), write A^{1} for A read over the interface $(I \times 1, \Sigma \times 1)$, which is the same coalgebra up to the isomorphisms $I \times 1 \cong I$

and $\Sigma \times 1 \cong \Sigma$, and let $\mathrm{Tr}^{1,*}(A^1)$ be the delayed trace of Part III, Definition 3.2, over the one-point port. That is the wired system consisting of A alone: no channel, no shared object, the lockstep discipline, and the one feedback port the construction requires, carrying nothing.

9.2 Invisibility lemmas

Both lemmas relate an agent to a construction on it by a relation that is the graph of an injection on the reachable states, and both use three clauses of Part I's lifting calculus: the lifting of the diagonal is the diagonal (Part I, Lemma 2.8(ii)), a related pair mapped through two functions that carry the relation into a second relation stays related under the lifting of the second (Part I, Lemma 2.8(viii)), and a pair whose images under two functions are related is related under the lifting of the preimage relation (Part I, Lemma 2.9).

Lemma 9.3 (A silent crash port is invisible)**.** *Let A be an agent over (I, Σ), let N be a finite name set, and let $T \in \{\mathrm{Id}, \mathcal{P}, \mathcal{D}\}$.*

(i) $\mathcal{E}(A) \sim A^{\uparrow}$, where $A^{\uparrow}$ is A read over $\Sigma \uplus M_N$.

(ii) No reachable step of $\mathcal{E}(A)$ emits a marker, the corestriction $\mathcal{E}(A)|_{\Sigma}$ exists, and $\mathcal{E}(A)|_{\Sigma} \sim A$ over (I, Σ).

Proof. (i) Let $R = \{(s, (s, \emptyset)) : s \in S\} \subseteq S \times ((S + \{\bot\}) \times 2^N)$. Clause (B1) of Part I, Definition 4.1 holds because both initialisers send x to $\mathsf{init}(x)$ paired, on the right, with $\emptyset$. Clause (B2) holds because $\mathsf{halt}_{\mathcal{E}(A)}(s, \emptyset) = \mathsf{halt}^{\bot}(s) = \mathsf{halt}(s)$ for $s \in S$, the lifting changing the outcome only at $\bot$.

For clause (B3), fix $(s, (s, \emptyset)) \in R$ and an input i. Write $\iota : \Sigma \to \Sigma \uplus M_N$ for the inclusion and $f : S \to (S + \{\bot\}) \times 2^N$ for $s' \mapsto (s', \emptyset)$, so that the graph of f is R. Suppose first that s is running. Then the second clause of the fail-stop lifting gives $\mathsf{step}^{\bot}(s, (i, \emptyset)) = \mathsf{step}(s, i)$, and so, writing $u = \mathsf{step}(s, i) \in T(\Sigma \times S)$,

$$\mathsf{step}_{A^{\uparrow}}(s, i) = T(\iota \times \mathrm{id})(u), \qquad \mathsf{step}_{\mathcal{E}(A)}\big((s, \emptyset), i\big) = T(\iota \times f)(u),$$

the second because $\mathsf{step}^{\bot}(s, (i, \emptyset))$ lands in $T(\Sigma \times S)$ and the lifting's step reads it inside $T((\Sigma \uplus M_N) \times (S + \{\bot\}))$ before applying $\mathrm{id} \times f$. By Part I, Lemma 2.8(ii), $(u, u) \in \mathrm{Rel}(T)(\mathrm{Eq}_{\Sigma \times S})$. The two functions $\iota \times \mathrm{id}$ and $\iota \times f$ send a pair $((\sigma, s'), (\sigma, s'))$ of the diagonal to $((\iota\sigma, s'), (\iota\sigma, f(s')))$, which lies in $\mathrm{Eq}_{\Sigma \uplus M_N} \times R$, so Part I, Lemma 2.8(viii) gives

$$(T(\iota \times \mathrm{id})(u),\ T(\iota \times f)(u)) \in \mathrm{Rel}(T)(\mathrm{Eq}_{\Sigma \uplus M_N} \times R),$$

which is clause (B3) at this pair. Suppose instead that s is not running. Then $\mathsf{step}(s, i) = \eta(\tau, s)$ by the stuttering requirement of Part I, Definition 2.10, and $\mathsf{step}^{\bot}(s, (i, \emptyset)) = \eta(\tau, s)$ by the third clause of the lifting, so the two steps are $T(\iota \times \mathrm{id})(\eta(\tau, s))$ and $T(\iota \times f)(\eta(\tau, s))$, and the same two clauses of Lemma 2.8 apply. In neither case is a marker emitted, since the first clause of the lifting requires $\kappa \neq \emptyset$ and the register holds $\emptyset$.

R is therefore a bisimulation. The states $(\bot, \kappa)$ and (s, κ) with $\kappa \neq \emptyset$ lie outside R and are unreachable from the initial states, which is permitted: a bisimulation in Part I's sense is a relation containing the initial pairs and closed under transitions from the pairs it contains, and need not be total on either state space.

(ii) Every reachable state of $\mathcal{E}(A)$ has register $\emptyset$, since the register is written with $\emptyset$ at every step, so no reachable step takes the first clause of the lifting and every reachable step has support inside $\Sigma \times \mathrm{Reach}(\mathcal{E}(A))$; the corestriction exists. Restrict R to $R' = R \cap (S \times \mathrm{Reach}(\mathcal{E}(A)))$. It

still contains the initial pairs and preserves outcomes. For (B3), the steps of A and of $\mathcal{E}(A)|_\Sigma$ at a pair of R' are u and $T(\mathrm{id}\times f)(u)$ read over Σ, and their images under $T(\iota\times\mathrm{id})$ are related by $\mathrm{Rel}(T)(\mathrm{Eq}_{\Sigma\uplus M_N}\times R)$ by (i); Part I, Lemma 2.9 applied to the two functions $\iota\times\mathrm{id}$ then gives relatedness under the lifting of the preimage relation, which is $\mathrm{Eq}_\Sigma\times R$, and the witness has support inside reachable pairs, so it lies in $\mathrm{Rel}(T)(\mathrm{Eq}_\Sigma\times R')$. □

Lemma 9.4 (A trivial wire is invisible)**.** *For every agent A over (I,Σ) and every $T\in\{\mathrm{Id},\mathcal{P},\mathcal{D}\}$, $\mathrm{Tr}^{1,*}(A^1)\sim A$.*

Proof. By Part III, Definition 3.2 the trace has state space $S\times 1$, initialiser $x\mapsto(\mathsf{init}(x),*)$, outcome $\mathsf{halt}(s,*)=\mathsf{halt}(s)$, and step

$$\begin{aligned}\mathsf{step}_{\mathrm{Tr}}\big((s,*),i\big)&=\mathsf{step}_{A^1}\big(s,(i,*)\big)\gg\!=\lambda\big((\sigma,*),s'\big).\,\eta\big(\sigma,(s',*)\big)\\&=T\big(\mathrm{id}\times(s'\mapsto(s',*))\big)\big(\mathsf{step}(s,i)\big),\end{aligned}$$

using that A^1 is A up to the interface isomorphisms and that a bind whose continuation is η of a function is the action of T on that function (the monad law $T(g)(u)=u\gg\!=(\eta\circ g)$, used in Part I, Lemma 2.8(v)). The relation $\{(s,(s,*)):s\in S\}$ satisfies (B1) and (B2) by construction and (B3) by Part I, Lemma 2.8(ii) and (viii) with the functions id and $\mathrm{id}\times(s'\mapsto(s',*))$, exactly as in the previous proof. This is the vanishing clause of Part III, Theorem 3.5(ii) extended from processes to agents with halting, which the trace inherits unchanged. □

9.3 The theorem

Theorem 9.5 (Conservativity of the wiring and execution layers)**.** *Fix (I,Σ), E and $T\in\{\mathrm{Id},\mathcal{P},\mathcal{D}\}$, and let s and t be marker-free Part II terms of the same sequential type over the same interface, so that $\Sigma_s=\Sigma_t=\Sigma$. Then:*

(i) $\mathcal{E}(\llbracket t\rrbracket)\sim\llbracket t\rrbracket^\uparrow$ over $(I,\Sigma\uplus M_N)$, and $\mathcal{E}_\ell(t)\sim\llbracket t\rrbracket^\uparrow$ over (I,Σ_t^ℓ); both executions corestrict to the base alphabet, with $\mathcal{E}(\llbracket t\rrbracket)|_\Sigma\sim\llbracket t\rrbracket$ and $\mathcal{E}_\ell(t)|_\Sigma\sim\llbracket t\rrbracket$ over (I,Σ); and $\mathrm{Tr}^{1,*}(\llbracket t\rrbracket^1)\sim\llbracket t\rrbracket$;

(ii) $\llbracket s\rrbracket\sim\llbracket t\rrbracket$ in Part II if and only if $\mathcal{E}(\llbracket s\rrbracket)\sim\mathcal{E}(\llbracket t\rrbracket)$, if and only if $\mathcal{E}_\ell(s)|_\Sigma\sim\mathcal{E}_\ell(t)|_\Sigma$, if and only if $\mathrm{Tr}^{1,*}(\llbracket s\rrbracket^1)\sim\mathrm{Tr}^{1,*}(\llbracket t\rrbracket^1)$;

(iii) the equivalences of (ii) hold with $\sim$ replaced by $\simeq_{\mathrm{tr}}$, by $\simeq_{\mathrm{acc}}$, or by $\sqsubseteq$ read in either direction;

(iv) erasure is the identity on every reachable output of the executions in (i), so on their corestrictions $\sim_{\mathrm{erase}}$ coincides with $\sim$, and $\mathcal{E}(\llbracket s\rrbracket)\sim_{\mathrm{erase}}\mathcal{E}(\llbracket t\rrbracket)$ if and only if $\llbracket s\rrbracket\sim\llbracket t\rrbracket$.

Proof. (i) The whole-term claims are Theorem 9.3(i) and (ii) applied to the agent $\llbracket t\rrbracket$ over the singleton name set, and the trace claim is Theorem 9.4 applied to $\llbracket t\rrbracket$.

For the leafwise execution, two facts about Part II's operators are used, both read off their definitions. The first is naturality in the output alphabet: every operator's step map is T of a function that leaves the components' output symbols unchanged, injects them, or pairs them, and the constants skip, $\mathbf{0}$, fail_e and delay_d emit τ, which every pointed inclusion preserves; hence reading each constant A_j over $\Sigma_j\uplus M_N$ and forming the term gives the term formed from the A_j and then read over Σ_t^ℓ, that is, $t[A_j^\uparrow/A_j]=\llbracket t\rrbracket^\uparrow$. The second is Part II, Theorem 5.1: replacing the constants of a term by bisimilar constants of the same interface yields bisimilar agents, one operator at a time, for $;$, $\oplus_b$, $\oplus$, $\oplus_p$, $\otimes$, race, $\rhd$, $\mathsf{delay}_d\,;-$ and guarded μ. Theorem 9.3(i) gives $\mathcal{E}(A_j)\sim A_j^\uparrow$ over $(I_j,\Sigma_j\uplus M_N)$ for each constant, so the congruence theorem gives $\mathcal{E}_\ell(t)=t[\mathcal{E}(A_j)/A_j]\sim t[A_j^\uparrow/A_j]=\llbracket t\rrbracket^\uparrow$ over

(I, Σ_t^ℓ), which is the second claim. Marker-freeness is what makes the reading well typed: M_N is disjoint from every Σ_j, as Part IV, Definition 2.1 requires.

For the corestriction, let R be the bisimulation just obtained, between $\mathcal{E}_\ell(t)$ and $[\![t]\!]^\uparrow$. Every reachable state of $\mathcal{E}_\ell(t)$ is related by R to some state of $[\![t]\!]^\uparrow$: the initial states are, and if $(p, q) \in R$ and p' is a successor of p then, by the form of the three liftings (Part I, Lemma 2.7), some successor q' of q has $(p', q') \in R$ with the same output symbol. Since $[\![t]\!]^\uparrow$ emits only symbols of Σ_t, so does every reachable step of $\mathcal{E}_\ell(t)$, and the corestriction $\mathcal{E}_\ell(t)|_\Sigma$ exists. The restriction of R to reachable states is then a bisimulation between $\mathcal{E}_\ell(t)|_\Sigma$ and $[\![t]\!]$ over (I, Σ_t) by the argument of Theorem 9.3(ii): the steps read over Σ_t^ℓ are related by hypothesis, and Part I, Lemma 2.9 along the inclusion $\Sigma_t \hookrightarrow \Sigma_t^\ell$ on both sides gives relatedness under the lifting of $\mathrm{Eq}_{\Sigma_t} \times R$, with a witness supported on reachable pairs.

(ii) Bisimilarity between agents of a fixed type, interface and failure alphabet is an equivalence relation (Part I, Lemma 4.2). The whole-term executions of s and t are both agents over $(I, \Sigma \uplus M_N)$ with N a singleton, and the corestricted leafwise executions and the trivial traces are agents over (I, Σ), so each of the three comparisons is well typed. If $[\![s]\!] \sim [\![t]\!]$ then $\mathcal{E}([\![s]\!]) \sim [\![s]\!]^\uparrow \sim [\![t]\!]^\uparrow \sim \mathcal{E}([\![t]\!])$ by (i) and transitivity, using that reading two bisimilar agents over the same enlarged alphabet preserves bisimilarity (Part I, Lemma 2.8(viii) applied to the inclusion); conversely $\mathcal{E}([\![s]\!]) \sim \mathcal{E}([\![t]\!])$ gives $[\![s]\!]^\uparrow \sim \mathcal{E}([\![s]\!]) \sim \mathcal{E}([\![t]\!]) \sim [\![t]\!]^\uparrow$, and a bisimulation between $[\![s]\!]^\uparrow$ and $[\![t]\!]^\uparrow$ is a bisimulation between $[\![s]\!]$ and $[\![t]\!]$ by Part I, Lemma 2.9 along the inclusion, since neither emits a marker. The same two lines apply to the corestricted leafwise executions, with (i) supplying $\mathcal{E}_\ell(s)|_\Sigma \sim [\![s]\!]$ and $\mathcal{E}_\ell(t)|_\Sigma \sim [\![t]\!]$ over (I, Σ) directly, and to $\mathrm{Tr}^{1,*}(\cdot)$.

(iii) Each of $\simeq_{\mathrm{tr}}$ and $\simeq_{\mathrm{acc}}$ is an equivalence relation containing $\sim$ (Part I, Theorem 4.4 and Corollary 4.5), so the argument of (ii) applies verbatim with the containment supplying $\mathcal{E}([\![s]\!]) \simeq_{\mathrm{tr}} [\![s]\!]^\uparrow$ from $\mathcal{E}([\![s]\!]) \sim [\![s]\!]^\uparrow$, and with the observation that reading an agent over an enlarged alphabet changes neither its semantics nor its accepted traces beyond the inclusion. For refinement, Part I, Proposition 5.5 gives that $\sqsubseteq$ is a preorder and that bisimilar agents refine each other; hence $[\![s]\!] \sqsubseteq [\![t]\!]$ implies $\mathcal{E}([\![s]\!]) \sqsubseteq [\![s]\!]^\uparrow \sqsubseteq [\![t]\!]^\uparrow \sqsubseteq \mathcal{E}([\![t]\!])$, and the converse chain is the same with the roles exchanged.

(iv) By Theorem 9.3(ii) and the corestriction argument of (i), no reachable step of any execution in (i) emits a symbol of M_N, and by marker-freeness no reachable step of $[\![t]\!]$ emits a symbol of M; the trivial trace adds no output. Erasure therefore fixes every reachable output of every execution, so the agent obtained by post-composing step with $T(\mathrm{er} \times \mathrm{id})$, which is how Part I, Definition 5.1 defines $\sim_{\mathrm{erase}}$, agrees with the execution on its reachable part, and a bisimulation between two such agents restricts to one between the executions and conversely. Combining with (ii) gives the last clause. □

Corollary 9.6 (The equational theory of Part II is unchanged)**.** *Every law of Part II that is stated for all agents at $\sim$, at $\simeq_{\mathrm{acc}}$ or at $\sqsubseteq$ holds for the lockstep crash-free executions of marker-free terms, and every counterexample of Part II built from marker-free constants remains a counterexample there. In particular the category laws (Part II, Theorem 6.1), the guarded axioms (Theorem 8.1), the star-free axioms on the engaged fragment (Theorem 9.4), the monoidal laws (Theorem 10.1), the recovery laws (Proposition 11.1) and unique solutions (Theorem 7.1) hold of the executions, and the failures of accepted-trace congruence (Counterexample 9.2), of left distributivity (Counterexample 9.6), of the exchange law (Counterexample 10.3), of distribution of $;$ over $\otimes$ (Counterexample 10.5) and of distribution of $\triangleright$ over $;$ (Counterexample 11.2) persist.*

Proof. A law of Part II is a statement $[\![s]\!]\ R\ [\![t]\!]$ for two terms and one of the three relations, and a counterexample is a statement that the relation fails between two terms built from named constants. The constants of every Part II counterexample emit letters drawn from $\Sigma \setminus M$, so the terms are marker-free. Theorem 9.5(ii) and (iii) transport each statement to the executions and back. □

9.4 Necessity of the hypotheses

The three hypotheses of Theorem 9.5 are each used, and each has a witness in the Parts. The empty crash schedule is used in the second clause of the fail-stop lifting. With one crash, the execution differs from the term at bisimilarity: if $\varphi(1) \neq \emptyset$ and $\mathsf{init}(x)$ is running, the execution emits crash at the first tick and enters $\bot$, whose outcome is $\mathrm{inm}(\mathsf{crashed})$, while $[\![t]\!]$ emits a symbol of Σ and its outcome is a value, a reason of E, or running. The two are not bisimilar, because clause (B2) fails at the successor states. The marker-erased comparison is no better in general: Part IV, Counterexample 8.5 gives a supervised counter child whose erased word, with idle positions deleted, is $1\,2\,1\,2\,3$ against the crash-free $1\,2\,3$, so even $\sqsubseteq$ over $\sim_{\mathrm{erase}}$ fails without checkpointing. What survives a nonempty schedule is exactly what Part IV proves: refinement over erasure under checkpointing at every tick (Theorem 8.3) and under retry with a deduplication key (Theorem 9.2), never bisimilarity.

The lockstep scheduler is used in the identification of the scheduled family with $\otimes$. Replacing lockstep by the interleaving discipline of Part III, Definition 4.5 changes which terms are related. Part II, Counterexample 10.3 exhibits leaves a, b, c, d for which $(a\,;c)\otimes(b\,;d)$ and $(a\otimes b)\,;(c\otimes d)$ are related by $\sqsubseteq$ in neither direction; under the interleaving discipline, with the accepted-trace sets of Part III, Definition 4.8, the corresponding terms satisfy the exchange inclusion $(a \parallel b)\,;(c \parallel d) \sqsubseteq (a\,;c) \parallel (b\,;d)$ (Part III, Theorem 4.12(iii)). A pair of terms that Part II leaves incomparable becomes comparable, so the interleaving discipline is not conservative even at the coarsest relation. Part II, Remark 13.1 records the same point from the other side: the product is commutative in execution as well as in semantics, and the interleaving under a left-biased scheduler is not.

Marker-freeness is used twice. It makes the marker set of the execution interface disjoint from the term's alphabet, without which the fail-stop lifting is not defined (Part IV, Definition 2.1 requires M_N fresh). And it makes erasure the identity on the term. If a constant emitted a marker symbol $m \in M$, then the agent A and the agent A' obtained by replacing m by τ would satisfy $A \sim_{\mathrm{erase}} A'$ and $A \not\sim A'$, and the refinement results of Part IV, all stated over $\sim_{\mathrm{erase}}$, would identify two terms that Part II distinguishes, so clause (iv) of the theorem would fail.

9.5 Scope

The theorem says that placing a marker-free term under the trivial wire and the crash-free lockstep runtime changes nothing at bisimilarity, and therefore nothing at any coarser relation of the series. Its reading for design is the contrapositive. Whenever a runtime distinguishes two composition terms that Part II identifies, or identifies two that Part II distinguishes, the difference is attributable to a crash, to a scheduler that is not lockstep, or to a marker in the base alphabet, and the Parts supply the theorem that governs each case: Part IV, Theorem 8.3 for crashes under checkpointing, Part III, Theorem 4.10 for the interleaving scheduler, and Part I, Definition 5.1 for markers.

The theorem concerns executions of terms, not contexts, and it is not a full-abstraction result. Part IV's constructions Sup_σ, $\mathsf{Persist}_C$ and Replay are compared at refinement over marker erasure and are not claimed to respect bisimilarity (Part IV, Table 1 and Section 13), and Part V adds no operator (Part V, Section 2), so the theorem says nothing about a term placed inside a supervisor under a nonempty crash schedule, which is where Part IV's theorems take over. And the theorem establishes that the extension equates no new Part II terms and separates none, which leaves open whether $\sim$ or $\sqsubseteq$ is fully abstract for a contextual equivalence over the full signature (Section 12).

10 Properties of the composition

Each property below is a consequence of results from at least two Parts, stated with the hypotheses those results carry. None is proved in any single Part, because each needs an object one Part defines and a law another Part proves about it.

Proposition 10.1 (Retry needs a clock)**.** *Let $A : X \Rightarrow X$ have failure alphabet E and be able to fail at some initial state. Then the equation $\mathsf{X} = A \rhd \mathsf{X}$ has no solution given by Part II's recursion, while for every $d \geq 1$ the equation $\mathsf{X} = A \rhd (\mathsf{delay}_d \,;\, \mathsf{X})$ has a solution unique up to $\sim$. The crash-tolerant retry of Part IV inherits both halves: $\mathsf{retry}_{\infty,d}(A^{\perp})$ is well defined exactly when $d \geq 1$, and, for a keyed* Id *agent A whose crash-free run halts, against an environment replay-idempotent for the key, under an isolating failure model with crash points at tick boundaries and crashes delivered to the attempt, the run of $\mathsf{retry}_{n,d}(A^{\perp})$ under any schedule with at most n crashes and the crash-free run refine each other on the environment's effects after erasure and halt with the same value, while a counting environment shows duplicates.*

Proof. The first two sentences are Part II, Proposition 7.3, whose proof exhibits the infinite silent rewriting chain in the first case and applies the syntactic criterion of Part II, Lemma 4.23 and the unique-solution theorem, Part II, Theorem 7.1, in the second. Part IV, Definition 4.3 defines $\mathsf{retry}_{k+1,d}(A)$ as $\widehat{A} \rhd (\ulcorner p_X \urcorner \,;\, \mathsf{delay}_d \,;\, \mathsf{retry}_{k,d}(A))$ over the task lifting $\widehat{A}$, and Part IV, Proposition 4.4 states that the defining context of $\mathsf{retry}_{\infty,d}$ is guarded for $d \geq 1$ and unguarded for $d = 0$ when A can fail without a tick, citing the same two results of Part II. A crash under Part IV, Definition 3.1 makes the fail-stop lifting $A^{\perp}$ fail with reason $\mathsf{crashed}$, so a crash of the attempt is a failure that the recovery operator observes, and Part IV, Theorem 9.2 is stated for $\mathsf{retry}_{n,d}(A^{\perp})$ against a replay-idempotent environment with the stated hypotheses; Part IV, Counterexample 9.3 is the counting environment. □

The delay is not a tuning parameter. Without it the recursive equation that defines a retry loop has no meaning in the model (Part II), and with it the loop denotes a unique agent whose effects against a deduplicating receiver are the crash-free ones (Part IV). The two conditions are of different kinds, one on the term and one on the environment, and only the pair yields a retry that both denotes something and does what is intended.

Proposition 10.2 (Supervision is transparent only with persistence)**.** *Let $A_1, \ldots, A_n$ be* Id *agents that never enter a failed state and let γ be an execution model that is isolating and bounded for a supervisor's budget, with a replay-idempotent environment E and effect projection ν. With every child checkpointed at every tick, the erased observed closed system of $\mathsf{Sup}_{\sigma}(\vec{P})$ under any admissible crash schedule and the observed closed system of $\bigotimes_j A_j$ refine each other, for each of the three strategies, and the three strategies have the same erased stutter-closed traces. Without checkpointing, a counter child restarted at its initial state after one crash produces a trace that no crash-free run produces, and neither a different strategy, a larger budget nor a fair scheduler repairs it.*

Proof. The refinement in both directions is Part IV, Theorem 8.3; the identification of the strategies is Part IV, Corollary 8.4; the counter child is Part IV, Counterexample 8.5. The statement uses four objects from three Parts. Refinement and its stuttering closure are Part I, Definitions 5.3 and 5.4, and Part I, Proposition 5.5(i) is what allows idle positions to be inserted and deleted without leaving the closure. Erasure of markers is Part I, Definition 5.1. The crash-free reference system is the synchronous product $\bigotimes_j A_j$ of Part II, Definition 4.14, whose associativity by Part II, Theorem 10.1 is what makes the reference independent of the bracketing of the children. The supervisor, the checkpoint and the observed closed system are Part IV, Definitions 5.1, 6.1 and 2.4. □

The property attaches to the pair and not to either construct. Supervision alone changes the observable behaviour of a stateful child, because a restart re-enters the child at a state it has already left. Checkpointing at every tick makes the restart target the current state, and at that point the identity of the restarted set no longer matters, which is why the Erlang strategies become a choice of cost rather than of behaviour. Part IV, Remark 8.6 adds a third ingredient from Part III: the isolation hypothesis on the failure model can be traded for the channels of Part III, whose contents survive a crash where the closed-loop register of Part I does not.

Proposition 10.3 (Reproducible runs of logged model calls)**.** *Let* $\mathrm{ReAct}_1, \dots, \mathrm{ReAct}_n$ *be tool loops under the bounded-generation abstraction and let* $\ell_1, \dots, \ell_n$ *be logs of one run of each, compatible from the start contexts. Then:*

(a) each $\mathsf{Replay}(\ell_v)(\mathrm{ReAct}_v)$ *is an* Id *agent whose trajectory and output word over the logged ticks are the logged ones;*

(b) if the replayed agents, placed at the nodes of an acyclic graph under the channel discipline with unbounded channels, satisfy the remaining clauses of the Kahn fragment and the scheduler is fair, then, with the source-port histories fixed, every edge history is the same under every scheduler;

(c) if two replayed agents are typed by dual session types and linked by their only channels, one in each direction, under a fair scheduler, the pair never reaches a deadlock and halts exactly when the session ends;

(d) replacing the kernel π_θ *of one node by* $\pi_{\theta'}$ *of the same shape keeps (a) exactly when* ℓ_v *stays compatible, and a change that removes a logged resolution from the support is reported on the tick where it occurs.*

Proof. (a) Part V, Proposition 5.4 makes each loop an agent over $\mathcal{D}$ with a finite state space whose choice resolutions are the draws from π_θ; Part IV, Theorem 7.3(1) makes replay against a compatible log an Id agent with the logged trajectory and outputs, and Part V, Corollary 5.5 records the application. (b) Part IV, Definition 10.2 requires nodes over Id, which (a) supplies, together with quiescence, blocking reads, productivity, unbounded channels, acyclicity and the absence of races, retries and timeouts inside nodes; under those clauses and fairness, Part IV, Theorem 10.3 gives scheduler-independent edge histories. (c) is Part III, Theorem 7.6, whose typing clauses are properties of a single agent's port traces (Part III, Definition 7.3) and therefore apply to the replayed agents as to any others. (d) is Part IV, Theorem 7.3(2) and Corollary 7.4, cited for the loop in Part V, Corollary 5.5. □

This is the formal content of the phrase deterministic workflows over nondeterministic models. The model call is stochastic at the time it runs; its log makes it deterministic on replay; and determinism of the nodes is what the Kahn theorem needs to make the workflow schedule-independent. The proposition claims deadlock freedom only for a pair typed by dual session types, because Part III's theorem is binary and the multiparty case is open (Section 12). It claims nothing about the first run, whose nodes are over $\mathcal{D}$ and lie outside the Kahn fragment.

Proposition 10.4 (Human approval is a shield with an idling fallback)**.** *Over the extended action alphabet* $Q \times \mathrm{Act}$ *and against a sensor-aligned environment, let* H *be the oracle that approves* (q, a) *exactly when* $a \in \mathrm{Safe}(q)$*. Then the linked approval gate* $\mathrm{Gate}_H(A_\pi)$ *and the shielded agent* $A_\pi^{\Phi,\tau}$ *whose fallback is the idle output* τ*, on which a sensor-aligned environment does not move, have the same marker-erased, stutter-collapsed traces and refine each other, and they are not bisimilar. If*

$q_0 \in \Phi$ *then every environment state reachable in the closed loop of either lies in* Φ*, whether or not* Φ *is controllable. Every property of stutter-closed traces proved for one therefore holds for the other, and no property that counts ticks transfers.*

Proof. The mutual refinement, the failure of bisimilarity and the invariance of the shield without controllability are Part V, Proposition 9.7, whose proof first exhibits a bisimulation between the gate and the shield with a delay_1 inserted before the emitter and then removes the delay as a stutter. The invariance of the gate follows from the first of those two steps: Part II, Theorem 5.1 together with Part III, Lemma 3.3 lifts the bisimulation through the product and the trace that form the closed loop (Part III, Proposition 3.8), so the two closed loops reach the same environment states, and the invariance argument of Part V, Theorem 9.3 with the idling emitter applies to the delayed shield unchanged, because the inserted tick emits τ and a sensor-aligned environment does not move on τ (Part V, Definition 4.1). The objects come from four Parts: the invariant and its greatest fixed point are Part I, Definition 8.2 and Theorem 8.3, which Part V, Theorem 9.3 uses to place the reachable set inside Φ; the guarded choice between two emitters is Part II, Definition 4.10; the gate, its freezing of the wrapped agent while a question is pending, and the guarantee that no action precedes its approval are Part III, Definition 8.2 and Theorem 8.3(i); and the shield is Part V, Definition 9.2. The transfer clause is Part I, Definition 5.4: a property of stutter-closed observation sets is preserved by mutual refinement, and the two agents' observation sets coincide. The non-transfer clause is the failure of bisimilarity, since the gate emits four symbols per epoch and the shield three. □

The identification explains two facts that are invisible while the constructions live in separate formalisms. A gate needs no controllability hypothesis because refusing to act is always safe against an environment that does not move when the agent idles; and for the same reason a gate guarantees no progress, since a gate that refuses everything keeps the invariant by keeping the environment still (Part V, Remark 9.8). A shield with an action fallback needs controllability and in return keeps the environment moving. Both statements hold only against an idle-stationary environment, and against a plant that drifts while the operator deliberates the gate is not a shield.

Proposition 10.5 (Value composes along handoff)**.** *Let* M *be a finite Markov decision process with rational data and discount* $\gamma \in [0,1)$*, and let* $A_1 : 1 \Rightarrow Q$*,* $A_2 : Q \Rightarrow Q$ *and* $A_3 : Q \Rightarrow Y$ *be actuator-aligned agents over* $\mathcal{D}$ *with finite state spaces such that* A_1 *and* $A_1 ; A_2$ *halt at epoch boundaries, with halting epochs* τ_1 *and* τ_{12} *of finite expectation. Then for every* $q \in Q$*, with* k *ranging over epochs and* r_k *the reward read out in epoch* k*,*

$$\begin{aligned} V_\gamma\big(\mathrm{Run}(A_1 ; A_2 ; A_3, E_M)\big)(q) &= \mathbb{E}_q\Big[\sum_{k<\tau_1} \gamma^k r_k + \gamma^{\tau_1}\, V_\gamma\big(\mathrm{Run}(A_2 ; A_3, E_M)\big)(q_{\tau_1})\Big] \\ &= \mathbb{E}_q\Big[\sum_{k<\tau_{12}} \gamma^k r_k + \gamma^{\tau_{12}}\, V_\gamma\big(\mathrm{Run}(A_3, E_M)\big)(q_{\tau_{12}})\Big], \end{aligned}$$

the two right-hand sides being the decompositions of the two bracketings $A_1 ; (A_2 ; A_3)$ *and* $(A_1 ; A_2) ; A_3$*. When* A_1 *and* A_2 *are option terms that halt at epoch boundaries in the sense of Part V, Definition 7.2, each identity is the semi-Markov value equation of the options framework.*

Proof. Part V, Theorem 7.3 is stated for an agent of type $1 \Rightarrow Q$ that halts at epoch boundaries with a halting epoch of finite expectation, composed with a continuation of type $Q \Rightarrow Y$. Applied to A_1 with continuation $A_2 ; A_3$ it gives the first identity, and applied to $A_1 ; A_2$, which has type $1 \Rightarrow Q$, with continuation A_3 it gives the second, in each case using the strong Markov property of the epoch chain that Part V, Lemma 4.2(4) and Part I, Proposition 9.3 establish. The resumption at the

boundary is exact because of where the boundary falls: by Part V, Definition 7.2 the halting state is reached after the emitting tick of an epoch, so the closed-loop register holds the action just emitted together with the idle input, the environment consumes that action on the next tick and announces the resulting state, and the continuation's sensing leaf reads the announcement on the tick after, in the same phase as a fresh run of $\mathrm{Run}(A_2 \,;\, A_3, E_M)$ started at q_{τ_1} reads its initial announcement; the reward carried by that announcement is the reward of the last action of A_1, which the read-out indexes to that action's epoch and which therefore belongs to the truncated sum. The two left-hand sides are the values of the two bracketings, and Part II, Theorem 6.1 proves associativity of ; by a coalgebra isomorphism between them; the isomorphism acts on the agent component of the closed loop alone and fixes the environment and the register, so the two closed loops have the same law of reward read-outs and the same value. Option terms are guarded recursions (Part V, Definition 7.1), meaningful by Part II, Theorem 7.1, and for option terms that halt at epoch boundaries their sequential composition again halts at epoch boundaries, since the handoff between them consumes no tick. For option terms each identity is equation (8) of Section 3 of Sutton, Precup and Singh [33], as Part V, Remark 7.4 records. □

An option hierarchy is a term of the algebra rather than a separate framework: the option is Part II's guarded recursion, the hierarchy is Part II's sequential composition, and the value equation that the options literature proves separately is the statement that value respects the handoff at a stopping time. Associativity of the handoff is what makes the hierarchy's value independent of how the stages are grouped. The finite-expectation hypothesis is what makes the discounted continuation integrable and is not implied by an option's data, as Part V, Section 12 records.

11 Objections

Three objections have enough force to require an answer, and the series answers each with a result rather than with a disclaimer.

The first is that the series is Mealy coalgebras with renaming and contains no new mathematics. The answer is that the series is scoped as a unifying semantics with proved laws, not as new category theory, and that the specific results it establishes together are not found together elsewhere. The exchange law fails for the synchronous product in both directions even under stuttering refinement (Part II, Counterexample 10.3), while the interleaving built from channels satisfies the exchange inclusion strictly and only at accepted traces with idle ticks deleted (Part III, Theorem 4.12); the guarded Kleene axioms are sound at bisimilarity for effectful terminating agents at all three monads, with the divergence axioms on the right of a composition demoted to accepted traces (Part II, Theorem 8.1 and Proposition 8.2); supervision is transparent only with checkpointing at every tick (Part IV, Theorem 8.3 and Counterexample 8.5); replay is deterministic for all three monads once the choice component of Part I's factorisation is logged (Part IV, Theorem 7.3); and the delayed trace, not the trace, is the structure feedback carries (Part III, Theorem 3.5 and Counterexample 3.6). Each is stated with its relation and its hypotheses, and each is a place where the object of Part I meeting an operator of a later Part produces a result that neither the coalgebraic nor the algebraic literature states on its own.

The second is that a synchronous lockstep product misrepresents asynchronous language-model agents. The answer is that asynchrony is a property of the runtime and not of the object. Part III defines the interleaving $\|$ for $T = \mathcal{P}$ with a scheduler that ranges over every component at every tick (Part III, Definitions 4.3 and 4.5), and proves that in the channel fragment its accepted-trace set is the shuffle of the component sets (Part III, Theorem 4.10), so a statement about $\|$ is a statement about the union of all schedules. Part IV quantifies over fair schedulers and proves schedule

independence for the Kahn fragment (Part IV, Definition 3.3 and Theorem 10.3). The synchronous product remains the right primitive because feedback is defined on it (Part III, Definition 3.2) and the closed loop is its trace (Part III, Proposition 3.8); asynchrony is derived from it by channels and a scheduler rather than assumed at the leaves. Theorem 9.5 is the precise form of the separation: the lockstep discipline adds nothing to the composition algebra, and the interleaving discipline changes it in a way the Parts measure.

The third is that a language model is not a fixed function, since context limits, sampling and learning break the model. The answer has three parts. At inference time, under the bounded-generation abstraction, a model is a kernel from bounded contexts to finite-support rational distributions over bounded token sequences, and a tool loop over it is an agent over $\mathcal{D}$ whose recursion is guarded and bisimilar to the loop written by hand (Part V, Definitions 5.1 and 5.3, Proposition 5.4); sampling is the effect, the bounded context is the state. A serving stack whose output distribution depends on the batch is not a kernel on contexts, and the proposition does not apply to it (Part V, Remark 5.2). Learning, meaning a change of θ, is an operation on agents rather than a term of the algebra and is declared out of scope by Part V, Section 12, with the categorical cybernetics of Capucci, Gavranović, Hedges and Rischel [8] and of Ghani, Hedges, Winschel and Zahn [14] named as the setting in which it would be a morphism. The claim is only that inference-time agents are objects of the algebra, and the projection table records it as a one-way simulation.

12 Open problems

Five problems are open at the end of the series, and each is located by a negative result or a stated limitation of a Part.

A parallel operator that is a bifunctor for sequential composition with early start. The synchronous product waits for its slower factor (Part II, Counterexample 10.3), the interleaving halts when every component has halted and is not a bifunctor for ; with early start (Part III, Section 10), and the N-shaped workflow under the early-start discipline is not bisimilar to any $\otimes$ and ; term (Part IV, Counterexample 10.9). An operator that lets a halted component hand its value on while its neighbour is still running, and that still respects $\sim$, is not constructed anywhere in the series. Pomset semantics and monoidal streams are candidate settings, and the problem is to find the operator and the relation at which its laws hold.

Full abstraction. Bisimilarity is sound for every operator of Parts II and III, and refinement is the relation at which Part IV's results hold, but whether either is fully abstract for a contextual equivalence defined by observing halting values is open (Part I, Section 12), and so is completeness of the guarded axioms for bisimilarity of agent terms over a fixed set of constants (Part II, Section 13). Theorem 9.5 settles the easier question that the extension is conservative on Part II terms; it says nothing about the coarsest congruence.

Non-commutative effects. Every law of Part II's product uses commutativity of T, and the free-monoid writer monad breaks the commutativity law of $\otimes$ (Part II, Section 10, remark after Theorem 10.1). State and input-output are therefore modelled as agents and wiring rather than as effects. A premonoidal agent algebra in the sense of Power and Robinson [25], and the question of which laws of Parts II and III survive in it, are not developed.

Learning as an operation on agents. Every policy in Part V is fixed, and policy improvement is treated as an external map on the family $\{A_\pi\}$ (Part V, Corollary 6.5 and Section 12). The interaction of a θ-update with checkpointing and replay is a concrete instance of the general question: a log of a run under π_θ is compatible with $\pi_{\theta'}$ only under the support condition of Part IV, Theorem

7.3(2), so a learning step that shrinks the support invalidates the log. A setting in which the update is itself a morphism, of the kind the categorical cybernetics literature studies, is where the question belongs.

Multiparty sessions under crash and restart. Deadlock freedom is proved for two parties (Part III, Theorem 7.6) by an invariant that uses a single linear order of protocol actions, and the projection and well-formedness machinery of multiparty session types [18] is not developed. Part IV's supervisors restart components without regard to session state, and the gate does not roll back a refused action (Part III, Section 10). Whether an n-party protocol whose parties are supervised and checkpointed remains deadlock-free under crash and restart, and under which cut sets, is open, and it combines the two limitations most likely to bind in practice.

13 Limitations

The series claim governs what can be limited, and it is repeated here, as every Part repeats it, so that each restriction can be read against the sentence it restricts.

> Part I fixes the object: discrete time, an effect monad T drawn from the list Id, $\mathcal{P}$, $\mathcal{D}$, and agents as effectful Mealy coalgebras with typed termination and failure as a terminal outcome. Parts II to IV introduce the operators under additional assumptions stated in each Part and prove, for each operator, that it is well-defined on agents, which relation it respects, and which laws hold at that relation: Part II for ;, $\oplus$, $\otimes$, $\triangleright$ and μ; Part III for Tr, channels, $\|$ and Gate_H; Part IV for Sup, Persist and Replay; Part V introduces no operator and interprets policies, plans, options, shields and controllers as terms of Parts I to IV, proving properties of their closed loops. For each of sixteen fields the series gives an explicit translation of one named mechanism into a term of the algebra together with an adequacy result for that translation, which is a soundness theorem, a one-way simulation, or a proved limitation; no field is claimed to reduce to the algebra.

The synthesis adds Theorem 9.5, which holds for marker-free Part II terms under the lockstep scheduler with the empty crash schedule and at bisimilarity, and it adds nothing to the claim's list of operators or fields. The following restrictions bind on the series as a whole.

The object is restricted in four ways that no Part relaxes. Time is discrete and synchronous, so continuous time and independent local clocks are outside the series, and Part V's linear results concern sampled systems only. The effect monad is one of three commutative monads, so state and input-output are not effects and the premonoidal case is open. Nondeterminism and probability are never combined: no law ranges over a convex combination of $\mathcal{P}$ and $\mathcal{D}$, and Part V derives its stochastic shield from the nondeterministic one through a monad morphism precisely to avoid such a model. The semantics is finite-trace, so liveness is expressible only through fairness hypotheses on schedulers and outcome fairness on environments, both of which are assumptions rather than consequences.

The adequacy results are of three kinds, and the weakest kind is a one-way simulation. Five rows of Table 2 are of that kind, and for each the converse is either false or unexamined: actor creation lies outside the actor fragment, Linda's eval is not covered, a real supervisor distinguishes child specifications and shutdown protocols the model omits, at-least-once delivery with deduplication is not exactly-once delivery, and a language model whose sampling depends on batch composition is not a kernel on contexts.

The execution layer holds under four assumptions that a deployment may not satisfy. Crashes are fail-stop and land between ticks; a crash tick is isolated; supervised children are deterministic and do

not fail internally; and the environment is quiescent on idle output and on markers. Part IV, Remark 8.6 shows that dropping isolation breaks supervision refinement even with checkpointing at every tick, and Part IV, Section 13 records that supervision of nondeterministic children, checkpointing at asynchronous cuts and the relation between the channel and early-start workflow disciplines are all unaddressed.

The executable checks that accompany the series are finite-model checks. Each Part's appendix reports property runs over generated finite agents and exhaustive enumerations up to stated bounds, inside a fragment with finite state spaces, finite channel capacities at least the tick budget, bounded recursion and rational weights. They rule out the cheap ways of being wrong and establish nothing about the theorems, which rest on their proofs; a finite-capacity check never validates an unbounded-channel statement, and Theorem 9.5 has no check at all. The series has been reviewed only within its own process, and an external human review of the mathematics is outside this work.

14 Conclusion

One object carries the constructions of sixteen fields, and for each construction the series names the relation at which its laws hold and proves them there. The composition algebra is a category with guarded laws at bisimilarity and star-free laws at accepted traces; the wiring layer is a delayed trace whose channel fragment composes by shuffle and whose shared-object fragment is linearizable; the execution layer refines the crash-free run after erasure under checkpointing and deduplication; and the decision layer reads policies, options, beliefs, shields, plans and controllers as terms whose closed loops satisfy the textbook equations once the one-tick register is accounted for.

The wiring and execution layers are conservative over the composition algebra on marker-free terms under the lockstep scheduler with no crashes. Every departure from Part II's equational theory that a runtime exhibits is therefore attributable to a crash, to a scheduler that is not lockstep, or to a marker in the base alphabet, and the Parts supply the theorem that governs each case. What remains uncertain is not the object but its reach: an early-start parallel operator, full abstraction, non-commutative effects, learning as a morphism, and multiparty protocols under restart are the five places where the algebra stops, and each is marked by a proved negative result or a stated obstruction rather than by silence.

References

[1] Abadi, M., Lamport, L. (1991). The existence of refinement mappings. *Theoretical Computer Science* 82(2):253-284. DOI: 10.1016/0304-3975(91)90224-P.

[2] Agha, G. (1986). *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA.

[3] Agha, G., Mason, I., Smith, S., Talcott, C. (1997). A foundation for actor computation. *Journal of Functional Programming* 7(1):1-72. DOI: 10.1017/S095679689700261X.

[4] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U. (2018). Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence* 32(1). arXiv:1708.08611.

[5] Armstrong, J. (2003). Making reliable distributed systems in the presence of software errors. PhD thesis, KTH Royal Institute of Technology, Stockholm. The supervisor behaviour and its restart strategies are described on pages 118 to 123.

[6] Åström, K.J., Murray, R.M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

[7] Burckhardt, S., Gillum, C., Justo, D., Kallas, K., McMahon, C., Meiklejohn, C. (2021). Durable functions: semantics for stateful serverless. *Proceedings of the ACM on Programming Languages* 5(OOPSLA), article 133. DOI: 10.1145/3485510.

[8] Capucci, M., Gavranović, B., Hedges, J., Rischel, E.F. (2021). Towards foundations of categorical cybernetics. *Applied Category Theory 2021*, EPTCS 372. arXiv:2105.06332.

[9] Chandy, K.M., Lamport, L. (1985). Distributed snapshots: determining global states of distributed systems. *ACM Transactions on Computer Systems* 3(1):63-75. DOI: 10.1145/214451.214456.

[10] Cimatti, A., Pistore, M., Roveri, M., Traverso, P. (2003). Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1-2):35-84.

[11] Erman, L.D., Hayes-Roth, F., Lesser, V.R., Reddy, D.R. (1980). The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. *ACM Computing Surveys* 12(2):213-253.

[12] Fischer, M.J., Lynch, N.A., Paterson, M.S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM* 32(2):374-382. DOI: 10.1145/3149.214121.

[13] Gelernter, D. (1985). Generative communication in Linda. *ACM Transactions on Programming Languages and Systems* 7(1):80-112. DOI: 10.1145/2363.2433.

[14] Ghani, N., Hedges, J., Winschel, V., Zahn, P. (2018). Compositional game theory. *Proceedings of LICS 2018*, pp. 472-481. DOI: 10.1145/3209108.3209165. arXiv:1603.04641.

[15] Herlihy, M., Wing, J. (1990). Linearizability: a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems* 12(3):463-492.

[16] Hoare, C.A.R., Möller, B., Struth, G., Wehrman, I. (2011). Concurrent Kleene algebra and its foundations. *Journal of Logic and Algebraic Programming* 80(6):266-296. DOI: 10.1016/j.jlap.2011.04.005. The exchange law is the axiom relating the two compositions, Section 2.

[17] Honda, K., Vasconcelos, V., Kubo, M. (1998). Language primitives and type discipline for structured communication-based programming. *ESOP 1998*, LNCS 1381, pp. 122-138. DOI: 10.1007/BFb0053567.

[18] Honda, K., Yoshida, N., Carbone, M. (2016). Multiparty asynchronous session types. *Journal of the ACM* 63(1), article 9.

[19] Joyal, A., Street, R., Verity, D. (1996). Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society* 119(3):447-468. DOI: 10.1017/S0305004100074338.

[20] Kaelbling, L.P., Littman, M.L., Cassandra, A.R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99-134. DOI: 10.1016/S0004-3702(98)00023-X.

[21] Kahn, G. (1974). The semantics of a simple language for parallel programming. Information Processing 74, Proceedings of the IFIP Congress, pp. 471-475, North-Holland.

[22] Kozen, D. (1997). Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems* 19(3):427-443. DOI: 10.1145/256167.256195.

[23] Larsen, K.G., Skou, A. (1991). Bisimulation through probabilistic testing. *Information and Computation* 94(1):1-28.

[24] Lee, E.A., Parks, T.M. (1995). Dataflow process networks. *Proceedings of the IEEE* 83(5):773-801.

[25] Power, J., Robinson, E. (1997). Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science* 7(5):453-468.

[26] Puterman, M.L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley. Chapter 6 gives policy evaluation and policy iteration for discounted models.

[27] Russell, S., Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*, 4th edition. Pearson. Chapter 2, agent function and agent program.

[28] Rutten, J.J.M.M. (2000). Universal coalgebra: a theory of systems. *Theoretical Computer Science* 249(1):3-80. DOI: 10.1016/S0304-3975(00)00056-6.

[29] Sangiorgi, D. (2011). *Introduction to Bisimulation and Coinduction*. Cambridge University Press. The unique-solution argument of Part II, Theorem 7.1 uses the bisimulation up-to-bisimilarity technique presented there.

[30] Schlichting, R.D., Schneider, F.B. (1983). Fail-stop processors: an approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems* 1(3):222-238.

[31] Smolka, S., Foster, N., Hsu, J., Kappe, T., Kozen, D., Silva, A. (2020). Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time. *Proceedings of the ACM on Programming Languages* 4(POPL), article 61. DOI: 10.1145/3371129. arXiv:1907.05920. The axiom system is the one introduced in Section 2.

[32] Sprunger, D., Katsumata, S. (2019). Differentiable causal computations via delayed trace. *LICS 2019*. arXiv:1903.01093.

[33] Sutton, R.S., Precup, D., Singh, S. (1999). Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181-211. DOI: 10.1016/S0004-3702(99)00052-1. Equation (8) and Theorem 1 of Section 3 are the results cited.

[34] Valdes, J., Tarjan, R.E., Lawler, E.L. (1982). The recognition of series parallel digraphs. *SIAM Journal on Computing* 11(2):298-313.