Agents as Effectful Mealy Coalgebras

Part I of *An Algebra of Agents*

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

September 2026

Abstract

An agent, in the sense used by actor systems, workflow engines, control loops and language-model tool loops, is usually specified by a diagram and a runtime rather than by a mathematical object. This Part fixes the object. An agent is a coalgebra of the functor $F(S) = (Y + E + 1) \times (T(\Sigma \times S))^I$: at each tick it consumes one input symbol, emits one output symbol under an effect monad T drawn from the identity, finite nonempty powerset and finite-support rational distribution monads, and reports one of three outcomes, halted with a value, failed with a reason, or running. Non-running states are required to stutter, which makes halting and failure silent and free of ticks. On this object we define the relations that the rest of the series quotes and settle their order. Bisimilarity implies equality of decorated traces at all three monads, hence trace equivalence, accepted-trace equivalence and mutual refinement. The converse holds at the identity monad, where the trace map is the unique morphism into a final coalgebra of causal halting stream functions, and fails at the other two, with an explicit witness. Invariants are closed under intersection and union, and the greatest invariant inside a predicate is a greatest fixed point. Finite agents have computable minimal quotients, canonical once unreachable states are discarded. Closing an agent with an environment gives a Markov chain whose kernel is the product of the component kernels. Time is discrete and effects are commutative, which excludes state and input-output.

1 Introduction

Software called agentic is built from a small vocabulary: 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 human approval step, a policy that chooses among tools. Each of these has a mature theory somewhere in computer science, and the theories do not talk to each other. Actor systems supply isolation and asynchronous messages; process calculi supply composition and its equational laws; Erlang and OTP supply supervision; workflow systems supply dependency graphs; control theory supplies feedback; Markov decision processes supply policies. What is missing is not another framework but a single object that all of these can be statements about.

This series proposes one and works out its algebra. The series claim, quoted here verbatim and again in Section 12, is this.

> 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 Part proves the sentence about the object and its relations, adding three assumptions: time is discrete and synchronous, the effect monad is one of the three named commutative monads, and state spaces are finite wherever a decision procedure or a Markov kernel is asserted. It proves no law about any operator, because it defines no operator.

The object is a Mealy machine with three modifications, and each modification is forced by something the later Parts need. The first is an effect monad on the step function, so that a nondeterministic router and a sampling policy are the same kind of thing as a deterministic function, differing only in the instance of T. The second is typed termination: an agent starts at a value of type X and, if it halts, hands on a value of type Y, which is what makes a handoff composable and what makes the eventual category have objects at all. The third is failure as a terminal outcome rather than an effect. An agent halts with a value, fails with a reason drawn from a finite alphabet, or is still running; the outcome is observed between ticks, outside the monad. Failure could have been modelled by an exception monad, and an earlier version of this design did so. It is modelled as a third outcome because recovery then becomes an operator on agents that is available at every instance of T, rather than a construct that exists only when the effect monad happens to carry exceptions.

Outcomes are state predicates evaluated between ticks, so halting and the handoff that follows it consume no tick and cost nothing observable. Non-running states must stutter: a halted or failed state emits the idle output and stays put on every input. Imposing that as a condition on the coalgebra rather than as a reading of it is what makes the stuttering closure of Section 5 a congruence on observations and cuts the final coalgebra of Section 6 down to the behaviours a machine can realise.

An algebra of agents is only as good as its notion of equality, and four notions are needed rather than one, because different laws are true at different grains. Bisimilarity is the finest and is what the composition operators of Part II respect. Trace equivalence is coarser and is what a black-box tester can see. Accepted-trace equivalence forgets everything about runs that never halt and is the grain at which the laws about divergence hold. Stuttering-closed refinement is a preorder rather than an equivalence and is what a supervised or retried execution satisfies against its crash-free specification. We prove the inclusions among them once, here, so that every later Part can name one relation per law and point at this Part for what the name means.

Bisimilarity implies equality of decorated traces, hence trace equivalence, at all three monads (Theorem 4.4). At the identity monad the implication reverses, because the trace map is the unique morphism into a final coalgebra whose carrier is the set of causal halting stream functions (Theorem 6.2). At the powerset and distribution monads it does not reverse, and the witness is the classical two-branch example transported to states with outcomes (Theorem 4.6). Invariants are closed under intersection and union, and the greatest invariant inside a predicate is the greatest fixed point of the predecessor operator (Theorem 8.3). Refinement is a preorder that identifies agents differing only in idle ticks, and bisimilar agents refine each other, but mutual refinement

Table 1: The semantic commitments of the series. Every later Part cites this table rather than restating it, and no later Part may weaken a row.

Commitment	Content
Time	Time is discrete. One input symbol is consumed and one output symbol is emitted per tick.
Alphabets	The input alphabet I and the output alphabet Σ are pointed sets with idle symbols $\varepsilon \in I$ and $\tau \in \Sigma$.
Effects	T is one of the commutative monads Id, $\mathcal{P}$ and $\mathcal{D}$ on sets, each of which preserves weak pullbacks. State and input-output are not effects of T.
Values	An agent starts at a value of type X and hands on a value of type Y when it halts. Value types and per-tick alphabets are never conflated.
Outcomes	An agent halts with a value, fails with a reason from a finite alphabet E, or is running. The outcome is a function of the state, evaluated between ticks, and consumes no tick.
Stuttering	A state whose outcome is not running emits τ and stays put on every input.
Relations	Every claim of the series names exactly one of $\sim$, $\simeq_{\mathrm{tr}}$, $\simeq_{\mathrm{acc}}$, $\sqsubseteq$ and $\sim_{\mathrm{erase}}$, defined in this Part.

is strictly weaker than bisimilarity (Theorem 5.5). Every finite agent has a minimal quotient by bisimilarity, computable by signature refinement, and the reachable minimal representative is unique up to isomorphism (Theorem 7.2). The closed loop of an agent with an environment co-agent, at the distribution monad and with finite state spaces, is a Markov chain whose kernel is the product of the two component kernels (Theorem 9.3).

None of these is new mathematics taken one at a time. Bisimulation is due to Park and Milner [12, 9]; universal coalgebra and the coinduction principle to Rutten [15]; the final Mealy coalgebra of causal stream functions to Rutten again [16]; probabilistic bisimulation to Larsen and Skou [7]; its decision procedure to Baier, Engelen and Majster-Cederbaum [3]; relation lifting for a weak-pullback-preserving functor and the coalgebraic treatment of effects to Jacobs and to Hasuo, Jacobs and Sokolova [5, 4]; stuttering-insensitive refinement to Abadi and Lamport [1]. What is new is the specific object, the specific relation vocabulary, and the fact that the six results hold together for one signature that the rest of the series then uses without restatement. Section 11.2 says where each borrowed idea comes from and what had to change.

A general coalgebraic argument would need a hypothesis on T at several points. Each such lemma is instead proved for Id, $\mathcal{P}$ and $\mathcal{D}$ separately, in one or two lines, and Theorem 2.9 is the concrete form of weak-pullback preservation. A companion code layer runs finite models of six results and reports agreement, as Section A records; those runs are finite-model checks and are not part of any proof.

2 The agent object

2.1 Semantic commitments

Several of the commitments in Table 1 constrain what the definitions below are allowed to say rather than following from them, which is why they come first. Parts II to V use the table unchanged.

2.2 Interfaces and effects

Definition 2.1 (Interface)**.** An *interface* is a pair (I, Σ) of pointed sets: an input alphabet I with a distinguished *idle input* $\varepsilon \in I$, and an output alphabet Σ with a distinguished *idle output* $\tau \in \Sigma$. The *product* of (I_1, Σ_1) and (I_2, Σ_2) is $(I_1 \times I_2,\ \Sigma_1 \times \Sigma_2)$, pointed at $(\varepsilon, \varepsilon)$ and (τ, τ). The *trivial*

interface is $(1, 1)$, in which both alphabets are singletons.

Reading ε as "nothing arrived this tick" and τ as "nothing was said this tick" is accurate but not part of the mathematics: the two symbols are distinguished elements, and what makes them idle is the stuttering requirement of Theorem 2.10 together with the stuttering closure of Theorem 5.3.

Definition 2.2 (Effect monad)**.** An *effect monad* is one of the following three monads on the category of sets, written T with unit η and Kleisli extension written $t \gg\!= f$.

- $\mathrm{Id}(Z) = Z$, with $\eta(z) = z$ and $z \gg\!= f = f(z)$.
- $\mathcal{P}(Z) = \{U \subseteq Z : U \text{ finite and nonempty}\}$, with $\eta(z) = \{z\}$ and $U \gg\!= f = \bigcup_{z \in U} f(z)$.
- $\mathcal{D}(Z) = \{\mu : Z \to \mathbb{Q}_{\geq 0} \mid \mu \text{ has finite support and } \sum_z \mu(z) = 1\}$, with $\eta(z)$ the Dirac distribution at z and $(\mu \gg\!= f)(z') = \sum_z \mu(z) f(z)(z')$.

All three are commutative: writing $t \otimes t' \in T(Z \times Z')$ for the double strength, computing t before t' and computing t' before t give the same element.

The empty set is excluded from $\mathcal{P}$ and subdistributions are excluded from $\mathcal{D}$ on purpose. An agent that cannot do anything at a tick is not a modelling need: an agent that has nothing to do emits τ, an agent that has failed emits τ and reports its reason, and an agent that never halts is the divergent agent of Part II, which emits τ forever. Admitting the empty set would make the zero of choice and the divergent agent the same thing, and they are not.

Weights in $\mathcal{D}$ are rational so that the equality of two distributions is decidable and so that the finite models of Section A compute exactly. Nothing in the proofs uses rationality; every statement about $\mathcal{D}$ holds verbatim for real weights of finite support.

Remark 2.3 (Excluded effects)*.* Commutativity is a real restriction and is what rules out the state and input-output monads. Part II's synchronous product needs the two components of a tick to be independent of the order in which they are computed, which for a strong monad is exactly commutativity in the sense of Kock [6]. Effectful computation in the general, possibly non-commutative case is Moggi's setting [10], and the premonoidal structure that survives when commutativity is dropped is the subject of Power and Robinson [13]. The series does not develop that case; Section 12 records what would have to change.

2.3 Support, predicate lifting and relation lifting

Three constructions on T are used throughout: the support of an effect, the predicate lifting that says an effect stays inside a subset, and the relation lifting that compares two effects.

Definition 2.4 (Support and predicate lifting)**.** For $t \in T(Z)$ the *support* $\mathrm{supp}(t) \subseteq Z$ is $\{t\}$ for Id, the set t itself for $\mathcal{P}$, and $\{z : t(z) > 0\}$ for $\mathcal{D}$. For $U \subseteq Z$ write $T(Z)|_U$ for the image of $T(U)$ under T of the inclusion $U \hookrightarrow Z$.

Lemma 2.5 (Support calculus)**.** *For each of the three instances and all $t \in T(Z)$, $f : Z \to T(W)$, $g : Z \to W$ and $U, V \subseteq Z$:*

(i) $\mathrm{supp}(\eta(z)) = \{z\}$*;*

(ii) $\mathrm{supp}(t \gg\!= f) = \bigcup_{z \in \mathrm{supp}(t)} \mathrm{supp}(f(z))$*;*

(iii) $\mathrm{supp}(T(g)(t)) = g(\mathrm{supp}(t))$*;*

(iv) $t \in T(Z)|_U$ *if and only if* $\mathrm{supp}(t) \subseteq U$*;*

(v) $T(Z)|_{U \cap V} = T(Z)|_U \cap T(Z)|_V$*, and the same holds for arbitrary intersections.*

Proof. For Id every clause is immediate. For $\mathcal{P}$, (i) to (iii) are the definitions of unit, union and image, (iv) says a set is a subset of U exactly when it is one, and (v) is the corresponding statement for subsets. For $\mathcal{D}$, (i) is immediate; in (ii) a weight $\sum_z \mu(z) f(z)(w)$ is positive exactly when some summand is, and no cancellation is possible because all weights are non-negative; (iii) is the same argument for $\sum_{g(z)=w} \mu(z)$; (iv) holds because a distribution factors through U exactly when it gives $Z \setminus U$ weight zero; (v) follows from (iv). □

Clause (ii) of Theorem 2.5 is where non-negativity matters. A signed measure would allow a positive and a negative branch to cancel, and the invariant theory of Section 8 would fail.

Definition 2.6 (Relation lifting)**.** For a relation $R \subseteq Z \times Z'$, the *lifting* $\mathrm{Rel}(T)(R) \subseteq T(Z) \times T(Z')$ is

$$\mathrm{Rel}(T)(R) = \{(t, t') : \exists w \in T(R).\ T(\pi_1)(w) = t \text{ and } T(\pi_2)(w) = t'\},$$

where π_1, π_2 are the projections of R viewed as a set of pairs.

Lemma 2.7 (The three instances of the lifting)**.** $\mathrm{Rel}(\mathrm{Id})(R) = R$*. For* $\mathcal{P}$*,* $(U, V) \in \mathrm{Rel}(\mathcal{P})(R)$ *if and only if every* $u \in U$ *has some* $v \in V$ *with* $u\, R\, v$ *and every* $v \in V$ *has some* $u \in U$ *with* $u\, R\, v$*. For* $\mathcal{D}$*,* $(\mu, \nu) \in \mathrm{Rel}(\mathcal{D})(R)$ *if and only if there is a distribution* ω *on* $Z \times Z'$ *with support inside* R *whose marginals are* μ *and* ν*.*

Proof. The case of Id is the definition. For $\mathcal{D}$, an element $w \in \mathcal{D}(R)$ is exactly a distribution on $Z \times Z'$ with support inside R, and $\mathcal{D}(\pi_1)(w)$, $\mathcal{D}(\pi_2)(w)$ are its marginals. For $\mathcal{P}$, if $w \subseteq R$ is finite and nonempty with projections U and V then each $u \in U$ occurs in some pair of w, which provides its partner, and symmetrically. Conversely, given the two conditions, choose v_u for each $u \in U$ and u_v for each $v \in V$ and put $w = \{(u, v_u) : u \in U\} \cup \{(u_v, v) : v \in V\}$, which is finite, nonempty, contained in R, and projects onto U and V. □

The two named instances are the Egli-Milner lifting for $\mathcal{P}$ and the Larsen-Skou lifting for $\mathcal{D}$ [7]; the coupling formulation is the one used in the proofs below.

Lemma 2.8 (Lifting calculus)**.** *Let* T *be one of the three instances. For relations* $R \subseteq Z \times Z'$*,* $R' \subseteq Z' \times Z''$ *and* $Q \subseteq W \times W'$*:*

(i) $R \subseteq R'$ *implies* $\mathrm{Rel}(T)(R) \subseteq \mathrm{Rel}(T)(R')$*, and* $\mathrm{Rel}(T)(R^{\mathrm{op}}) = \mathrm{Rel}(T)(R)^{\mathrm{op}}$*;*

(ii) $\mathrm{Rel}(T)(\mathrm{Eq}_Z) = \mathrm{Eq}_{T(Z)}$*;*

(iii) $\mathrm{Rel}(T)(R)\,;\mathrm{Rel}(T)(R') \subseteq \mathrm{Rel}(T)(R\,;R')$*, where* ; *is relational composition;*

(iv) for a function $f : Z \to Z'$*,* $\mathrm{Rel}(T)(\mathrm{graph}(f)) = \mathrm{graph}(T(f))$*;*

(v) if $(t, t') \in \mathrm{Rel}(T)(R)$ *and* $f : Z \to T(W)$*,* $g : Z' \to T(W')$ *satisfy* $(f(z), g(z')) \in \mathrm{Rel}(T)(Q)$ *for all* $(z, z') \in R$*, then* $(t \gg\!= f,\ t' \gg\!= g) \in \mathrm{Rel}(T)(Q)$*;*

(vi) if $(t, t') \in \mathrm{Rel}(T)(R)$ *and* $f : Z \to W$*,* $g : Z' \to W$ *satisfy* $f(z) = g(z')$ *for all* $(z, z') \in R$*, then* $T(f)(t) = T(g)(t')$*;*

(vii) if $(t, t') \in \mathrm{Rel}(T)(R)$*,* $\mathrm{supp}(t) \subseteq U$ *and* $R \cap (U \times Z')$ *is the graph of a function* $f : U \to Z'$*, then* $t' = T(f)(t)$*;*

(viii) if $(t,t') \in \mathrm{Rel}(T)(R)$ *and* $f : Z \to W$, $g : Z' \to W'$ *satisfy* $(f(z), g(z')) \in Q$ *for all* $(z,z') \in R$, *then* $(T(f)(t), T(g)(t')) \in \mathrm{Rel}(T)(Q)$.

Proof. (i) Apply T to the inclusion $R \hookrightarrow R'$ to transport a witness; the statement about R^{op} holds because swapping the two projections is an isomorphism $R \cong R^{\mathrm{op}}$.

(ii) The diagonal Eq_Z is isomorphic to Z by $z \mapsto (z,z)$, and under this isomorphism both projections become the identity, so $T(\pi_1)(w) = T(\pi_2)(w)$ for every $w \in T(\mathrm{Eq}_Z)$, and $w = T(\delta)(t)$ witnesses (t,t).

(iii) For Id this is the definition of relational composition. For $\mathcal{P}$, let $(U,V) \in \mathrm{Rel}(\mathcal{P})(R)$ and $(V,W) \in \mathrm{Rel}(\mathcal{P})(R')$. Each $u \in U$ has $v \in V$ with $u\,R\,v$ and that v has $w \in W$ with $v\,R'\,w$, so $u\,(R;R')\,w$; the symmetric argument covers each $w \in W$. For $\mathcal{D}$, let ω_1 couple (μ,ν) along R and ω_2 couple (ν,ρ) along R', and set $\omega(z,z'') = \sum_{z' \in \mathrm{supp}(\nu)} \omega_1(z,z')\,\omega_2(z',z'')/\nu(z')$. Each summand is a non-negative rational; summing over z'' gives $\sum_{z'} \omega_1(z,z')\nu(z')/\nu(z') = \mu(z)$ and summing over z gives $\rho(z'')$, and the support of ω lies inside $R\,;R'$.

(iv) The graph of f is isomorphic to Z with π_1 the identity and $\pi_2 = f$, so T of it is $T(Z)$ with projections the identity and $T(f)$.

(v) If R is empty then $\mathrm{Rel}(T)(R)$ is empty and the statement is vacuous, so assume R and hence Q nonempty. Let $w \in T(R)$ witness (t,t') and choose for each (z,z') in the finite set $\mathrm{supp}(w)$ a witness $h(z,z') \in T(Q)$ for $(f(z),g(z'))$, extending h to the rest of R by a fixed element of $T(Q)$; only the values on $\mathrm{supp}(w)$ affect $w \gg\!= h$. Then $w \gg\!= h \in T(Q)$, and $T(\pi_1)(w \gg\!= h) = w \gg\!= (T(\pi_1) \circ h) = w \gg\!= (f \circ \pi_1) = T(\pi_1)(w) \gg\!= f = t \gg\!= f$, using naturality of $\gg\!=$ in its continuation and the monad law $T(g)(t) = t \gg\!= (\eta \circ g)$. The second projection is symmetric.

(vi) With w as above, $f \circ \pi_1$ and $g \circ \pi_2$ are equal as functions $R \to W$, so $T(f)(t) = T(f \circ \pi_1)(w) = T(g \circ \pi_2)(w) = T(g)(t')$.

(viii) The assignment $(z,z') \mapsto (f(z),g(z'))$ is a function $R \to Q$; applying T to it sends a witness for (t,t') to a witness for $(T(f)(t),T(g)(t'))$, because the projections commute with it.

(vii) Let $w \in T(R)$ witness the pair. By Theorem 2.5(iii), $\pi_1(\mathrm{supp}(w)) = \mathrm{supp}(t) \subseteq U$, so $\mathrm{supp}(w) \subseteq R \cap (U \times Z')$, which is the graph of f. Hence π_2 agrees with $f \circ \pi_1$ on $\mathrm{supp}(w)$, and $t' = T(\pi_2)(w) = T(f)(t)$. □

Clause (iii) is where weak-pullback preservation appears in the general theory [5]; the proof above is its concrete form at the three instances, and the gluing construction for $\mathcal{D}$ is the only place where a division is performed.

Lemma 2.9 (Reflection along a function)**.** *Let* $f : Z \to W$ *and* $g : Z' \to W'$ *be functions and* $Q \subseteq W \times W'$ *a relation. If* $(T(f)(t),T(g)(t')) \in \mathrm{Rel}(T)(Q)$ *then* $(t,t') \in \mathrm{Rel}(T)\big((f \times g)^{-1}(Q)\big)$.

Proof. For Id this is the definition of a preimage. For $\mathcal{P}$, let U,V be the two sets. Every $u \in U$ has $f(u) \in f(U)$, which by hypothesis has a partner in $g(V)$, of the form $g(v)$ with $(f(u),g(v)) \in Q$, so (u,v) lies in the preimage; the other direction is symmetric. For $\mathcal{D}$, let ω couple $(\mathcal{D}(f)\mu, \mathcal{D}(g)\nu)$ along Q and set

$$\omega'(z,z') \;=\; \omega\big(f(z),g(z')\big) \cdot \frac{\mu(z)}{(\mathcal{D}(f)\mu)(f(z))} \cdot \frac{\nu(z')}{(\mathcal{D}(g)\nu)(g(z'))},$$

which is well defined on the support of $\mu \times \nu$ because $(\mathcal{D}(f)\mu)(f(z)) = \mu(f^{-1}\{f(z)\}) \geq \mu(z) > 0$ there. Summing $\omega'(z,z')$ over z' gives $\mu(z)\,\omega(f(z),\cdot)$ summed and renormalised, which is $\mu(z)$, and symmetrically for the other marginal; the support of ω' lies inside $(f \times g)^{-1}(Q)$ because ω is supported in Q. □

Theorem 2.9 for $\mathcal{D}$ is the concrete content of the statement that the distribution monad preserves weak pullbacks. It is used once, in Theorem 4.3, and that use is what makes minimality of the quotient in Section 7 true.

2.4 Agents

Definition 2.10 (Agent)**.** Fix an interface (I, Σ), an effect monad T and a finite *failure alphabet* E, possibly empty. An *agent of sequential type* $X \Rightarrow Y$ is a tuple $A = (S, \mathsf{init}, \mathsf{step}, \mathsf{halt})$ 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*: if $\mathsf{halt}(s) \neq \mathrm{inr}(*)$ then $\mathsf{step}(s, i) = \eta(\tau, s)$ for every $i \in I$.

A state s is *halted with value* y when $\mathsf{halt}(s) = \mathrm{inl}(y)$, *failed with reason* e when $\mathsf{halt}(s) = \mathrm{inm}(e)$, and *running* when $\mathsf{halt}(s) = \mathrm{inr}(*)$. We write $\mathrm{Ag}(I, \Sigma)$ for the agents over a fixed interface and failure alphabet, and $A : X \Rightarrow Y$ to record the sequential type.

An agent is a pointed coalgebra. Writing $O = Y + E + 1$ for the outcome set and

$$F(S) \;=\; O \times \big(T(\Sigma \times S)\big)^I,$$

the pair $\langle \mathsf{halt}, \mathrm{curry}(\mathsf{step}) \rangle : S \to F(S)$ is an F-coalgebra, and $\mathsf{init} : X \to S$ is the point. The stuttering requirement is a condition on the coalgebra, not part of the functor; Section 6 shows that the condition cuts the final coalgebra down to a subcoalgebra rather than destroying finality.

The two typings in Theorem 2.10 are independent and are never conflated. The interface (I, Σ) says what crosses the boundary during a tick, and it is fixed for the whole of $\mathrm{Ag}(I, \Sigma)$; sequential composition, defined in Part II, never changes it. The sequential type $X \Rightarrow Y$ says what is handed in at the start and handed on at a halt, and it is what makes composition typed. An agent that consumes tokens and emits tool calls while running, and returns a document when it halts, has I the tokens, Σ the tool calls, X the prompt type and Y the document type.

Definition 2.11 (Transition notation)**.** Write $s \xrightarrow{i/\sigma} s'$ when $(\sigma, s') \in \mathrm{supp}(\mathsf{step}(s, i))$. At $T = \mathcal{D}$ write $s \xrightarrow{i/\sigma}_p s'$ for the same branch when its weight $\mathsf{step}(s, i)(\sigma, s')$ is p.

Remark 2.12 (The informal tuple)*.* The object corrects an informal one. A common informal presentation lists internal state, observations, actions, messages, an environment, a transition operator, a decision operator and an execution semantics. Here observations and incoming messages are the single alphabet I; actions and outgoing messages are the single alphabet Σ; the transition and decision operators are the two halves of the factorisation in Theorem 2.13; the environment is a second agent over the dual interface (Section 9); and the execution semantics is not part of the object at all, but the subject of Part IV. The agent-function and agent-program distinction of Russell and Norvig [14] is, in this vocabulary, the distinction between the semantics $[\![A]\!]$ of Theorem 3.2 and the coalgebra that computes it.

Definition 2.13 (Policy and update)**.** A *policy-update factorisation* of an agent A is a set K of *choices* together with functions $\pi : S \times I \to T(\Sigma \times K)$ and $\delta : S \times I \times \Sigma \times K \to S$ such that

$$\mathsf{step}(s, i) \;=\; \pi(s, i) \gg\!= \lambda(\sigma, k).\ \eta\big(\sigma,\ \delta(s, i, \sigma, k)\big).$$

Every agent has the trivial factorisation with $K = S$, $\pi = \mathsf{step}$ and δ the projection onto the state, so the definition is not a restriction. Its content is quantitative: a factorisation with a small

K isolates the effect into a small choice, and the update is then a deterministic function of the state, the input, the emitted symbol and that choice. Part IV's log records exactly the sequence of choices k, which is why replay is deterministic there and not here; Part V instantiates π by a policy class and δ by a belief or context update.

Remark 2.14 (A retry agent). Fix $I = \{\varepsilon, \mathsf{ok}, \mathsf{err}\}$, $\Sigma = \{\tau, \mathsf{req}\}$, $E = \{\mathsf{exhausted}\}$, $X = Y = 1$ and a budget $n \geq 1$. The agent R_n has states $s_0, \dots, s_{n-1}$, h and f with $\mathsf{init}(*) = s_0$, and, at $T = \mathrm{Id}$,

$$\mathsf{step}(s_k, \varepsilon) = \eta(\mathsf{req}, s_k), \quad \mathsf{step}(s_k, \mathsf{ok}) = \eta(\tau, h), \quad \mathsf{step}(s_k, \mathsf{err}) = \begin{cases} \eta(\mathsf{req}, s_{k+1}) & k+1 < n, \\ \eta(\tau, f) & k+1 = n, \end{cases}$$

with $\mathsf{halt}(s_k) = \mathrm{inr}(*)$, $\mathsf{halt}(h) = \mathrm{inl}(*)$, $\mathsf{halt}(f) = \mathrm{inm}(\mathsf{exhausted})$, and with h and f stuttering as the definition requires. The agent asks, waits, counts errors, and fails with a reason when the budget is gone. Section 10 follows this agent through the trace semantics, the invariant lattice, the quotient and the closed loop.

3 Trace semantics

The semantics of an agent is what an observer sees who feeds it an input word and watches the outputs and the outcome. Because outcomes are visible between ticks and not only at the end, we define a decorated run that records the outcome after every tick, and obtain the undecorated semantics, the accepted traces and the refinement observations as three images of it. The decoration costs nothing and saves three separate inductions later.

Definition 3.1 (Runs)**.** For $n \geq 0$ define $\mathsf{run}_n : S \times I^n \to T\big((\Sigma \times O)^n \times S\big)$, where $O = Y + E + 1$, by

$$\mathsf{run}_0(s, \langle\rangle) = \eta(\langle\rangle, s),$$

$$\mathsf{run}_{n+1}(s, i\,w) = \mathsf{step}(s, i) \gg\!= \lambda(\sigma, s').\ T\big(\lambda(v, s'').\ ((\sigma, \mathsf{halt}(s'))\,v,\ s'')\big)\big(\mathsf{run}_n(s', w)\big).$$

The decoration attached to tick k is the pair of the symbol emitted at that tick and the outcome of the state reached after it, which is the outcome an observer would read between tick k and tick $k+1$.

Definition 3.2 (Semantics)**.** The *decorated semantics* of A is $[\![A]\!]^+ : X \times I^* \to T\big((\Sigma \times O)^*\big)$, $[\![A]\!]^+(x, w) = T(\pi_1)\big(\mathsf{run}_{|w|}(\mathsf{init}(x), w)\big)$. The *semantics* of A is $[\![A]\!] : X \times I^* \to T\big(\Sigma^* \times O\big)$ obtained from it by keeping the output word and the last outcome,

$$[\![A]\!](x, w) \ = \ T\big(\lambda v.\ (\sigma_1 \cdots \sigma_n,\ o_n)\big)\big([\![A]\!]^+(x, w)\big), \qquad v = (\sigma_1, o_1) \cdots (\sigma_n, o_n),$$

with $[\![A]\!](x, \langle\rangle) = \eta(\langle\rangle, \mathsf{halt}(\mathsf{init}(x)))$.

Definition 3.3 (Trace equivalence)**.** Agents A and B of the same sequential type over the same interface and failure alphabet are *trace equivalent*, written $A \simeq_{\mathrm{tr}} B$, when $[\![A]\!] = [\![B]\!]$.

Lemma 3.4 (Splitting)**.** *For $w \in I^m$ and $w' \in I^n$,*

$$\mathsf{run}_{m+n}(s, ww') \ = \ \mathsf{run}_m(s, w) \gg\!= \lambda(v, s').\ T\big(\lambda(v', s'').\ (vv', s'')\big)\big(\mathsf{run}_n(s', w')\big).$$

Consequently the length-m prefix of $[\![A]\!]^+(x, ww')$ is $[\![A]\!]^+(x, w)$: writing take_m *for truncation of a word to its first m letters, $T(\mathrm{take}_m)\big([\![A]\!]^+(x, ww')\big) = [\![A]\!]^+(x, w)$.*

Proof. Induction on m. For $m = 0$ both sides are $\mathsf{run}_n(s, w')$ tagged with the empty prefix, using the left unit law. For the step, expand $\mathsf{run}_{m+n+1}(s, iww')$ by Theorem 3.1, apply the induction hypothesis inside the continuation, and reassociate with the monad associativity law; the map that prefixes $(\sigma, \mathsf{halt}(s'))$ commutes with the map that concatenates v and v'. The consequence follows by applying $T(\mathrm{take}_m \circ \pi_1)$ to both sides and using $\mathrm{take}_m(vv') = v$. □

Definition 3.5 (Accepted traces)**.** A pair (w, v) with $w \in I^n$ and $v = (\sigma_1, o_1) \cdots (\sigma_n, o_n)$ is *accepting with value* y when $o_n = \mathrm{inl}(y)$ and $o_k = \mathrm{inr}(*)$ for every $k < n$. For $T \in \{\mathrm{Id}, \mathcal{P}\}$ the *accepted-trace set* is

$$\mathrm{Acc}(A, x) \;=\; \{(w, \sigma_1 \cdots \sigma_n, y) : v \in \mathrm{supp}([\![A]\!]^+(x, w)) \text{ and } (w, v) \text{ is accepting with value } y\},$$

and for $T = \mathcal{D}$ the *accepted-trace weights* assign to each such triple the total weight of the accepting v that produce it. Agents are *accepted-trace equivalent*, written $A \simeq_{\mathrm{acc}} B$, when these agree for every x.

The convention that an accepted trace is recorded at the first tick whose outcome is a value, rather than at any later tick, is what makes $\simeq_{\mathrm{acc}}$ insensitive to how long a halted agent is left running. It also makes the accepted-trace set of an agent that never halts empty, which is the property Part II needs when it states the laws about divergence at this relation and not at bisimilarity.

Remark 3.6 (Why three relations and not one)*.* The undecorated semantics $[\![A]\!]$ is what a black-box observer sees who reads the outcome only at the end of the experiment. It does not in general determine $[\![A]\!]^+$: an agent that halts at tick 3 and then stutters and an agent that halts at tick 5 having emitted τ at ticks 4 and 5 can produce the same output word and the same final outcome. The accepted traces and the refinement observations of Section 5 are therefore defined from the decorated semantics, and the theorem of Section 4 is proved at that level so that all three consequences follow at once.

4 Bisimilarity

Definition 4.1 (Bisimulation)**.** Let A and B be agents of the same sequential type over the same interface and failure alphabet. A relation $R \subseteq S_A \times S_B$ is a *bisimulation* when

(B1) $(\mathsf{init}_A(x), \mathsf{init}_B(x)) \in R$ for every $x \in X$;

(B2) $\mathsf{halt}_A(s) = \mathsf{halt}_B(s')$ for every $(s, s') \in R$;

(B3) $(\mathsf{step}_A(s, i), \mathsf{step}_B(s', i)) \in \mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times R)$ for every $(s, s') \in R$ and every $i \in I$.

A and B are *bisimilar*, written $A \sim B$, when such an R exists. On a single agent, *state bisimilarity* $\sim_A$ is the largest relation on S_A satisfying (B2) and (B3).

Lemma 4.2 (Coinduction)**.** *Fix an agent A. The union of any family of relations satisfying (B2) and (B3) again satisfies them, so $\sim_A$ exists and is the union of all of them; $\sim_A$ is an equivalence relation; and any relation satisfying (B2) and (B3) is contained in $\sim_A$. For two agents A and B over the same interface and failure alphabet, write $A + B$ for the agent whose state space is the disjoint union of S_A and S_B, with* step *and* halt *inherited componentwise and with* init *that of A; then $A \sim B$ holds if and only if $\mathsf{init}_A(x) \sim_{A+B} \mathsf{init}_B(x)$ for every $x \in X$. Bisimilarity between agents is an equivalence relation on agents of a fixed sequential type, interface and failure alphabet.*

Proof. Closure under unions is Theorem 2.8(i), since enlarging R enlarges $\mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times R)$. Reflexivity of $\sim_A$ is Theorem 2.8(ii) applied to the diagonal, symmetry is the statement about R^{op} in (i), and transitivity is (iii): if R and R' satisfy (B2) and (B3) then so does $R\,;R'$, because $(\mathrm{Eq}_\Sigma \times R)\,;(\mathrm{Eq}_\Sigma \times R') = \mathrm{Eq}_\Sigma \times (R\,;R')$. For the disjoint union, a relation on $S_A + S_B$ satisfying (B2) and (B3) restricts and corestricts to relations between the components and back, since step_{A+B} and halt_{A+B} agree with those of the components; so a bisimulation between A and B is exactly a relation satisfying (B2) and (B3) on $A + B$ that relates the two initial states at every start value, and the two formulations agree. The last sentence adds (B1), which is preserved by the same three constructions. □

Lemma 4.3 (Homomorphisms)**.** *Let A and B be agents and $f : S_A \to S_B$ a function with $f(\mathsf{init}_A(x)) = \mathsf{init}_B(x)$, $\mathsf{halt}_B(f(s)) = \mathsf{halt}_A(s)$ and $\mathsf{step}_B(f(s), i) = T(\mathrm{id}_\Sigma \times f)(\mathsf{step}_A(s, i))$ for all s and i. Then $A \sim B$, and for all $s, s' \in S_A$, $s \sim_A s'$ if and only if $f(s) \sim_B f(s')$.*

Proof. The graph of f satisfies (B1) and (B2) by hypothesis and (B3) by Theorem 2.8(iv), since $\mathrm{Eq}_\Sigma \times \mathrm{graph}(f) = \mathrm{graph}(\mathrm{id}_\Sigma \times f)$; hence $A \sim B$ and $s \sim f(s)$ in the disjoint union, which gives the forward direction by Theorem 4.2. For the converse, let $R = \{(s, s') : f(s) \sim_B f(s')\}$. Condition (B2) holds because halt factors through f. For (B3), the hypothesis and $f(s) \sim_B f(s')$ give $\big(T(\mathrm{id} \times f)(\mathsf{step}_A(s, i)),\ T(\mathrm{id} \times f)(\mathsf{step}_A(s', i))\big) \in \mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times \sim_B)$, and Theorem 2.9 applied to $\mathrm{id}_\Sigma \times f$ on both sides yields $(\mathsf{step}_A(s, i), \mathsf{step}_A(s', i)) \in \mathrm{Rel}(T)\big((\mathrm{id} \times f \times \mathrm{id} \times f)^{-1}(\mathrm{Eq}_\Sigma \times \sim_B)\big) = \mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times R)$. So R satisfies (B2) and (B3) and is contained in $\sim_A$. □

Theorem 4.4 (Bisimilarity implies trace equivalence)**.** *Let T be any of* Id*, $\mathcal{P}$, $\mathcal{D}$ and let $A \sim B$. Then $[\![A]\!]^+ = [\![B]\!]^+$, and hence $A \simeq_{\mathrm{tr}} B$.*

Proof. Let R be a bisimulation. We show by induction on n that for every $(s, s') \in R$ and every $w \in I^n$,

$$\big(\mathsf{run}_n^A(s, w),\ \mathsf{run}_n^B(s', w)\big) \in \mathrm{Rel}(T)\big(\mathrm{Eq}_{(\Sigma \times O)^n} \times R\big).$$

For $n = 0$ both sides are units, and $(\eta(\langle\rangle, s), \eta(\langle\rangle, s'))$ is related because η of a related pair is a witness. For the step, write $w = i\,w'$ and let $f(\sigma, u) = T\big(\lambda(v, u'').((\sigma, \mathsf{halt}_A(u))\,v, u'')\big)\big(\mathsf{run}_n^A(u, w')\big)$ and let g be the same expression built from B, so that $\mathsf{run}_{n+1}^A(s, w) = \mathsf{step}_A(s, i) \gg\!= f$ and likewise for B. Take (σ, u) and (σ, u') with uRu'. Then $\mathsf{halt}_A(u) = \mathsf{halt}_B(u')$ by (B2), so the two maps prefix the same letter, and the two continuations are related by $\mathrm{Rel}(T)(\mathrm{Eq}_{(\Sigma \times O)^n} \times R)$ by the induction hypothesis; Theorem 2.8(viii) applied to those prefixing maps gives $(f(\sigma, u), g(\sigma, u')) \in \mathrm{Rel}(T)(\mathrm{Eq}_{(\Sigma \times O)^{n+1}} \times R)$. By (B3) the two effects $\mathsf{step}_A(s, i)$ and $\mathsf{step}_B(s', i)$ are related by $\mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times R)$, so Theorem 2.8(v) gives the claim for $n + 1$.

Now take $(s, s') = (\mathsf{init}_A(x), \mathsf{init}_B(x))$, which lies in R by (B1). The two projections π_1 out of $\mathrm{Eq}_{(\Sigma \times O)^n} \times R$ agree on related pairs, so Theorem 2.8(vi) gives $[\![A]\!]^+(x, w) = [\![B]\!]^+(x, w)$. The undecorated semantics is an image of the decorated one under a function that does not depend on the agent, so $A \simeq_{\mathrm{tr}} B$. □

Corollary 4.5. *$A \sim B$ implies $A \simeq_{\mathrm{acc}} B$.*

Proof. The accepted-trace set and weights of Theorem 3.5 are computed from $[\![A]\!]^+$ by a function that does not mention the agent, and Theorem 4.4 makes the two decorated semantics equal. □

Proposition 4.6 (Trace equivalence does not imply bisimilarity)**.** *At $T = \mathcal{P}$ and at $T = \mathcal{D}$ there are agents L and N with $L \simeq_{\mathrm{tr}} N$ and $L \not\sim N$.*

Table 2: The relations defined in this Part and the inclusions established here. Each later Part names exactly one relation per law.

Relation	Defined in	Content
$\sim$	Theorem 4.1	Bisimilarity by relation lifting of T, preserving the outcome and relating the initial states at every start value.
$\simeq_{\mathrm{tr}}$	Theorem 3.3	Equality of the semantics $[\![\cdot]\!]$, that is of output words paired with the final outcome, at every start value and input word.
$\simeq_{\mathrm{acc}}$	Theorem 3.5	Equality of the traces of runs that halt with a value, recorded at the first such tick.
$\sqsubseteq$	Theorem 5.4	Inclusion of stuttering-closed observation sets, and of their supports at $\mathcal{D}$.
$\sim_{\mathrm{erase}}$	Theorem 5.1	Bisimilarity after the reserved marker outputs are replaced by τ.

$\sim$ implies $\simeq_{\mathrm{tr}}$, $\simeq_{\mathrm{acc}}$ and refinement in both directions (Theorem 4.4, Theorem 4.5, Theorem 5.5).
The converse of the first holds at Id (Theorem 6.2) and fails at $\mathcal{P}$ and $\mathcal{D}$ (Theorem 4.6).

Proof. Take $I = \{\varepsilon\}$, $\Sigma = \{\tau, a, b, c\}$, $X = Y = 1$ and $E = \emptyset$. The agent L has states ℓ_0, ℓ_1, h with $\mathsf{step}(\ell_0, \varepsilon) = \eta(a, \ell_1)$, $\mathsf{halt}(h) = \mathrm{inl}(*)$ and $\mathsf{halt}(\ell_0) = \mathsf{halt}(\ell_1) = \mathrm{inr}(*)$; the agent N has states n_0, n_b, n_c, h with $\mathsf{halt}(h) = \mathrm{inl}(*)$ and the other three running. At $T = \mathcal{P}$ put

$$\mathsf{step}_L(\ell_1, \varepsilon) = \{(b, h), (c, h)\}, \qquad \mathsf{step}_N(n_0, \varepsilon) = \{(a, n_b), (a, n_c)\},$$

$$\mathsf{step}_N(n_b, \varepsilon) = \{(b, h)\}, \qquad \mathsf{step}_N(n_c, \varepsilon) = \{(c, h)\}.$$

At $T = \mathcal{D}$ replace each two-element set by the uniform distribution on it and each singleton by a Dirac distribution.

Both agents produce, on the input word of length one, the output a with outcome running, and on the input word of length two, the outputs ab and ac with outcome halted, with equal weights at $\mathcal{D}$; longer words extend both by stuttering at h. Since I is a singleton these are all the input words, so $L \simeq_{\mathrm{tr}} N$, and by Theorem 3.5 also $L \simeq_{\mathrm{acc}} N$.

Suppose R were a bisimulation. By (B1), $(\ell_0, n_0) \in R$. At $T = \mathcal{P}$, (B3) at ℓ_0 requires every element of $\{(a, n_b), (a, n_c)\}$ to have a partner in $\{(a, \ell_1)\}$, so $\ell_1 \, R \, n_b$ and $\ell_1 \, R \, n_c$. Then (B3) at the pair (ℓ_1, n_b) requires the element (c, h) of $\mathsf{step}_L(\ell_1, \varepsilon)$ to have a partner in $\{(b, h)\}$ with an equal first component, and $b \neq c$. At $T = \mathcal{D}$, a coupling of $\eta(a, \ell_1)$ with the uniform distribution on $\{(a, n_b), (a, n_c)\}$ must put all mass of its first marginal on (a, ℓ_1), so it is the product of that Dirac with the second distribution, forcing $\ell_1 \, R \, n_b$ and $\ell_1 \, R \, n_c$ again; a coupling of the uniform distribution on $\{(b, h), (c, h)\}$ with $\eta(b, h)$ must likewise put mass on the pair $((c, h), (b, h))$, whose first components differ. In both cases no bisimulation exists. □

The witness is the two-branch example of the process-calculus literature [9], transported to agents by making the halting state carry a value. Nothing about typed termination or failure repairs it: the two agents differ in when the choice is made, and only bisimilarity sees that.

5 Stuttering and refinement

Refinement compares what two agents can be seen to do when idle ticks are not counted. Two executions that differ only by inserted idle ticks are the same observation, which is the discipline that makes a supervised or replayed execution comparable with the crash-free one in Part IV.

Definition 5.1 (Markers and erasure)**.** A fixed subset $M \subseteq \Sigma \setminus \{\tau\}$ of the output alphabet is reserved for *markers*, the outputs by which Parts III and IV announce a crash, a restart, a checkpoint

or a replay. The *erasure* map $\mathrm{er} : \Sigma \to \Sigma$ sends every marker to τ and fixes everything else, and it is extended to words letterwise. Two agents are related by $\sim_{\mathrm{erase}}$ when the agents obtained by post-composing step with $T(\mathrm{er} \times \mathrm{id})$ are bisimilar. A Part that presents its output alphabet as a product $\Sigma \times M$ with idle symbol (τ, none) takes the reserved subset to be the symbols whose marker component is not none, and then er replaces that component by none; this agrees with the definition above exactly when a marker is emitted only on a tick whose Σ component is already τ, which is what a fail-stop crash rule guarantees. In this Part M is empty, so erasure is the identity and $\sim_{\mathrm{erase}}$ coincides with $\sim$.

Definition 5.2 (Observation policy)**.** The series observes agents under the following policy, fixed once here because Parts III and IV cite it and define no second version; the clauses about hiding, fairness and crash points are stated for their use and are not used in this Part. Markers are outputs, so an observer that has not erased them sees them. Ports consumed by the feedback and linking operators of Part III are removed from the interface and are not observable. A scheduler is *fair* when every component that is enabled infinitely often is scheduled infinitely often. Crashes, in the sense of Part IV, occur between ticks and never inside one.

Definition 5.3 (Idle positions and stuttering closure)**.** Let $w = i_1 \cdots i_n$ be an input word and let $v = (\sigma_1, o_1) \cdots (\sigma_n, o_n)$ be a decorated output word. Position k is *idle* when $i_k = \varepsilon$ and $\mathrm{er}(\sigma_k) = \tau$. The *collapse* $\mathsf{c}(w, v)$ is the word over $I \times \Sigma$ obtained by deleting every idle position and applying er to the remaining output letters. Two pairs are *stutter equivalent* when they have the same collapse, and for a set U of pairs the *stuttering closure* $\mathsf{st}(U)$ is the set of pairs stutter equivalent to a member of U.

Definition 5.4 (Observations and refinement)**.** The *observation set* of A at x is

$$\mathrm{Obs}(A, x) \;=\; \{(\mathsf{c}(w, v),\; o_{|w|}) : w \in I^*,\; v \in \mathrm{supp}([\![A]\!]^+(x, w))\},$$

with o_0 read as $\mathsf{halt}(\mathsf{init}(x))$. Agent B *refines* agent A, written $A \sqsubseteq B$, when $\mathrm{Obs}(A, x) \subseteq \mathrm{Obs}(B, x)$ for every $x \in X$.

Taking supports means that at $\mathcal{D}$ refinement compares which observations are possible and not with what probability, which is the reading fixed by the series claim ledger. The observation set records the outcome after every prefix, not only at the end, so a difference in when an agent halts is visible even when the output words agree.

Proposition 5.5 (Refinement is a preorder)**.** *For all three instances of T:*

(i) st *is a closure operator on sets of pairs and is idempotent, and* $\mathsf{st}(U) \subseteq \mathsf{st}(V)$ *if and only if the collapses of U are among the collapses of V;*

(ii) $\sqsubseteq$ *is reflexive and transitive;*

(iii) $A \sim B$ *implies* $A \sqsubseteq B$ *and* $B \sqsubseteq A$*;*

(iv) the converse of (iii) fails: at $\mathcal{P}$ and $\mathcal{D}$ there are agents that refine each other and are not bisimilar.

Proof. (i) Stutter equivalence is the kernel of the function c, hence an equivalence relation, and $\mathsf{st}(U)$ is the union of the classes meeting U; a union of classes is its own closure, which gives idempotence, and inclusion of unions of classes is inclusion of the sets of classes involved.

(ii) Inclusion of sets is reflexive and transitive.

(iii) By Theorem 4.4 the decorated semantics agree, so the supports agree, so the observation sets are equal.

(iv) The two agents of Theorem 4.6 have equal decorated semantics, hence equal observation sets, hence refine each other, and they are not bisimilar. □

Clause (iv) is the reason the series states congruence results at $\sim$ and refinement results at $\sqsubseteq$ and never mixes them: $\sqsubseteq$ is a preorder whose symmetric part is strictly coarser than bisimilarity.

Remark 5.6 (Relation to refinement mappings). Stuttering-closed inclusion is the discrete-tick form of the stuttering-insensitive refinement of Abadi and Lamport [1], whose completeness theorem produces a refinement mapping from a machine-closed specification with finite invisible nondeterminism, using history and prophecy variables. No auxiliary variables appear here, because the object already carries its own observation set and the comparison is made directly on observations. The price is that $\sqsubseteq$ says nothing about timing: an agent that answers in one tick and an agent that idles for ten ticks and then answers have the same observations. Part IV therefore states its timeout results as tick counts rather than as refinements.

6 Finality for deterministic agents

At $T = \mathrm{Id}$ the semantics of an agent is the unique map into a final coalgebra, and bisimilarity and trace equivalence coincide. This is the coalgebraic content of the classical statement that a Mealy machine denotes a causal stream function [16], extended to typed termination and failure and restricted to coalgebras that satisfy the stuttering requirement.

Throughout this section $T = \mathrm{Id}$ and $O = Y + E + 1$, so that $F(S) = O \times (\Sigma \times S)^I$ and an agent is a set S with maps $\mathsf{halt} : S \to O$ and $\mathsf{step} : S \times I \to \Sigma \times S$ satisfying the stuttering requirement.

Definition 6.1 (Behaviours)**.** Let $Z = O^{I^*} \times \Sigma^{I^+}$, whose elements are pairs (h, c) of a function h assigning an outcome to every input word and a function c assigning an output symbol to every nonempty input word. Give Z the coalgebra structure

$$\mathsf{halt}_Z(h,c) = h(\langle\rangle), \qquad \mathsf{step}_Z\big((h,c),i\big) = \big(c(i),\ (h_i, c_i)\big), \quad h_i(w) = h(iw),\ c_i(w) = c(iw).$$

Let $Z_{\mathrm{st}} \subseteq Z$ consist of those (h, c) such that $h(w) \neq \mathrm{inr}(*)$ implies $c(wi) = \tau$ and $h(wi) = h(w)$ for every $i \in I$.

Theorem 6.2 (Finality at the identity monad)**.** *Let $T = \mathrm{Id}$.*

(i) Z with the structure above is a final F-coalgebra, and the unique morphism from a coalgebra S sends s to the pair (h_s, c_s) where $h_s(w)$ is the outcome of the state reached from s on w and $c_s(wi)$ is the symbol emitted on the last tick.

(ii) Z_{st} is a subcoalgebra of Z, it satisfies the stuttering requirement, and it is final among F-coalgebras that satisfy the stuttering requirement.

(iii) For an agent A and $x \in X$, the pair $(h_{\mathsf{init}(x)}, c_{\mathsf{init}(x)})$ determines and is determined by $[\![A]\!](x, \cdot)$.

(iv) For agents A and B of the same type, $A \sim B$ if and only if $A \simeq_{\mathrm{tr}} B$.

Proof. (i) The map $(h, c) \mapsto (h(\langle\rangle),\ \lambda i.(c(i), (h_i, c_i)))$ is a bijection $Z \to F(Z)$: an element of $F(Z)$ is an outcome o together with, for each i, a symbol σ_i and a pair (h_i, c_i), and the unique preimage is $h(\langle\rangle) = o$, $h(iw) = h_i(w)$, $c(i) = \sigma_i$, $c(iw) = c_i(w)$. Given a coalgebra S, define h_s and c_s as

stated; the two equations for a coalgebra morphism are exactly the recursions $h_s(\langle\rangle) = \mathsf{halt}(s)$, $h_s(iw) = h_{s'}(w)$ and $c_s(i) = \sigma$, $c_s(iw) = c_{s'}(w)$ where $\mathsf{step}(s,i) = (\sigma, s')$, and these determine $h_s(w)$ and $c_s(w)$ by induction on $|w|$, which gives existence and uniqueness.

(ii) Let $(h,c) \in Z_{\mathrm{st}}$. If $h(\langle\rangle) = \mathrm{inr}(*)$ there is nothing to check for the stuttering requirement at that state. Otherwise the defining condition gives, by induction on $|w|$, that $h(w) = h(\langle\rangle)$ for every w and $c(w) = \tau$ for every nonempty w; hence $c(i) = \tau$ and $h_i = h$, $c_i = c$, so $\mathsf{step}_Z((h,c),i) = (\tau,(h,c))$, which is the stuttering requirement. The same computation shows step_Z maps Z_{st} into itself for running states as well, since the defining condition for (h_i, c_i) is a restriction of the one for (h,c). If S satisfies the stuttering requirement then $(h_s, c_s) \in Z_{\mathrm{st}}$ for every s: when $h_s(w) \neq \mathrm{inr}(*)$, the state reached on w is not running, so by the requirement it emits τ and stays, giving $c_s(wi) = \tau$ and $h_s(wi) = h_s(w)$. Finality is inherited from (i).

(iii) At $T = \mathrm{Id}$, $[\![A]\!](x,w) = (c_s(i_1)c_s(i_1 i_2)\cdots c_s(w),\ h_s(w))$ with $s = \mathsf{init}(x)$, which is a rewriting of the pair; conversely $h_s(w)$ is the second component of $[\![A]\!](x,w)$ and $c_s(w)$ is the last letter of the first component.

(iv) If $A \sim B$ then $A \simeq_{\mathrm{tr}} B$ by Theorem 4.4. Conversely, suppose $[\![A]\!] = [\![B]\!]$ and let β_A, β_B be the unique morphisms into Z. Put $R = \{(s,s') : \beta_A(s) = \beta_B(s')\}$. By (iii) and the hypothesis, $(\mathsf{init}_A(x), \mathsf{init}_B(x)) \in R$ for every x, so (B1) holds. For $(s,s') \in R$, equality of behaviours gives $\mathsf{halt}_A(s) = h_s(\langle\rangle) = h_{s'}(\langle\rangle) = \mathsf{halt}_B(s')$, which is (B2), and for each i the emitted symbols agree and the successors again have equal behaviours, which is (B3) because Rel(Id) is the identity on relations. □

Clause (iv) does not survive the addition of effects, by Theorem 4.6. What does survive at $\mathcal{P}$ and $\mathcal{D}$ is the forward implication, and that is the reason the series proves congruence at $\sim$: a law proved at bisimilarity holds at every coarser relation, and a law proved at trace equivalence does not lift.

Remark 6.3 (Effectful finality). For $T \in \{\mathcal{P}, \mathcal{D}\}$ a final coalgebra for $S \mapsto O \times (T(\Sigma \times S))^I$ still exists in the category of coalgebras, and the induced equality is bisimilarity rather than trace equivalence; trace semantics for effectful systems is obtained instead by moving to the Kleisli category of T, which is the framework of Hasuo, Jacobs and Sokolova [4]. This Part uses neither construction: Theorem 4.4 is proved directly from the lifting calculus, so that the argument is checkable instance by instance.

7 Quotients and minimization

Definition 7.1 (The quotient agent)**.** Let A be an agent and $q : S_A \to S_A/{\sim_A}$ the quotient map for state bisimilarity. Define $A/\sim$ by

$$S_{A/\sim} = S_A/{\sim_A}, \qquad \mathsf{init}_{A/\sim}(x) = q(\mathsf{init}_A(x)),$$

$$\mathsf{step}_{A/\sim}(q(s),i) = T(\mathrm{id}_\Sigma \times q)(\mathsf{step}_A(s,i)), \qquad \mathsf{halt}_{A/\sim}(q(s)) = \mathsf{halt}_A(s).$$

Theorem 7.2 (Minimal quotients)**.** *Let A be an agent over any of the three instances of T.*

(i) $A/\sim$ is well defined and satisfies the stuttering requirement, so it is an agent.

(ii) $A \sim A/\sim$.

(iii) No two distinct states of $A/\sim$ are bisimilar.

(iv) If S_A is finite, $\sim_A$ is computable by iterated signature refinement, in at most $|S_A|$ rounds.

(v) If A and B are bisimilar, reachable in the sense of Theorem 8.1, and have no two distinct bisimilar states, then there is a bijection $S_A \to S_B$ commuting with init, step *and* halt.

Proof. (i) The outcome is constant on bisimilarity classes by (B2), so $\mathsf{halt}_{A/\sim}$ is well defined. For $\mathsf{step}_{A/\sim}$, let $s \sim_A s'$. Then $(\mathsf{step}_A(s,i), \mathsf{step}_A(s',i)) \in \mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times \sim_A)$, and the two functions $\mathrm{id} \times q$ agree on pairs related by $\mathrm{Eq}_\Sigma \times \sim_A$, so Theorem 2.8(vi) gives $T(\mathrm{id} \times q)(\mathsf{step}_A(s,i)) = T(\mathrm{id} \times q)(\mathsf{step}_A(s',i))$. For the stuttering requirement, if $\mathsf{halt}_{A/\sim}(q(s))$ is not running then $\mathsf{halt}_A(s)$ is not running, so $\mathsf{step}_A(s,i) = \eta(\tau, s)$ and $\mathsf{step}_{A/\sim}(q(s), i) = T(\mathrm{id} \times q)(\eta(\tau, s)) = \eta(\tau, q(s))$.

(ii) By construction q satisfies the hypotheses of Theorem 4.3.

(iii) Suppose $q(s) \sim_{A/\sim} q(s')$. By the second part of Theorem 4.3 applied to q, this gives $s \sim_A s'$, hence $q(s) = q(s')$.

(iv) Define $\approx_0$ to be equality of outcomes and $\approx_{m+1}$ to be $\approx_m$ intersected with the relation that holds of s, s' when $(\mathsf{step}(s,i), \mathsf{step}(s',i)) \in \mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times \approx_m)$ for every i. Each $\approx_m$ is an equivalence relation by Theorem 2.8(i) to (iii), and the sequence is decreasing, so on a finite state space it stabilises after at most $|S_A|$ rounds at some $\approx_\infty$; the limit satisfies (B2) and (B3), so $\approx_\infty \subseteq \sim_A$, and an induction on m shows $\sim_A \subseteq \approx_m$ for every m, so the two coincide. Each round is computable because the membership test for $\mathrm{Rel}(T)(\mathrm{Eq}_\Sigma \times \approx_m)$ is a finite condition at each instance: set equality of the images at Id, the two Egli-Milner conditions at $\mathcal{P}$, and equality of the total weight assigned to each pair of a symbol and an $\approx_m$ class at $\mathcal{D}$, where the last is the standard characterisation of the lifting of an equivalence relation by a coupling. An algorithm for the case $T = \mathcal{D}$ with the stated bound $O(mn(\log m + \log n))$ on transitions and states is given by Baier, Engelen and Majster-Cederbaum [3]; the argument here claims decidability and the round bound, not that complexity.

(v) Let R be a bisimulation between A and B and let $\sim$ denote bisimilarity on the disjoint union, which contains R. Every reachable $s \in S_A$ has some $s' \in S_B$ with $s \sim s'$: this holds at the initial states by (B1), and if $s \sim s'$ and $s \xrightarrow{i/\sigma} u$ then (B3) provides u' with $s' \xrightarrow{i/\sigma} u'$ and $u \sim u'$, at each of the three instances by Theorem 2.7. Such an s' is unique because two candidates are bisimilar to each other and B has no two distinct bisimilar states; write $f(s)$ for it. Symmetrically define g, and note $g(f(s)) \sim s$, hence $g \circ f$ is the identity and likewise $f \circ g$, so f is a bijection. It commutes with init and halt by (B1) and (B2). For step, the relation $\sim$ restricted to $S_A \times S_B$ is the graph of f by uniqueness, and $\mathrm{supp}(\mathsf{step}_A(s,i))$ consists of reachable states, so Theorem 2.8(vii) applied to $\mathrm{Eq}_\Sigma \times \sim$ gives $\mathsf{step}_B(f(s), i) = T(\mathrm{id} \times f)(\mathsf{step}_A(s,i))$. □

Clause (v) is the sense in which the minimal quotient is canonical. Without reachability it is false: adding to a minimal agent an isolated state whose behaviour differs from every other state leaves it minimal and changes its state count.

8 Invariants

Definition 8.1 (Reachability)**.** $\mathrm{Reach}(A) \subseteq S_A$ is the least set containing $\mathsf{init}(x)$ for every $x \in X$ and closed under successors: if $s \in \mathrm{Reach}(A)$ and $(\sigma, s') \in \mathrm{supp}(\mathsf{step}(s,i))$ for some i, then $s' \in \mathrm{Reach}(A)$. An agent is *reachable* when $\mathrm{Reach}(A) = S_A$.

Definition 8.2 (Invariants)**.** A set $\Psi \subseteq S_A$ is *inductive* when $\mathsf{step}(s,i) \in T(\Sigma \times S_A)|_{\Sigma \times \Psi}$ for every $s \in \Psi$ and $i \in I$, equivalently when every successor of every state of Ψ lies in Ψ. It is an *invariant*, written $\mathrm{Inv}_\Psi(A)$, when it is inductive and contains $\mathsf{init}(x)$ for every x. The *predecessor operator* is $\mathsf{pre}(\Psi) = \{s \in S_A : \mathrm{supp}(\mathsf{step}(s,i)) \subseteq \Sigma \times \Psi \text{ for every } i \in I\}$.

The equivalence stated inside the definition is Theorem 2.5(iv). At $\mathcal{P}$ an inductive set is closed under every branch and at $\mathcal{D}$ under every branch of positive weight, so an invariant at $\mathcal{D}$ is an almost-sure statement about one step and, by induction, about every finite run.

Theorem 8.3 (Greatest invariants)**.** *Let A be an agent over any of the three instances and $\Phi \subseteq S_A$.*

(i) Inductive sets are closed under arbitrary unions and arbitrary intersections, and so are invariants.

(ii) $G_\Phi(\Psi) = \Phi \cap \mathsf{pre}(\Psi)$ is monotone on the lattice of subsets of S_A, and its greatest fixed point $\nu G_\Phi = \bigcup\{\Psi \subseteq \Phi : \Psi \subseteq \mathsf{pre}(\Psi)\}$ is the largest inductive subset of Φ.

(iii) An invariant contained in Φ exists if and only if $\mathsf{init}(X) \subseteq \nu G_\Phi$, and then νG_Φ is the greatest such invariant.

(iv) $\mathrm{Reach}(A)$ *is the least invariant, and an invariant contained in Φ exists if and only if* $\mathrm{Reach}(A) \subseteq \Phi$.

(v) If S_A is finite then $\nu G_\Phi = G_\Phi^k(\Phi)$ for some $k \le |\Phi|$.

Proof. (i) A state of a union of inductive sets lies in one of them, and its successors stay in that one, hence in the union. For intersections, a successor of a state of the intersection lies in every member by inductiveness of each, hence in the intersection; Theorem 2.5(v) is the same statement in the predicate-lifting formulation. Invariants add the condition $\mathsf{init}(X) \subseteq \Psi$, which is preserved by unions and by intersections of families of sets each containing $\mathsf{init}(X)$.

(ii) Monotonicity of pre is immediate from the definition, and the formula for the greatest fixed point is the Knaster-Tarski theorem on the complete lattice of subsets of S_A. A set Ψ satisfies $\Psi \subseteq \mathsf{pre}(\Psi)$ exactly when it is inductive, so the post-fixed points of G_Φ are the inductive subsets of Φ, and their union is the largest one; it is itself inductive by (i).

(iii) If $\Psi \subseteq \Phi$ is an invariant then Ψ is an inductive subset of Φ, so $\Psi \subseteq \nu G_\Phi$ and $\mathsf{init}(X) \subseteq \nu G_\Phi$. Conversely if $\mathsf{init}(X) \subseteq \nu G_\Phi$ then νG_Φ is itself an invariant, and it contains every other invariant inside Φ by the previous sentence.

(iv) $\mathrm{Reach}(A)$ is inductive and contains $\mathsf{init}(X)$ by construction, and it is contained in every invariant, by induction along the path witnessing reachability. If an invariant $\Psi \subseteq \Phi$ exists then $\mathrm{Reach}(A) \subseteq \Psi \subseteq \Phi$; conversely if $\mathrm{Reach}(A) \subseteq \Phi$ then $\mathrm{Reach}(A)$ is an invariant inside Φ.

(v) The chain $\Phi \supseteq G_\Phi(\Phi) \supseteq G_\Phi^2(\Phi) \supseteq \cdots$ is decreasing, and every strict step removes at least one state, so it stabilises after at most $|\Phi|$ steps at a fixed point that contains every post-fixed point below Φ. □

Clause (iv) is the practical statement: a safety property either holds of every reachable state, in which case the greatest fixed point computes a certificate for it, or the gfp iteration deletes an initial state and there is no invariant to be found. Section 10 shows both outcomes on the retry agent.

Remark 8.4 (Invariants and the rest of the series). Part IV's restart budget is an invariant of a supervisor state in the sense of Theorem 8.2, and Part V's shields preserve an invariant of the closed loop under a controllable-predecessor hypothesis. Both cite this section for what an invariant is and for the fixed-point characterisation, and neither redefines it.

9 Environments and the closed loop

An agent is not a closed system: it consumes inputs it does not produce. Closing it requires a second agent over the dual interface and a discipline for who speaks when.

Definition 9.1 (Environment co-agent)**.** An *environment* for $\mathrm{Ag}(I, \Sigma)$ is an agent over the dual interface (Σ, I), that is a tuple $E = (S_E, \mathsf{init}_E, \mathsf{step}_E, \mathsf{halt}_E)$ with $\mathsf{step}_E : S_E \times \Sigma \to T(I \times S_E)$ and $\mathsf{halt}_E : S_E \to Y_E + E + 1$ over the same failure alphabet, subject to the same stuttering requirement. The letter E has two roles fixed by the notation of the series, the failure alphabet inside a sum of outcomes and the environment as an agent, and the position distinguishes them.

Definition 9.2 (The closed loop)**.** Let $A : X \Rightarrow Y$ over (I, Σ) and let $E : X_E \Rightarrow Y_E$ be an environment. The *closed loop* $\mathrm{Run}(A, E)$ is the agent of type $X \times X_E \Rightarrow Y \times Y_E$ over the trivial interface with

$$\hat{S} = S_A \times S_E \times I \times \Sigma, \qquad \mathsf{init}(x, x_E) = (\mathsf{init}_A(x),\ \mathsf{init}_E(x_E),\ \varepsilon,\ \tau),$$

with $\mathsf{halt}(s, e, i, \sigma)$ equal to $\mathrm{inl}(y, y_E)$ when both components have halted with those values, to inm of the failure reason of the agent when A has failed and of the environment when A has not failed and E has, and to $\mathrm{inr}(*)$ otherwise, and with

$$\mathsf{step}\big((s, e, i, \sigma),\ *\big) \ = \ \eta\big(\tau,\ (s, e, i, \sigma)\big) \quad \text{when } \mathsf{halt}(s, e, i, \sigma) \neq \mathrm{inr}(*),$$

and at a running state

$$\mathsf{step}\big((s, e, i, \sigma),\ *\big) \ = \ \mathsf{step}_A(s, i) \gg\!= \lambda(\sigma', s').\ \mathsf{step}_E(e, \sigma) \gg\!= \lambda(i', e').\ \eta\big(\tau,\ (s', e', i', \sigma')\big).$$

The first clause of step is forced by the stuttering requirement of Theorem 2.10 and is not a modelling choice. Without it the definition would not produce an agent: when both components have halted they emit τ and ε and stay put, so the second clause would send the register to (ε, τ) and move the closed state, which a non-running state may not do. The guard costs nothing observable. Once the outcome is decided it stays decided, since a component that has halted or failed never moves again, and every output over the trivial interface is τ; so the relation that pairs a guarded state with every state the unguarded formula reaches from it satisfies (B2) and (B3) of Theorem 4.1, and the guarded and unguarded systems are bisimilar.

The register (i, σ) carries the one-tick delay in both directions: what the environment emitted at tick t is what the agent consumes at tick $t + 1$, and symmetrically. Composition in lockstep with a one-tick delay is a choice, and it is the choice Part III justifies by proving that $\mathrm{Run}(A, E)$ is the delayed feedback trace of the synchronous product; that proof is not available here because neither the product nor the trace is defined in this Part.

Proposition 9.3 (The closed loop is a Markov chain)**.** *Let $T = \mathcal{D}$ and let S_A, S_E, I and Σ be finite. Then the state sequence of* $\mathrm{Run}(A, E)$ *is a time-homogeneous Markov chain on $\hat{S}$. At a running state the one-step kernel is the product of the two component kernels,*

$$K\big((s, e, i, \sigma),\ (s', e', i', \sigma')\big) \ = \ \mathsf{step}_A(s, i)(\sigma', s') \cdot \mathsf{step}_E(e, \sigma)(i', e'),$$

and at a state whose outcome is decided it is the Dirac distribution at that state. Every row sums to one, and the kernel is unchanged if the environment is bound before the agent.

Proof. At a state whose outcome is decided the first clause of Theorem 9.2 gives the Dirac distribution, whose single row sums to one. At a running state the two binds of the second clause multiply the weights of the two independent choices and the final unit contributes a Dirac distribution, so the weight of the successor (s', e', i', σ') is the displayed product. That weight depends only on the current state, which is the Markov property, and not on the tick, which is homogeneity. Summing over the successor,

$$\sum_{s', e', i', \sigma'} K = \Big(\sum_{\sigma', s'} \mathsf{step}_A(s, i)(\sigma', s')\Big)\Big(\sum_{i', e'} \mathsf{step}_E(e, \sigma)(i', e')\Big) = 1 \cdot 1,$$

a finite rearrangement of a finite sum of non-negative terms. Binding the environment first produces the same product because $\mathcal{D}$ is commutative. □

Corollary 9.4. *Under the hypotheses of Theorem 9.3, the distribution of the state after n ticks is $\delta_{\mathsf{init}(x,x_E)}K^n$, and the probability that* $\mathrm{Run}(A,E)$ *has halted with value (y,y_E) by tick n is the total mass that $\delta_{\mathsf{init}(x,x_E)}K^n$ assigns to the states whose outcome is* $\mathrm{inl}(y,y_E)$.

Remark 9.5 (The register is not redundant). Projecting the chain onto $S_A \times S_E$ destroys the Markov property. Take $I = \{\varepsilon, a, b\}$, $\Sigma = \{\tau, x, y\}$, an agent with $\mathsf{step}(s,\varepsilon) = \frac{1}{2}\eta(x,s) + \frac{1}{2}\eta(y,s)$, $\mathsf{step}(s,a) = \eta(\tau,u)$ and $\mathsf{step}(s,b) = \eta(\tau,v)$ with $u \neq v$ and distinguishable, and an environment with one state e answering a to x, b to y and ε to τ. The projected process is at (s,e) after zero ticks and after one tick. After zero ticks the register is (ε,τ) and the next projected state is (s,e) with probability one; after one tick the register holds a fair choice between $(a,\cdot)$ and $(b,\cdot)$ and the next projected state is (u,e) or (v,e) with probability one half each. Two different conditional distributions from the same projected state make the projection non-Markov, so the register belongs in the state space.

Remark 9.6 (Other instances). At $T = \mathrm{Id}$ the closed loop is a deterministic dynamical system on $\hat{S}$ and the corollary reads as an orbit computation. At $T = \mathcal{P}$ the reachable set is the least fixed point of the successor operator of Theorem 8.1 applied to $\mathrm{Run}(A,E)$, and the safety questions of Section 8 are the corresponding greatest fixed points. Neither case needs Theorem 9.3, which is stated at $\mathcal{D}$ because that is where the product kernel has content.

10 A worked example

We follow the retry agent R_n of Theorem 2.14 through the constructions, with $n = 3$.

10.1 Traces

On the input word $\varepsilon\,\mathsf{err}\,\mathsf{ok}$ the decorated semantics is the single word

$$(\mathsf{req},\mathrm{inr}(*))\,(\mathsf{req},\mathrm{inr}(*))\,(\tau,\mathrm{inl}(*)),$$

so $[\![R_3]\!]$ at that word is $(\mathsf{req}\,\mathsf{req}\,\tau,\ \mathrm{inl}(*))$, and the triple $(\varepsilon\,\mathsf{err}\,\mathsf{ok},\ \mathsf{req}\,\mathsf{req}\,\tau,\ *)$ is an accepted trace. On $\mathsf{err}\,\mathsf{err}\,\mathsf{err}$ the outcome after three ticks is $\mathrm{inm}(\mathsf{exhausted})$ and the accepted-trace set contains nothing with that input word. The collapse of the first pair deletes no position, because the input letters err and ok are not idle and the first position emits req; the pair $\varepsilon\,\varepsilon\,\mathsf{ok}$ with outputs $\mathsf{req}\,\mathsf{req}\,\tau$ also has no idle position, since $\mathsf{req} \neq \tau$. An agent that waited silently, emitting τ rather than req while idle, would have those positions deleted, and would be refinement-equivalent to the variant that waits for a different number of ticks.

10.2 Invariants

Let $\Phi = S\setminus\{f\}$, the property that the budget is never exhausted. The gfp iteration of Theorem 8.3(v) computes $G_\Phi(\Phi) = \Phi \setminus \{s_2\}$, since s_2 has the successor f on input err; then $\Phi \setminus \{s_1, s_2\}$, then $\Phi \setminus \{s_0, s_1, s_2\} = \{h\}$, which is a fixed point. The initial state s_0 is not in it, so by clause (iii) there is no invariant inside Φ, which is the correct answer: three consecutive errors reach f. The set $\Psi = \{s_2, h, f\}$ shows that clauses (ii) and (iii) are different statements: Ψ is inductive, since every successor of s_2 lies in $\{s_2, h, f\}$, and it is therefore the greatest inductive subset of itself, but it is not an invariant because it omits s_0. The least invariant is $\mathrm{Reach}(R_3) = S$, as clause (iv) requires.

10.3 The quotient

No two states of R_3 are bisimilar. The outcomes separate h, f and the running states; among the running states, s_2 reaches f in one tick on err while s_0 and s_1 do not, and s_1 reaches a state that does while s_0 does not. The signature refinement of Theorem 7.2(iv) therefore stabilises after three rounds with the discrete partition, and $R_3/\!\sim$ is R_3. Increasing the budget to n gives $n+2$ states, all inequivalent, so bisimilarity does not compress a counter: the budget is observable.

10.4 The closed loop

Let $T = \mathcal{D}$ and let the environment E_p have one state e with $\mathsf{step}_E(e, \mathsf{req}) = p\,\eta(\mathsf{ok}, e) + (1-p)\,\eta(\mathsf{err}, e)$ and $\mathsf{step}_E(e, \tau) = \eta(\varepsilon, e)$, for a rational $p \in (0,1)$. In $\mathrm{Run}(R_3, E_p)$ the register starts at (ε, τ), so the first tick has the agent emit req while the environment answers the idle output with ε; the second tick has the agent emit req again while the environment answers the first req; from the third tick on, the agent consumes one independent answer per tick. The agent therefore fails exactly when the first three answers are err, which happens with probability $(1-p)^3$, and at $p = 1/2$ the closed loop fails with probability $1/8$ and halts with value $*$ with probability $7/8$. The same number is what Theorem 9.3 gives by summing the mass that $\delta_{\mathsf{init}} K^n$ puts on the failed states as n grows, and the kernel used in that computation is the product of the two component kernels on the state space $S_{R_3} \times \{e\} \times I \times \Sigma$.

Three of the constructions have visible content on an agent this small. Bisimilarity distinguishes states by their remaining budget, so the quotient is trivial and the retry counter is not an artefact of the presentation. The invariant lattice answers a safety question with a certificate or with a proof that none exists, and here it correctly reports that failure is reachable. The closed loop turns a qualitative failure mode into a number, and the number depends on the environment, not on the agent alone.

11 Discussion

11.1 Encoded mechanisms

Two of the sixteen mechanism translations that the series is accountable for are settled by the results above, and a third is half settled.

State machines contribute explicit labelled transitions. The translation is the step function of Theorem 2.10, and the adequacy result is soundness in the strongest available form at the deterministic instance: by Theorem 6.2 the agent's semantics is the unique morphism into a final coalgebra whose carrier is the set of causal halting stream functions, so nothing about a deterministic machine is lost or added by the translation. At the effectful instances the translation remains sound for bisimilarity but no longer for traces, which is the content of Theorem 4.6.

Formal methods contribute invariants and refinement. The translation is Theorem 8.2 and Theorem 5.4, and the adequacy result is Theorem 8.3: the invariants of an agent form a complete lattice under intersection and union, the greatest invariant inside a predicate is a greatest fixed point, and a certificate exists exactly when the reachable set satisfies the predicate. Refinement is the discrete-tick, stuttering-closed form of the classical notion, and Theorem 5.5 fixes its order-theoretic status.

Control theory contributes feedback, and this Part supplies half of the translation: $\mathrm{Run}(A, E)$ of Theorem 9.2 is the closed loop with a one-tick delay, and Theorem 9.3 identifies its kernel at the stochastic instance. The other half, the identification of Run with a delayed trace on the synchronous

product, belongs to Part III, and the stability statement for linear sampled-data loops belongs to Part V.

11.2 Prior art

The question that organises the prior work is which object deserves the name. Russell and Norvig separate the agent function, a map from percept histories to actions, from the agent program that computes it [14]; that distinction is the one between $[\![A]\!]$ and the coalgebra, and it is where a formal treatment has to start, because the two have different notions of equality. Mealy machines give the finite-state version of the agent program [8], and Rutten's coalgebraic treatment identifies the agent function with the unique morphism into a final coalgebra of causal stream functions [16, 15]. Theorem 6.2 is that result with two additions, typed termination and failure, and with the stuttering requirement, which is what makes the final object a set of realisable behaviours rather than of arbitrary pairs.

Effects are the point at which the classical story stops being enough. Moggi's monadic account of notions of computation supplies the general apparatus [10], Kock's commutative monads supply the condition under which two effects computed in one tick do not depend on their order [6], and Power and Robinson describe what remains when that condition is dropped [13]. The series takes commutativity as a hypothesis rather than a conclusion, and pays for it by excluding state and input-output from T; those are modelled as agents and wiring in Parts III and IV instead. Jacobs develops relation lifting and bisimulation for functors composed with a monad [5], and Hasuo, Jacobs and Sokolova obtain trace semantics for such systems by a final-coalgebra construction in a Kleisli category [4]. Section 4 could have been derived from that framework; it is instead proved from a seven-clause lifting calculus, so that each step can be checked at each of the three instances by hand.

Bisimulation itself is due to Park and to Milner [12, 9], and its probabilistic form to Larsen and Skou, whose lifting by couplings is the definition used here and whose logical characterisation explains why the relation is the right one for a stochastic policy [7]. Deciding it on finite systems is classical; Baier, Engelen and Majster-Cederbaum give the partition-refinement algorithm for the probabilistic case with the bound quoted in Theorem 7.2(iv) [3]. The separating example of Theorem 4.6 is the standard one from the process-calculus literature, and the only new work in transporting it is checking that typed termination does not repair it.

Two contemporary lines are close relatives rather than ancestors. Interaction trees represent effectful, possibly non-terminating programs as coinductive trees of uninterpreted events with continuations, and reason about them up to a bisimulation that ignores silent steps [17]. The object here differs in three ways that matter for the series: the interface is fixed and synchronous, so a tick is a unit of time and not only of computation; effects are interpreted in a commutative monad on sets rather than left uninterpreted; and termination is typed and failure is an outcome, which is what gives the eventual category its objects. Polynomial functors, in the treatment of Niu and Spivak, generalise the fixed pair (I, Σ) to dependent interfaces where the available inputs depend on the state [11]; that generality would subsume Theorem 2.1, and Section 12 records it as the natural direction in which the interface commitment could be relaxed.

Reactive Modules and the refinement-mapping literature supply the discipline for comparing systems rather than states. Alur and Henzinger organise synchronous and asynchronous composition with hiding and assume-guarantee reasoning [2]; the one-tick-one-step commitment of Table 1 follows their synchronous round. Abadi and Lamport characterise when a refinement can be witnessed by a mapping, with history and prophecy variables as the price [1]. The refinement of Theorem 5.4 is coarser and cheaper: it compares observation sets directly and needs no auxiliary state, and it is exactly what Part IV requires when it compares a supervised execution with a crash-free one.

12 Limitations

The series claim, quoted verbatim in Section 1 and repeated here, is this.

> 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 Part proves only the sentence about the object. What is established here is the signature of Theorem 2.10, the relations of Table 2 with the inclusions of Theorem 4.4, Theorem 6.2 and Theorem 4.6, the invariant lattice of Theorem 8.3, the minimal quotient of Theorem 7.2 for finite state spaces, and the closed-loop kernel of Theorem 9.3. The last of these adds three hypotheses: the distribution monad, finite state spaces, and lockstep scheduling with the register initialised at (ε, τ). No operator is treated, and the sixteen-field adequacy results other than the two rows of Section 11 belong to later Parts.

Seven restrictions are real and are not artefacts of the presentation.

Time is discrete and synchronous. Every tick consumes one input and emits one output, so an agent that computes for a long time between observable actions is modelled by a run of ticks emitting τ, and the number of such ticks is part of the model. Continuous time is outside the series, and sampled-data control is treated in Part V only in its sampled form.

Effects are commutative. State and input-output are therefore not available as instances of T, and the premonoidal setting that would admit them [13] is not developed. Shared state and logging appear in the series as agents and wiring, which is a modelling choice with observable consequences: a shared store has its own state space and its own ticks.

Nondeterminism and probability are never combined. No instance of T is a convex combination of $\mathcal{P}$ and $\mathcal{D}$, so a claim about an agent that is both nondeterministically scheduled and stochastically sampled is outside every law of the series. This is the restriction most likely to bind in practice, because a fair scheduler over stochastic components is exactly that combination.

The semantics is finite-trace. $[\![A]\!]$ is defined on finite input words, so liveness properties, fairness assumptions and infinite runs are not expressible at this level, and the fairness clause of Theorem 5.2 is stated for the use of later Parts rather than used here. Divergence is visible only as the absence of a halting outcome at every finite length.

Refinement ignores idle timing. By Theorem 5.3 an inserted idle tick is invisible, so no real-time or throughput property can be stated as a refinement. Part IV's timing results are therefore tick counts, not refinements.

Decidability claims assume finite state spaces. Theorem 7.2(iv) is stated for finite S_A and rational weights; nothing here decides bisimilarity for an agent whose state is an unbounded history, and the code layer's checks explore agents only up to a stated state cap and tick budget.

Full abstraction is not established. Bisimilarity is sound for the operators of the later Parts, but whether it is fully abstract for a contextual equivalence defined by observing halting values

is open, and so is the question of what the coarsest such congruence is. Section 11.2 names the polynomial-functor setting as the natural place to ask the same questions with dependent interfaces.

Two further caveats concern evidence rather than scope. The finite-model checks reported in Section A run the constructions of this Part on generated finite agents and on the fixed witnesses; they are checks against a finite model and they establish nothing about the theorems, which rest on the proofs given above. And the design records one correction to the series planning documents: Theorem 9.3 is stated on $S_A \times S_E \times I \times \Sigma$ rather than on $S_A \times S_E$, because Theorem 9.5 shows that the projection to the component states is not a Markov chain.

13 Conclusion

An agent is a coalgebra of $F(S) = (Y + E + 1) \times (T(\Sigma \times S))^I$ with a required stuttering rule at non-running states, and the four relations that the rest of the series quotes are ordered by the results proved here: bisimilarity is contained in trace equivalence and in accepted-trace equivalence and implies mutual refinement, the first containment is an equality exactly at the identity monad, and the failure at the other two instances is witnessed by a two-state example. The lattice of invariants is complete and the greatest invariant inside a predicate is a greatest fixed point; finite agents have minimal quotients that are canonical once unreachable states are discarded; and closing an agent with an environment yields, at the distribution monad, a Markov chain whose kernel is the product of the component kernels.

What remains uncertain at this level is not the object but its adequacy as a foundation for the operators. Nothing proved here shows that a composition operator respects bisimilarity, that feedback has a fixed point, or that a supervised execution refines an unsupervised one; those are the subjects of Parts II to IV, and each of them is a separate proof obligation with its own hypotheses. The object earns its place only if those proofs go through at the relations named here.

A Code evidence

The companion code layer implements the constructions of this Part over a finite fragment: finite state spaces explored to a stated cap, bounded tick budgets, and rational weights for $\mathcal{D}$. Each row below names the claim, the implementing identifier, the test that exercises it, and what the test shows. The character › separates the test file, the describe block and the test title, which are quoted as they appear in the code layer's index. These are finite-model checks and exhaustive bounded checks over that fragment; they rule out the cheap ways of being wrong and they are not verification. Each row states the bound its check carries, and a row that names one instance or one witness says nothing about the others. Every theorem and proposition of this Part rests on the proof given in its section.

Claim (label)	File and identifier	Test (file › describe › it)	Runs	What the test shows
Theorem 7.2(iv), decidability	`src/equiv.ts`, `bisim` and `bisimBrute`	test/part1-agent-object.test.ts › bisim-checker-agreement › partition refinement agrees with brute-force bisimulation on random Id, P and D agents	100 per instance, seed 20260903	On generated three-state agents at each of the three instances, the refinement procedure returns the same answer as an exhaustive search over all partitions. The check is bounded to those agents and that sample.

Claim (label)	File and identifier	Test (file › describe › it)	Runs	What the test shows
Theorem 4.4	`src/equiv.ts`, `bisim` and `traceEqFinite`	test/part1-agent-object.test.ts › bisim-implies-trace › bisimilar random pairs are finite-trace equivalent	100 per instance, seed 20260903	Among the generated pairs at each instance, every pair the checker reports bisimilar is also trace equivalent on input words up to length three. The check is a bounded search for a counterexample and is vacuous on samples that contain no bisimilar pair.
Theorem 4.6	`src/equiv.ts`, `bisim` and `traceEqFinite`	test/part1-agent-object.test.ts › bisim-implies-trace › the a.(b+c) versus a.b+a.c witness is trace-equivalent but not bisimilar	one fixed witness, exhaustive to length four	At the powerset instance the fixed witness is trace equivalent on all input words up to length four and is not bisimilar, by both the refinement procedure and the exhaustive partition search. The distribution instance rests on the proof.
Theorem 8.3	`src/equiv.ts`, `greatest Invariant` and `allInvariants`	test/part1-agent-object.test.ts › greatest-invariant › the gfp iteration returns an invariant containing every enumerated invariant inside Phi	100, exhaustive over all subsets	At the powerset instance, on generated agents small enough for every subset of the predicate to be enumerated, the fixed-point iteration returns an inductive set containing every inductive subset of it. The other two instances rest on the proof.
Theorem 7.2(ii) and (iii)	`src/equiv.ts`, `quotient` and `bisim`	test/part1-agent-object.test.ts › quotient › A/~ is bisimilar to A and has no two bisimilar states	100 per instance, seed 20260903	On generated four-state agents at each instance, the computed quotient is bisimilar to the original, has no two distinct bisimilar states, and has no more states. Clause (i) rests on its proof.
Theorem 9.3	`src/ semantics.ts`, `runKernel`; `src/ effects.ts`, `eprod`	test/part1-agent-object.test.ts › closed-loop-markov › Run(A,E) for T = D is row-stochastic and equals the product-kernel formula	one fixed loop, exhaustive over reachable states	For one fixed closed loop with rational weights, the one-step matrix has rows summing to one and equals the product of the two component kernels entry by entry, in either binding order. Every reachable state of that loop is running, so the check bears on the running case of the proposition.

Claim (label)	File and identifier	Test (file › describe › it)	Runs	What the test shows
Theorem 6.2	`src/equiv.ts`, `bisim` and `traceEqFinite`	test/part1-agent-object.test.ts › final-mealy › for T = Id the bisimulation and trace-equivalence decisions agree on random Mealy machines	100, seed 20260903	At the identity monad the bisimulation and trace-equivalence decisions agree on generated three-state machines with input words up to length six, which is the decidable and bounded shadow of the finality theorem.
Theorem 5.5, stutter insensitivity	`src/equiv.ts`, `refines` and `stutter`	test/part1-agent-object.test.ts › refinement-preorder › inserting random stutter ticks gives A refines A' and A' refines A	100, seed 20260903	On generated powerset agents, a stutter-inserted agent and the original include each other's stuttering-closed observations within the stated tick budgets, and the quotient of an agent refines it in both directions. Reflexivity and transitivity of $\sqsubseteq$ rest on the proof.

References

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

[2] Alur, R. and Henzinger, T. A. (1999). Reactive modules. *Formal Methods in System Design* 15(1):7-48. DOI: 10.1023/A:1008739929481.

[3] Baier, C., Engelen, B. and Majster-Cederbaum, M. (2000). Deciding bisimilarity and similarity for probabilistic processes. *Journal of Computer and System Sciences* 60(1):187-231. DOI: 10.1006/jcss.1999.1683.

[4] Hasuo, I., Jacobs, B. and Sokolova, A. (2007). Generic trace semantics via coinduction. *Logical Methods in Computer Science* 3(4):11:1-36. arXiv:0710.2505.

[5] Jacobs, B. (2016). *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge Tracts in Theoretical Computer Science 59, Cambridge University Press. ISBN 978-1-107-17789-6.

[6] Kock, A. (1972). Strong functors and monoidal monads. *Archiv der Mathematik* 23:113-120. DOI: 10.1007/BF01304852.

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

[8] Mealy, G. H. (1955). A method for synthesizing sequential switching circuits. *Bell System Technical Journal* 34(5):1045-1079.

[9] Milner, R. (1989). *Communication and Concurrency*. Prentice Hall.

[10] Moggi, E. (1991). Notions of computation and monads. *Information and Computation* 93(1):55-92. DOI: 10.1016/0890-5401(91)90052-4.

[11] Niu, N. and Spivak, D. I. (2023). *Polynomial Functors: A Mathematical Theory of Interaction*. arXiv:2312.00990; published as London Mathematical Society Lecture Note Series 498, Cambridge University Press, 2025.

[12] Park, D. (1981). Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, LNCS 104, pages 167-183. Springer.

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

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

[15] 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.

[16] Rutten, J. J. M. M. (2006). Algebraic specification and coalgebraic synthesis of Mealy automata. *Electronic Notes in Theoretical Computer Science* 160:305-319.

[17] Xia, L., Zakowski, Y., He, P., Hur, C.-K., Malecha, G., Pierce, B. C. and Zdancewic, S. (2020). Interaction trees: representing recursive and impure programs in Coq. *Proceedings of the ACM on Programming Languages* 4(POPL), article 51. DOI: 10.1145/3371119.