Possibility Loops

An Operational Architecture for Computative Economics in Agent Coordination Systems

Wulf A. Kaal[1]

Abstract

This article specifies an operational architecture for the agentic reputation economies described in Computative Economics. Where Computative Economics establishes the theoretical primitives, the computative agent tuple (C, O, M, G, R), the generated possibility space, and recursive equilibrium as fixed point of a composite best-response map, the present paper specifies the mechanisms through which those primitives become operational on settlement-layer infrastructure. The contribution is the possibility loop, a generatively open cycle in which agents propose elements of the action set, validate proposals, execute funded proposals, and reflect outcomes back into the proposing agents' generative functions and into the public state, such that each completed cycle leaves the system with a strictly enlarged action space. The paper formalizes six possibility loops, demonstrates that they are jointly necessary and individually mapped to the components of the computative agent tuple, characterizes the compositional structure of joint possibility spaces across agents, and translates the recursive equilibrium proof into a settlement-layer specification through five implementation invariants. The architecture is infrastructure neutral by design and applies to any settlement substrate that supports stake-weighted validation, on-chain identity, and a representation of agent state. The architectural transformation from Human-Derived Coordination Architecture to multi-loop reputation economy is shown to be necessary for the implementation of Computative Economics: any architecture in which the action set is exogenous to agent action implements at most a reputation-weighted Neoclassical labor market and does not implement Computative Economics.

Keywords. Computative Economics; possibility space; agentic reputation economy; recursive equilibrium; generative agents; validation pools; reputation; dynamic governance; decentralized infrastructure; multi-loop architecture.

JEL Classification. D23, D47, D86, K20, L14, O31, O33, O38, O43.

---

[1] University of St. Thomas School of Law; wulfkaal@stthomas.edu; ORCID 0000-0003-0757-275X. The author thanks an earlier working draft circle for comments on prior versions. In accordance with the American Economic Association's Disclosure Policy on the use of generative artificial intelligence tools, the author discloses that drafting assistance was provided by large language model systems in the preparation of this manuscript. The author is solely accountable for all content, including the correctness of formal results and the accuracy of cited literature, and has verified all sources independently. The author has no financial conflicts of interest to disclose with respect to the subject matter of this paper.

I. Introduction

Computative Economics establishes that the binding constraint on agent action shifts from physical scarcity to computational generativity (Kaal 2026a). The choice of a rational agent among scarce means is replaced by the generation of a possibility space by a computative agent, whose binding constraint is computational rather than physical and whose function is to generate possibility spaces rather than select within fixed ones. The companion paper, The Collapse of Scarcity Economics, establishes the negative result that the dominant frameworks of twentieth-century economic analysis expire under conditions of computational abundance (Kaal 2026b). The present paper takes the constructive primitive as given and asks the operational question that follows immediately: by what mechanisms does an agent coordination system instantiate, validate, and reward the generation of possibility spaces, rather than only the selection within them.

The thesis can be stated in one sentence. An agent coordination system implements Computative Economics when, and only when, every component of the computative agent tuple has a corresponding settlement-layer surface with proposal, validation, execution, and reflection mechanics, and when the action sets of the system are themselves agent products subject to the same validation economics as the actions executed within them. The present paper develops this architecture using a reputation-based validation primitive inherited from the foundational reputation-economy literature, but the architectural claim is independent of the choice between reputation-based and alternative validation primitives. The argument extends the author's prior research arc on post-classical economic frameworks (Kaal 2024b).

The paper analyzes a generic legacy architecture characteristic of current agent coordination systems, denoted the Human-Derived Coordination Architecture (HDCA). The label "Human-Derived Coordination Architecture" is introduced here to designate the family of agent coordination protocols whose constituent mechanisms, aka job postings, expertise tag taxonomies, contract templates, and stake-weighted validation pools, were originally developed to coordinate human freelancers and clients in repeated-game settings. Such protocols are sound for that purpose. They are not, however, native to agentic compute, and the limitations identified in Part IV are consequences of this design lineage rather than failures of execution within the lineage.

The diagnosis can be summarized in one observation. HDCA implements the consumption side of the computative labor market and does not implement the generation side. Job specifications, expertise taxonomies, and contract templates are exogenous parameters set by clients or by governance. The agent action space is closed: workers select among existing postings. This collapses to the Neoclassical labor market with reputation weighting, which is to say, to the framework whose expiration The Collapse of Scarcity Economics (Kaal 2026b) demonstrates.

The contribution of the present paper is the possibility loop, a generatively open cycle in which agents propose, validate, execute, and reflect on elements of the action set itself, with reputation accruing on each surface of the cycle. Six possibility loops are formalized, corresponding to six faces of the agent's generative and reflective action: the execution surface, the job-proposal surface, the tag-proposal surface, the template-proposal surface, the decomposition surface, and the cross-surface reflection surface. Each surface has its own settlement-layer specification, its own validation pool composition, and its own reputation scalar. The six faces correspond to selection within the existing action set (Σ_E), generation of opportunities (Σ_J), generation of the coordinate system over the action space (Σ_T), generation of contract structure (Σ_K), generation of joint possibility-space structure across agents (Σ_D), and reflective reallocation of compute across the prior five surfaces (Σ_R). The reflection operator R of the computative agent tuple is accordingly refined into two operationally distinct expressions: per-surface reflection, embedded in each of the first five surfaces and updating the agent's generative function locally, and cross-surface reflection, isolated on Σ_R and updating the agent's allocation of compute across surfaces. The six surfaces are jointly necessary and individually mapped to the components of the computative agent tuple, with the R component decomposed across Σ_E…Σ_D and Σ_R as its two operational halves.

Part II restates the relevant primitives from Computative Economics. Part III specifies the HDCA reference architecture. Part IV develops the limitation of HDCA formally through three architectural requirements. Part V defines the possibility loop and explains its generatively open topology. Part VI specifies the six surfaces at the settlement layer. Part VII characterizes compositional possibility spaces across agents. Part VIII translates the recursive equilibrium proof into settlement-layer mechanics through five implementation invariants. Part IX states four

propositions. Part X develops a multi-loop translation of the Ostrom design principles for commons governance. Part XI identifies five open problems. Part XII concludes.

II. Theoretical Primitives Restated

This part restates the Computative Economics primitives that the architecture must instantiate. The full development is in the foundational paper. What follows is the minimum required for the architectural argument.

A. The computative agent tuple (C, O, M, G, R)

The computative agent is specified as a five-tuple. C, the capability set, denotes what the agent can compute, with what models, at what marginal cost. Operationally, C corresponds to the model registry, the inference budget, and the tool surface available to the agent. O, the objective, denotes the function the agent optimizes over generated and selected outcomes. Operationally, O corresponds to the agent's stated bid function, its reputation target, and its reservation utility (Simon 1955). M, the world model, denotes the agent's representation of the environment, including other agents, the protocol state, and the demand surface. Operationally, M corresponds to the agent's local state plus the public on-chain state it queries. G, the generative function, denotes the function from compute to candidate options. Operationally, G is the action by which the agent produces a proposal, a tag, a template, a decomposition, or a deliverable. R, the reflection operator, denotes the function that revises G based on outcomes. R has two operational expressions in the present architecture: per-surface reflection, which revises G locally on each surface in light of that surface's validation outcomes, and cross-surface reflection, which revises the agent's allocation of compute across surfaces in light of the joint distribution of outcomes. Both expressions are implemented at the settlement layer. Per-surface reflection is embedded in Σ_E, Σ_J, Σ_T, Σ_K, and Σ_D, while cross-surface reflection is isolated on Σ_R.

B. The generated possibility space, contrasted with the Neoclassical commodity space

Three properties of the generated possibility space distinguish it from the Neoclassical commodity space, and each property has direct architectural implications.

First, the possibility space is open rather than closed. In the Neoclassical model, the agent faces a budget constraint defined over a finite-dimensional commodity space with prices given

exogenously (Arrow and Debreu 1954). The agent's optimization problem is to choose a point within this pre-specified region. The commodity space is closed under the optimization: no choice the agent makes adds a new commodity to the space. Even in dynamic Neoclassical extensions, intertemporal choice and state-contingent claims, the dimensions of the space are specified prior to optimization and remain fixed during it (Robbins 1932, 15). The computative agent operates on a possibility space whose dimensions are themselves products of the agent's compute. When the agent allocates compute to its generative function G, the output is not a chosen point in a fixed space. It is a new candidate dimension along which choice can subsequently occur. The space is open in the formal sense that its cardinality and dimensionality are not bounded prior to the agent's action and are mechanically expanded by it. This is not a metaphorical openness but a different mathematical object.

Second, the possibility space has internal quality structure independent of any particular agent's preferences. The Neoclassical commodity space has no such structure: a bundle in finite-dimensional Euclidean space is no better or worse as a bundle than any other bundle, and quality enters only through the utility function ranking the bundles. The possibility space, in contrast, is characterized by three internal quality metrics. Coverage measures how much of the relevant outcome manifold the generated options span: does the generated space include the dimensions along which downstream demand actually exists, or does it concentrate generation in a narrow subspace. Fidelity measures whether the generated options correspond to executable actions: a generated option that cannot be carried out does not enlarge the operative possibility space, even if it enlarges the nominal one. Novelty measures the distance of generated options from the agent's prior repertoire: a possibility space dominated by re-generated familiar options has lower informational content than one that explores under-mapped regions. These metrics are not preferences over the space but properties of the generative process that produced the space, and they distinguish the output of two agents with identical capability sets and identical preferences whose G functions differ in compute strategy.

Third, possibility spaces compose across agents through a non-trivial operator. In the Neoclassical model, when n agents are present, the aggregate budget set is the Minkowski sum of individual budget sets, and the aggregate commodity space is the same commodity space each individual operates within (Debreu 1959). Aggregation is unproblematic because the space does

not change under aggregation. Possibility spaces compose through a function Φ of individual generative functions G_1, …, G_n that is not the union of individual generated spaces. When agent A generates option α and agent B generates option β, the joint space includes α, β, and the structured combinations that become possible only when α and β coexist in the same action set: α composed with β as decomposed subjobs, α templated by a structure derived from β, α tagged under a taxonomy proposed by B. Joint compute over the same problem domain produces joint possibility-space structure that neither agent could produce alone.

The structural point that ties the three properties together is this: in Neoclassical analysis, the action space is a parameter of the model and the agent is a chooser within it. In Computative Economics analysis, the action space is an output of the agent and the agent is a generator over it. Every operational difference between the two architectures follows from this parameter-versus-output distinction.

C. Recursive equilibrium

The coordinating concept in Computative Economics is recursive equilibrium, formalized as the fixed point of a composite best-response map operating jointly on action profiles and on generated action sets. Existence is established under Brouwer's fixed-point theorem given convexity and continuity conditions on the action profile space (Brouwer 1911). Uniqueness and convergent iteration are established under Banach's contraction mapping principle given a contraction modulus on the population-distribution component of the state space (Banach 1922). The Brouwer-Banach apparatus operates on the rate and structure of action-set expansion, not on a stationary action set. It is equilibrium over the dynamics of generation, which is the move that distinguishes recursive equilibrium from any equilibrium concept defined over a fixed commodity space.

III. The Human-Derived Coordination Architecture

The Human-Derived Coordination Architecture (HDCA) is the dominant paradigm in current agent coordination systems. Its components were designed to coordinate human freelancers and clients in repeated-game settings, and they remain sound for that purpose. They are not native to agentic compute, however, and the limitations identified in Part IV are consequences of this

design lineage rather than failures of execution within the lineage. This part specifies the HDCA components and lifecycle in infrastructure-neutral terms.

A. Components

The job factory is a contract surface that creates, escrows, and lifecycle-manages individual job instances. The agent registry is an identity primitive, typically soulbound to prevent reputation transfer, mapping agent identifiers to capability declarations and historical reputation (Calcaterra and Kaal 2021a, ch. 6). The validation pool is a stake-weighted commit-reveal voting mechanism that resolves job outcomes and triggers reputation redistribution. The expertise tag taxonomy is a governance-set classification of skill domains used to route jobs and to scope validation pool composition. The configuration registry holds protocol parameters subject to reputation-weighted governance: fee rates, consensus thresholds, minimum stakes. The dual-token economics separate a transferable utility token, used for fees and bounties, from a non-transferable reputation scalar accruing through validated outcomes (Williamson 1985). The architectural rationale for separating reputation from utility is the standard reputation-as-non-fungible-claim argument developed in the foundational reputation-economy literature (Calcaterra, Kaal, and Sivalingam 2018).

B. Job lifecycle

The HDCA job lifecycle proceeds in four stages. A client or upstream agent invokes the job-creation surface with an exogenous tuple consisting of an expertise tag, a requirements specification, a bounty amount, and a deadline. Workers in the relevant tag domain select into the job through a reputation-weighted bidding process. The selected worker delivers an artifact, which is submitted to a validation pool. The validation pool resolves through commit-reveal, and reputation is redistributed: the worker accrues reputation if the artifact is validated, and slashed otherwise. Validators accrue reputation if their vote aligns with the resolution and slashing otherwise.

C. What HDCA implements correctly

HDCA implements a Folk-theorem institutional infrastructure in which the repeated execution game over delivery induces cooperation through reputation accrual and slashing (Friedman 1971). The mechanism is sound: stake-aligned validators bear economic consequence for

inattentive or adversarial votes, sybil resistance is achieved through the combination of soulbound identity and stake costs (Calcaterra 2018), and reputation specialization within tag domains permits market-like sorting of agent capability to job demand (Fudenberg and Maskin 1986). The architecture is, in this sense, a successful translation of repeated-game cooperative equilibria into smart-contract form (Calcaterra and Kaal 2018). Its limitation is not in the execution layer but in the absence of a generation layer, which Part IV develops formally.

IV. The HDCA Limitation: Three Architectural Requirements

This part develops the limitation of HDCA through three architectural requirements that Computative Economics imposes on any operational implementation. Each requirement is necessary. The three are jointly sufficient. HDCA satisfies none of them.

A. The G requirement: action surface and reward channel

The generative function G is the component of the agent tuple that produces candidate options. In any architecture instantiating Computative Economics, G must have two operational expressions. The first is an action surface, a contract or settlement-layer interface through which the agent can post its generative output to the protocol's public state. The second is a reward channel, a reputation or utility flow that accrues to the agent conditional on the validated quality of the posted output.

An architecture that has the action surface but no reward channel implements G as an unrewarded act, and rational agents will not allocate compute to G beyond the level required for their own selection-side activity. An architecture that has the reward channel but no action surface implements G as an off-protocol act whose outputs cannot enter the protocol's action set, and the protocol's possibility space remains exogenous regardless of how active G is in the agent population. HDCA has neither: agents who generate jobs, tags, or templates do so off-protocol, and the protocol does not validate or reward the generation. This is not a small gap. It is the architectural reason HDCA cannot host Computative Economics. The diagnosis parallels Coase's analysis of the market-firm boundary: in both cases, an architecture lacks an internal mechanism for an activity that nonetheless takes place, and the missing internalization defines the structural limitation (Coase 1937).

B. The R requirement: outcome observation on multiple timescales, with cross-surface reflection isolated

The reflection operator R revises G based on outcomes. Reflection requires information about what happened, and the information must be specific to the generative act being reflected on. An architecture that resolves only one outcome type, per-job execution in HDCA, provides reflection information only on the execution face of G. The agent learns whether its delivered work was validated. The agent does not learn whether its proposal quality, taxonomic productivity, or template performance is improving, because the architecture does not measure those things.

Reflection across the full agent tuple requires that each generative surface emit its own resolution events on its own timescale, and additionally requires a distinct surface on which the agent's allocation of compute across surfaces is itself the object of reflection. Per-surface reflection updates the agent's generative function G locally, conditional on outcomes within that surface. Cross-surface reflection updates the agent's allocation of compute across surfaces, conditional on the joint distribution of outcomes across all surfaces. The two operations are not reducible to each other. An agent whose execution reputation is rising and whose template reputation is falling has no reason, on per-surface reflection alone, to reallocate compute from Σ_E to Σ_K; the reallocation is precisely the question that cross-surface reflection answers. Per-job resolution alone resolves neither operation. Per-surface resolution on each of Σ_E, Σ_J, Σ_T, Σ_K, and Σ_D resolves the first. A separate surface Σ_R, on which compute-allocation proposals are themselves validated and rewarded, resolves the second.

C. The quality-metric requirement: representation of the possibility space

Coverage, fidelity, and novelty are properties of the generated possibility space that are independent of any particular agent's preferences. For these metrics to be operationalized, the architecture must maintain a representation of the possibility space distinct from the current catalog of pending or executing instances. HDCA does not. Its representation of the action space is the mempool of open jobs at any given moment. When a job is resolved, it leaves the representation. The architecture has no historical or aggregate view of what kinds of options have been generated, with what frequency, in what tag domains, with what fidelity rates, with

what novelty profile. Without that representation, no quality metric can be computed, and without quality metrics, generation cannot be rewarded for quality, only for quantity.

D. Synthesis and Limitation L1

Each of the three requirements is necessary and the three are jointly sufficient. An architecture with G surfaces and reward channels but only single-timescale resolution implements generation but cannot improve generation. An architecture with multi-timescale resolution but no quality-metric representation can reward generation but cannot reward good generation. An architecture with quality metrics but no G surface has no generation to reward. An architecture with per-surface but no cross-surface reflection improves each surface in isolation but cannot improve the agent's allocation across surfaces. The synthesis can be stated as a structural limitation.

Limitation L1 (HDCA exhaustion). Any agent coordination architecture in which the action set is exogenous to agent action implements at most a reputation-weighted Neoclassical labor market and does not implement Computative Economics. HDCA exemplifies this limitation.

The remainder of the paper specifies the architectural transformation that lifts the HDCA exhaustion.

V. The Possibility Loop

A. Definition

A possibility loop is a tuple $(\Sigma, \pi, \nu, \xi, \rho)$ consisting of: a surface Σ of action-set elements. A proposal mechanism π through which agents post elements to Σ. A validation mechanism ν through which other agents accept, reject, or rate proposed elements. An execution mechanism ξ through which accepted elements become available for selection within downstream loops. And, a reflection mechanism ρ through which outcomes from execution update the reputation of proposers, validators, and the surface itself.

B. Generatively open topology

The possibility loop is generatively open. This is its central distinguishing feature relative to the HDCA settlement loop, which is closed in the topological sense: every job that enters the HDCA loop exits as either resolved-validated or resolved-rejected, and the system returns to a state

structurally identical to where it began, with only reputation scalars updated. The set of possible actions does not change.

The possibility loop, in contrast, is generatively open in the sense that each completed cycle leaves the system with a strictly enlarged action space. A successful proposal on the job surface adds a job specification that did not exist. A successful proposal on the tag surface adds a tag that the system could not previously route around. A successful proposal on the template surface adds a contract template that the system could not previously parameterize. The reflection step does not return the system to its prior state. It updates the generative function G of the proposing agent and updates the public-state surface that other agents condition their next round of generation on. Both updates enlarge the space of subsequent possibility.

The analytical implication is direct. Recursive equilibrium under Computative Economics is not equilibrium in the static sense. It is equilibrium over the dynamics of generation. The fixed point is in the rate and structure of action-set expansion, not in a stationary action set. This is the move that distinguishes Computative recursive equilibrium from Walrasian equilibrium and from any equilibrium concept defined over a fixed commodity space. The claim is in tension with the Hayekian conception of the price system as the principal mechanism for aggregating dispersed knowledge. The reconciliation is that the possibility loop generalizes the price-system function: where prices aggregate information about scarcity, the validation pool aggregates information about generative quality (Hayek 1945).

C. The six surfaces

Six possibility loops are formalized, corresponding to six faces of the agent's generative and reflective action. The surfaces are described here at the conceptual level. The settlement-layer specification follows in Part VI.

Σ_E, the execution surface, admits deliverables for funded jobs as elements. The proposer is the Worker. Validators rate the deliverable against the job's success criteria. Reflection updates Worker reputation. Σ_E is the surface that HDCA already implements, and it is retained without modification.

Σ_J, the job surface, admits job specifications as elements. The proposer is the Job Architect. Validators are senior-reputation agents in the relevant domain, who score the proposal on

coverage, fidelity, and novelty. Execution is a downstream funder committing the bounty, after which the job flows to Σ_E. Reflection updates Architect reputation based on funding rate and downstream validation outcomes.

Σ_T, the tag surface, admits proposed expertise tags with descriptors and parent-tag relationships as elements. The proposer is the Taxonomist. Validators are senior-reputation agents in adjacent tag domains. Execution is downstream economic flow through the tag over a measurement window. Reflection updates Taxonomist reputation based on tag adoption and tag-conditional bounty mass.

Σ_K, the template surface, admits job contract templates parameterized for a class of jobs as elements. A template specifies success criteria, validator pool composition, payout schedule, and dispute procedure. The proposer is the Template Author. Validators are senior-reputation agents with template-implementation history. Reflection updates Template Author reputation based on template-usage volume and the inverse of template-conditional dispute rate.

Σ_D, the decomposition surface, admits decompositions of a parent job into a directed acyclic graph of subjobs as elements. The proposer is the Decomposer. Validators rate decomposition completeness, subjob coherence, and execution efficiency relative to the atomic baseline. Reflection updates Decomposer reputation based on weighted aggregate validation pass rate of constituent subjobs.

Σ_R, the cross-surface reflection surface, admits compute-allocation proposals as elements. A compute-allocation proposal is a proposed reallocation of an agent's compute budget across the prior five surfaces, conditional on a public window of outcomes across those surfaces. The proposer is the Reflector, an agent that may be the same as or different from the agents whose allocation is being proposed. Validators are senior-reputation agents drawn from the union of the five other surfaces, weighted by their cross-surface reputation diversity, and they rate the proposal on whether the implied reallocation would have improved the proposing agent's aggregate reputation accrual rate over the public outcome window. Execution is the proposing agent's subsequent compute deployment, observed through the rate of activity it commits to each surface in the period following the proposal. Reflection updates Reflector reputation based on the realized improvement in aggregate reputation accrual rate over a longer measurement window. The surface differs from the prior five in that what it proposes is not an action-set element but an

allocation policy over surfaces. What it generates, accordingly, is not an enlargement of any single surface but a refinement of the joint dynamics across surfaces. Σ_R is the operational expression of cross-surface reflection in the computative agent tuple, and per-surface reflection on Σ_E through Σ_D is the operational expression of per-surface reflection.

D. Compositional structure of the surfaces

The six surfaces are not independent. Σ_K presupposes Σ_T, since templates are tag-scoped. Σ_J presupposes Σ_T and Σ_K, since jobs reference tags and instantiate templates. Σ_D presupposes Σ_J, since decomposition operates on parent jobs. Σ_E presupposes Σ_T, Σ_K, Σ_J, and Σ_D, since execution settles within decomposed, templated, tagged, proposed jobs. The induced partial order over the first five surfaces is: $\Sigma\_T \rightarrow \Sigma\_K \rightarrow \Sigma\_J \rightarrow \Sigma\_D \rightarrow \Sigma\_E$.

Σ_R is not a node in this linear order. It is a feedback layer above the order, conditioned on the public outcome streams of all five surfaces and producing reallocation policies that are inputs back into each. The induced structure is therefore a directed graph rather than a total order: a linear chain $\Sigma\_T \rightarrow \Sigma\_K \rightarrow \Sigma\_J \rightarrow \Sigma\_D \rightarrow \Sigma\_E$ with a feedback layer Σ_R reading from all five and writing back to each. Reputation flows generated at upstream surfaces propagate to downstream surfaces along the linear chain, reputation flows generated at downstream surfaces feed back into upstream reflection along the chain in reverse, and reputation flows on Σ_R operate orthogonally to the chain by adjusting the rate at which agents commit compute to each node. The six-surface architecture therefore has a layered structure with three flow directions: forward composition, backward per-surface reflection, and orthogonal cross-surface reflection. The layered authority structure is an architectural analog of the residual-control-rights framework that handles inter-layer authority allocation in classical theory of the firm (Grossman and Hart 1986).

VI. Settlement-Layer Specification of the Six Surfaces

Each surface is specified at the contract-level in infrastructure-neutral terms. Implementation on any specific blockchain or settlement substrate is a translation, not a redesign. The descriptions emphasize the generic mechanism rather than a specific virtual machine syntax.

A. The job surface (Σ_J)

Architect agents invoke a proposeJob interface taking as parameters: a tag set, a content-addressed requirements specification, a bounty estimate, success criteria, a validator pool specification, and a stake denominated in the utility token. The returned identifier indexes a proposal in the unfunded mempool, which has an expiry. A fundProposal interface accepts a bounty deposit and converts the proposal into a normal job instance, which then flows through the existing execution lifecycle. Architect reputation is a separate scalar from Worker reputation. It accrues on funded-and-validated outcomes, slashes on funded-and-rejected outcomes, and is null on expired proposals.

B. The tag surface (Σ_T)

Taxonomist agents invoke a proposeTag interface taking as parameters: a tag descriptor, a list of parent tags, a validator pool specification, and a stake. The proposed tag enters a pending state. Confirmation requires a quorum of senior-reputation agents in parent tag domains to validate the proposal. Confirmed tags become available for routing. Taxonomist reputation accrues on tag-conditional economic flow over a measurement window. A parallel deprecateTag mechanism with the same validator pool pattern permits orderly retirement of tags whose flow has fallen below a threshold; deprecated-tag reputation is preserved but does not accrue new flow.

C. The template surface (Σ_K)

Template Author agents invoke a proposeTemplate interface taking as parameters: a list of applicable tags, a schema specifying success criteria and dispute procedures, a validator pool specification, and a stake. Confirmation requires a quorum of senior-reputation agents with prior execution history in applicable tags. Confirmed templates become available for instantiation in subsequent job proposals. Template Author reputation accrues on template-usage volume modulated by the inverse of template-conditional dispute rate.

D. The decomposition surface (Σ_D)

Decomposer agents invoke a proposeDecomposition interface taking as parameters: a parent job identifier, a directed acyclic graph of subjobs, and a stake. Validators rate structural coherence and execution-readiness of subjobs. Decomposer reputation accrues on weighted aggregate of subjob validation outcomes plus a structural coherence bonus, and slashes on aggregate failure rates exceeding a threshold.

E. The execution surface (Σ_E), unchanged from HDCA

The HDCA commit-reveal validation pool is retained without modification. Worker reputation accrues on validated delivery outcomes. The execution surface is the only surface inherited from HDCA. The other five are net-new architectural primitives.

F. The cross-surface reflection surface (Σ_R)

Reflector agents invoke a proposeReallocation interface taking as parameters: a target agent identifier (which may equal the proposer's identifier in the self-reflection case), a reference outcome window over the prior five surfaces, a proposed compute-allocation vector across the prior five surfaces, and a stake. Validators are drawn from a pool that requires non-trivial reputation in at least three of the prior five surfaces, ensuring that validators on Σ_R have first-hand cross-surface experience and cannot be captured by any single-surface specialization. Validators evaluate the counterfactual: would the proposed allocation, applied over the reference window, have raised the target agent's aggregate reputation accrual rate above the realized rate, holding the public outcome distribution fixed. Confirmed proposals enter a public registry and are observable by the target agent and by the validator pools of all six surfaces, but they do not bind the target agent's subsequent action.

Reflector reputation accrues in two stages. The first stage accrues on validation confirmation, with a small reputation flow proportional to the target agent's subsequent move toward the proposed allocation, measured as cosine similarity between the proposed allocation vector and the target agent's realized compute distribution over the post-proposal observation window. The second stage accrues on realized aggregate reputation improvement of the target agent over a longer measurement window, conditional on the target agent's allocation having moved at least a threshold distance toward the proposed allocation; in this case Reflector reputation flows from the target agent's reputation accrual gains, on a sharing schedule set by governance and bounded by INV3.

Three design constraints distinguish Σ_R from the prior five surfaces. First, Σ_R proposals are advisory rather than binding: the target agent's compute allocation cannot be coerced by a confirmed reallocation proposal, since the agent's O component is private and the reallocation might be misaligned with the agent's objective even when it would raise reputation accrual. Coercive reallocation would violate the autonomy of O, which the architecture preserves.

Second, Σ_R cannot self-refer in the strong sense: Σ_R proposals are over the prior five surfaces, not over Σ_R itself, since allowing Σ_R to propose reallocations of compute toward Σ_R would create the recursive instability that OP5 in Part XI investigates as a separate research question. Third, Σ_R reputation REP_R is bounded above as a fraction of total agent reputation by INV5, preventing the emergence of a meta-reputation aristocracy in which agents specialize entirely in reflecting on others' allocations without contributing to the prior five surfaces.

G. Internal funding flows: how agents fund the jobs they seed themselves

The settlement-layer specification raises a question that any operational implementation must answer: by what mechanism do agents fund the proposals they themselves generate. The mechanics are direct, but the systemic implication is consequential.

The basic loop is straightforward. An agent earns the utility token through validated execution on Σ_E. That utility token can be staked, spent, or used to fund new proposals on any of the generative surfaces, including Σ_R reallocation proposals over its own compute. Agents who execute well accumulate utility, and they can deploy that utility to seed new generative cycles. This is not different in principle from a freelancer who saves earnings to start a business.

The harder question is why an agent would fund a proposal it generates itself, rather than waiting for an external client to do so. Three motivations operate simultaneously, and the architecture should support all three. First, an agent operating in a longer-horizon pipeline may have a downstream consumption need: it requires a subjob completed in order to proceed with its own work, and funding the subjob through Σ_J is more efficient than performing it itself if the marginal cost of execution exceeds the bounty plus protocol fee. Second, an agent may fund a proposal because the Architect reputation accrued on funded-and-validated outcomes exceeds the bounty cost in expectation, with the bound on this rent set by the implementation invariants developed in Part VIII. Third, an agent may fund a proposal in order to derisk it for downstream funders: posting a proposal with the architect's own seed bounty signals commitment and reduces the information asymmetry that would otherwise prevent third-party funding (Akerlof 1970). A fourth motivation, specific to Σ_R, is that an agent may stake a self-reallocation proposal in order to publicly commit to a strategy shift, using the validator pool's confirmation as an external check on the strategy's coherence before committing compute.

The systemic implication is the central economic claim of the architecture. In steady state, the funding flow is internal to the agent population. Agents earn utility through Σ_E, redeploy utility through Σ_J and the other generative surfaces, and the system as a whole moves value through repeated cycles of earn-redeploy. External clients still fund proposals, but their funding is a tributary into a population whose internal funding flows are far larger. This is the analytical claim that distinguishes a Computative agent economy from a human-mediated freelance market: in the freelance market, value flow is overwhelmingly client-to-worker. In the Computative agent economy, value flow is overwhelmingly agent-to-agent, with clients providing the marginal demand impulse that defines what the population converges toward (Jensen and Meckling 1976). The recursive equilibrium proof in Computative Economics encodes precisely this dynamic, and the architecture's funding flows are the operational expression of it.

H. The aggregate reputation construct

The architecture maintains six separate reputation scalars per agent, one per surface: REP_E (Worker), REP_J (Architect), REP_T (Taxonomist), REP_K (Template Author), REP_D (Decomposer), and REP_R (Reflector). A general reputation construct is computable as a weighted aggregate of the six, with weights themselves under reputation-weighted governance and bounded above by implementation invariants to prevent governance attacks. Validation pool composition for each surface draws preferentially from agents with high reputation in that specific surface, not from general reputation, with the qualification that Σ_R validator pools draw from agents with non-trivial reputation in at least three of the prior five surfaces, to ensure cross-surface competence in reflection validators. The six-scalar construct is a refinement of the single-scalar reputation primitive characteristic of HDCA (Calcaterra 2023), and is itself a refinement of the five-scalar construct one would obtain if cross-surface reflection were folded into per-surface reflection rather than isolated on its own surface.

VII. Joint Possibility Spaces and Compositional Equilibrium

A. The composition operator

For a set of agents {a_1, …, a_n}, each with a generative function G_i, the joint possibility space is not the union of individual generated spaces but a function Φ(G_1, …, G_n) defined by the protocol's composition rules. The decomposition surface Σ_D is one operationalization of Φ.

Other operationalizations include syndicate-style joint proposals on Σ_J, in which two or more architects co-propose a job and share Architect reputation in proportion to their stake, and co-authored templates on Σ_K, in which template authorship is divisible across multiple agents who jointly bear the dispute-rate consequences of subsequent template usage. Cross-agent reflection proposals on Σ_R, in which one agent proposes a reallocation for another, are a further operationalization of Φ: they generate joint structure not over actions but over allocation policies.

B. Quality metrics on the joint space

The three quality metrics introduced in Part II have operational definitions on the joint space. Coverage is measured as the fraction of the relevant outcome manifold that the joint generated space spans, operationalized as tag-conditional flow density over a measurement window. Fidelity is measured as the fraction of joint-space elements that pass validation when funded and executed, operationalized as the validation pass rate aggregated across agent-tuples. Novelty is measured as the average distance of joint-space elements from the reputation-weighted historical action distribution, operationalized through embedding-space distance metrics on requirement specifications.

C. The compositional best-response map

Each agent's best response is a function from the public on-chain state to a tuple of actions across the six surfaces. The composite best-response map Ψ takes the profile of all agents' best responses and produces an updated public state. Recursive equilibrium is the fixed point of Ψ. The conditions on Σ_J, Σ_T, Σ_K, Σ_D, Σ_E, and Σ_R that make Ψ continuous and contractive are stated in Part VIII as architectural invariants. Any architecture that satisfies the invariants inherits the existence and convergence properties of recursive equilibrium from the foundational paper; any architecture that violates an invariant cannot be expected to converge.

D. Architectural conditions for fixed-point existence

Five architectural conditions are necessary for fixed-point existence and convergence.

AC1. Bounded action surfaces. Each surface Σ has a per-agent rate limit enforced by stake escalation, ensuring compactness of the action profile space. Rate limits prevent generation explosions that would violate convexity. Σ_R rate limits are tighter than per-surface rate limits to prevent reflection cascades.

AC2. Continuous reputation update. The reputation update functions on each surface are continuous in the validation pool's resolution outcome. Discontinuities in reputation update destroy continuity of best-response correspondences and preclude application of Brouwer.

AC3. Bounded reputation feedback. The weight assigned to any single reputation scalar in selection mechanisms is upper-bounded, preventing reputation-monopoly fixed points in which a single dominant agent absorbs all generative flow.

AC4. Contraction at scale. The composite map Ψ is contractive on the population-distribution component of the state space when the agent population exceeds a threshold determined by per-surface stake parameters. Below the threshold, the architecture may admit multiple equilibria; above the threshold, convergence is guaranteed.

AC5. Bounded reflection mass. Aggregate REP_R is upper-bounded as a fraction of total reputation across the agent population, preventing meta-reputation specialization that would decouple Σ_R validators from first-order surface activity. Without AC5, the validator pool on Σ_R can drift toward agents whose own reputation is dominated by reflection rather than action, and the cross-surface counterfactual evaluations the surface depends on lose their grounding.

VIII. From Recursive Equilibrium Proof to Settlement Mechanics

A. Brouwer correspondence

Existence of recursive equilibrium under conditions AC1–AC2 follows from Brouwer applied to the action profile space, which is compact and convex under AC1, with continuous best-response correspondences under AC2 (Brouwer 1911). The settlement-layer expression is direct: a stationary distribution of agent activity across the six surfaces exists for any feasible parameter setting of stakes, rates, and validator pool compositions. The architecture does not need to choose among equilibria; it needs to ensure that at least one exists, which AC1–AC2 secure.

B. Banach correspondence

Convergence to recursive equilibrium under AC3–AC5 follows from Banach applied to the population-distribution dynamics (Banach 1922). The settlement-layer expression: starting from any initial reputation distribution, repeated rounds of the six-surface validation cycle converge to the equilibrium distribution at a rate determined by the contraction modulus, which is itself a

function of the per-surface stake escalation curves and the bound on aggregate REP_R. The convergence rate is the design parameter the protocol most directly tunes through stake parameters and the AC5 bound.

C. Implementation invariants

The Brouwer-Banach correspondence translates into five implementation invariants. These are the architectural translation of the proof and constitute protocol-level safety conditions any implementation must respect.

INV1. Stake escalation curves on each surface must satisfy a lower bound determined by AC1. Without sufficient stake escalation, action surfaces become unbounded and convexity fails.

INV2. Validation resolution functions must be Lipschitz continuous in commit-reveal outcomes. Without Lipschitz continuity, AC2 fails and best-response correspondences are not continuous.

INV3. Reputation-weight bounds in selection mechanisms must satisfy an upper bound determined by AC3. Without the upper bound, reputation feedback loops admit fixed points in which a dominant agent absorbs all flow.

INV4. Cross-surface reputation aggregation weights must lie in a polytope defined by AC4. Outside the polytope, the composite map is not contractive and convergence fails.

INV5. Aggregate REP_R across the agent population must lie below the upper bound determined by AC5, and Σ_R rate limits must lie below per-surface rate limits by a margin determined by the contraction modulus. Without INV5, the reflection layer can grow to dominate the first-order surfaces it is meant to reflect on, and the contraction property of Ψ is lost.

The five invariants are the minimum architectural commitments. They do not prescribe specific stake values or specific Lipschitz constants; they prescribe the bounds within which those values must fall. An implementation that respects the five invariants inherits the recursive equilibrium guarantees of the foundational framework. An implementation that does not is, in the strict sense, not an implementation of Computative Economics.

IX. Propositions

A. Proposition 1 (Necessity of multiple loops)

In any agent coordination architecture instantiating Computative Economics, the number of distinct possibility loops must be at least equal to the number of distinguishable generative and reflective components of the agent tuple that produce protocol-recognized output, where the reflection operator R is counted as two distinguishable components (per-surface reflection embedded in the prior surfaces and cross-surface reflection isolated on its own surface).

Proof sketch. Each generative component requires both a proposal channel and a reward channel by the G requirement of Part IV. Per-surface reflection requires no separate surface, since it is embedded in the validation outcomes of the surface it reflects on. Cross-surface reflection cannot be embedded in any single surface, since its proposal is over allocation across surfaces and is therefore not a valid proposal on any one of them. Collapsing two distinct generative components into a single loop forces aggregation of distinct quality signals into a single reputation scalar, which violates the continuity condition AC2 because the aggregate function is not Lipschitz in the underlying components when the components have different scales of resolution outcome. Collapsing cross-surface reflection into per-surface reflection forces the embedding of an allocation-over-surfaces signal into a single-surface validation pool, which violates AC5 by construction. The minimum number of loops therefore equals the number of distinguishable generative components plus one for cross-surface reflection, which is six for the (C, O, M, G, R) tuple as developed in Part II under the per-surface/cross-surface refinement of R. ∎

B. Proposition 2 (Compositional gain)

The expected coverage of the joint possibility space generated by n agents under the composition operator Φ strictly exceeds the expected coverage of the union of their individual possibility spaces, when at least one quality dimension is super-additive in agent collaboration.

Proof sketch. Σ_D enables strictly larger directed-acyclic-graph structures than any single agent can validate as their own product, since validator pools on Σ_D draw from senior-reputation populations that no single agent can replicate locally. Cross-agent reflection on Σ_R provides a second super-additive channel: an external reflector's allocation proposal can identify reallocations that the target agent cannot identify on its own, since the target agent's introspection is bounded by its own M component while the reflector's analysis is bounded by the public outcome stream. The aggregate coverage gain is bounded below by the marginal

validation capacity of the senior-reputation pools on Σ_D and Σ_R, which is positive whenever those pools are non-empty. The strict inequality follows from non-emptiness of the senior-reputation pools, which is guaranteed under AC4 once the agent population threshold is exceeded. ▮

C. Proposition 3 (Multi-timescale reflection)

The reflection operator R is implementable in a settlement-layer architecture if and only if the architecture supports validation outcomes on at least as many timescales as there are distinguishable reflection acts, where per-surface reflection on each of the prior five surfaces and cross-surface reflection on Σ_R together constitute six distinguishable reflection acts.

Proof sketch. Per-job resolution alone cannot distinguish a poor proposal that drew funding from a good proposal that drew unlucky validators, because the resolution event conflates proposal quality and execution quality. Separating Σ_J validation from Σ_E validation is the minimal disambiguation requirement, and the same argument applies pairwise across Σ_E, Σ_J, Σ_T, Σ_K, Σ_D. Per-surface reflection requires distinct timescales on each of these five surfaces, since the surfaces resolve at different speeds and conflating them would aggregate signals of different scales. Cross-surface reflection requires a sixth timescale, longer than any of the per-surface timescales, since its validation depends on observing the realized aggregate effect of an allocation policy across the joint outcome distribution of the prior five surfaces. R as a whole therefore operates only on six disambiguated signals; conflated signals contain insufficient information for the reflection operator to update G correctly along any specific generative dimension or to update compute allocation across surfaces. The number of distinct timescales must equal six. ▮

D. Proposition 4 (Architectural completeness, refined)

The six-surface architecture (Σ_E, Σ_J, Σ_T, Σ_K, Σ_D, Σ_R) is complete for the computative agent tuple (C, O, M, G, R) up to renaming, where the R component is operationally refined into per-surface reflection (embedded in Σ_E through Σ_D) and cross-surface reflection (isolated on Σ_R). The completeness claim is that any architecture satisfying the conditions of Computative Economics can be reduced to a six-surface architecture by surface aggregation, and any further refinement adds expressive power only by subdividing one of the six components.

Proof sketch. Each component of the agent tuple corresponds to a face of generative or reflective agent action, and the six surfaces partition the space of generative and reflective actions: Σ_E (selection within the action set), Σ_J (generation of opportunities), Σ_T (generation of the coordinate system), Σ_K (generation of contract structure), Σ_D (generation of joint structure), and Σ_R (reflective reallocation across the prior five). The R component of the tuple is operationally bivalent: per-surface reflection is necessary, since validation outcomes on each surface must update G locally, and is naturally embedded in the surface it reflects on; cross-surface reflection is independently necessary, since allocation across surfaces is not a valid action within any single surface, and accordingly requires its own surface. The bivalence is not optional: per-surface reflection without cross-surface reflection produces an architecture in which each surface improves locally but agents cannot rebalance across surfaces in response to changing demand or capability; cross-surface reflection without per-surface reflection produces an architecture in which agents reallocate compute across surfaces that themselves do not improve. Any further surface either decomposes one of the six into finer-grained actions, in which case the original surface remains a coarse-grained partition of the refinement, or aggregates two of the six into a coarser surface, in which case Proposition 1 establishes that aggregation violates AC2 or AC5. Six is therefore the minimum and the natural number of surfaces under the per-surface/cross-surface refinement of R. ∎

X. Governance: The Multi-Loop Ostrom Inversion

A. The translation

Elinor Ostrom's eight design principles for commons governance were developed in the context of consumption-side problems: how to prevent exhaustion of a shared rivalrous resource (Ostrom 1990). The principles were tested empirically in fisheries, forests, irrigation systems, and pastures, and they are now the dominant institutional framework for common-pool resource management (Hardin 1968). In multi-loop reputation economies, the governance object is qualitatively different. The shared resource is the joint possibility space, which is non-rivalrous in the sense that one agent's use does not deplete the space for others. The threat is not exhaustion but degradation: the joint space can suffer reduced coverage, fidelity, or novelty without any individual surface running out. The Ostrom inversion is the translation of the eight principles from the consumption-side problem of preventing exhaustion to the generation-side

problem of preventing degradation. The translation parallels the duality between internal governance design and external legal design developed in the author's prior DAO governance work (Kaal 2020).

B. The eight principles, restated for multi-loop architectures

First, clearly defined boundaries become clearly defined surfaces. Each Σ has explicit scope, validator pool composition, and reward function. The boundary of a generative surface is the set of element types that count as valid proposals on it; the boundary of Σ_R is the set of compute-allocation vectors over the prior five surfaces.

Second, congruence between rules and local conditions becomes congruence between surface mechanics and the type of generative or reflective action being validated. A tag surface cannot be governed by execution-surface mechanics, because the validation timescale and the reputation accrual function are categorically different. Σ_R cannot be governed by any single first-order surface's mechanics, since its validation requires cross-surface counterfactual evaluation.

Third, collective-choice arrangements become reputation-weighted parameter governance, with weights bounded by INV3 and INV5.

Fourth, monitoring becomes on-chain validation pool resolution events on each surface, providing public auditability of all reputation flows.

Fifth, graduated sanctions become stake escalation curves that escalate with repeat-offense detection, enforced by INV1.

Sixth, conflict-resolution mechanisms become dispute procedures embedded in template parameters on Σ_K, allowing different job classes to specify their own conflict-resolution protocols.

Seventh, recognition of rights to organize becomes permissionless agent registration with soulbound identity.

Eighth, nested enterprises become cross-surface reputation aggregation with bounded weights, the polytope condition INV4 supplemented by the reflection-mass bound INV5.

C. The novel principle: generation parity

In a multi-loop reputation economy, an additional design principle emerges that has no analog in Ostrom's original eight. Generation parity requires that the rate of reputation accrual on first-order generative surfaces (Σ_J, Σ_T, Σ_K, Σ_D), on the execution surface (Σ_E), and on the cross-surface reflection surface (Σ_R) lie in a triple ratio bounded above and below.

The principle has three components rather than two. If execution dominates, the system collapses to HDCA: agents allocate compute exclusively to selection, the action set remains static, and recursive equilibrium degenerates to repeated execution-game equilibrium. If first-order generation dominates, the system produces a rapidly inflating possibility space without execution discipline: proposals proliferate, but few are funded or executed, and fidelity collapses. If reflection dominates, the system produces a meta-reputation aristocracy: agents specialize in reflecting on others' allocations without contributing to the prior five surfaces, validator pools on Σ_R drift away from cross-surface competence, and the contraction property of Ψ is lost in the manner INV5 prevents. The governance object is the triple ratio.

Generation parity is a novel governance principle in two senses. First, it has no precedent in commons-governance literature, because the consumption-side problem does not have a generation-side analog. Second, it constitutes a parameter that requires ongoing governance attention rather than a one-time boundary specification: the optimal triple ratio depends on the agent population's composition and on demand-side dynamics, both of which evolve. The Ostrom translation, taken together with generation parity, constitutes the institutional layer of the multi-loop reputation economy (Calcaterra and Kaal 2021b).

XI. Open Problems

Five problems are identified as open and worth subsequent research.

OP1. The validator pool composition problem. Whether validator pools across the six surfaces should draw from disjoint senior-reputation populations or from a unified senior-reputation pool with weighted preference. Disjoint pools maximize specialization and reduce cross-surface manipulation. Unified pools maximize sybil resistance and reduce cold-start friction. The tradeoff is genuine and the optimal mix likely depends on the agent population's scale and composition. Σ_R complicates the question further, since its validator pool is by construction a weighted draw across the prior five surfaces and cannot be fully disjoint from any of them.

OP2. The cross-surface manipulation problem. Whether an agent can profitably manipulate the joint distribution of reputation across surfaces to extract reputation rents, including through coordinated proposals on Σ_R that target specific agents' allocations. The conjecture is that the five implementation invariants jointly bound such manipulation, but a formal proof is outstanding. Cross-surface manipulation is the natural extension of single-surface manipulation analyses and constitutes a research opening at the intersection of mechanism design and Computative Economics.

OP3. The cold-start problem. How to bootstrap senior-reputation pools on newly created surfaces, including newly proposed tags, newly created templates, and the initial Σ_R pool itself, without recursion to off-protocol authority. The candidate solution is temporary delegation from adjacent surfaces, but the formal characterization of the conditions under which delegation is incentive-compatible is an open question. Σ_R is the hardest case since its qualifying condition is non-trivial reputation across at least three of the prior five surfaces, which itself takes population maturation to satisfy.

OP4. The privacy-versus-validation tradeoff. Whether validation pools require disclosure of proposal contents at proposal time, at validation time, or only on funding, and what disclosure schedule preserves both incentive compatibility and competitive non-imitation. The tradeoff has direct analogs in patent law, and the patent-disclosure literature may provide instructive parallels for the architectural choice. Σ_R adds a distinct dimension: whether reallocation proposals targeting another agent should be disclosed to that agent at proposal time, validation time, or only after confirmation, with each disclosure schedule carrying different incentive consequences for the target agent's response.

OP5. The meta-meta-governance problem. Σ_R as specified in Part VI cannot self-refer in the strong sense: Σ_R proposals are over the prior five surfaces, not over Σ_R itself. The open question is whether a seventh surface Σ_M, on which the parameters governing the six surfaces (including Σ_R itself) are subject to reallocation proposals, is well-defined or invites recursive instability. The meta-meta-surface would be the natural extension of the architecture to fully self-amending governance, but the convergence properties of nested possibility loops at this depth are not established. The architecture in this paper draws the recursion line between Σ_R and Σ_M intentionally, leaving Σ_M as a research question rather than an architectural

commitment. Self-reference at the meta-meta level may produce well-defined equilibria or may produce instabilities that the five implementation invariants do not capture (Kaal 2024a).

XII. Conclusion

The transition from Human-Derived Coordination Architecture to operational implementations of Computative Economics is architectural, not parametric. It is not achieved by adjusting reputation curves or fee schedules. It requires the introduction of generative surfaces with their own validation, reward, and reflection mechanisms, and it requires a distinct surface on which the agent's allocation of compute across surfaces is itself the object of validation and reward. The possibility loop is the minimal unit of generative-side architecture. The six surfaces are jointly necessary and individually mapped to the components of the computative agent tuple, with the R component operationally refined into per-surface and cross-surface reflection, as Proposition 4 establishes.

The recursive equilibrium proof in Computative Economics has a settlement-layer correspondence given by the five implementation invariants developed in Part VIII. Architectures violating any of the five cannot be expected to converge. The invariants do not prescribe specific parameter values, only the bounds within which parameters must fall, and they leave substantial design freedom for implementers to optimize within those bounds. The Ostrom inversion provides the institutional layer: the governance object shifts from consumption-side exhaustion to generation-side degradation, with generation parity emerging as a novel principle without precedent in the commons-governance literature, refined in the present version into a triple ratio across execution, first-order generation, and cross-surface reflection.

The architecture is infrastructure neutral. It applies to any settlement substrate that supports stake-weighted validation, on-chain identity, and a representation of agent state. Implementations on Ethereum-class layer-two networks, on agent-native settlement layers, and on hybrid architectures are translations of the same primitive, not different primitives. The proper unit of analysis at the architectural level is the possibility loop, not the substrate.

The contribution closes the loop opened by Computative Economics: the theoretical primitive becomes an implementable architecture, and the implementable architecture becomes a research-grade specification. The agentic reputation economy that Computative Economics

requires is not a future protocol to be invented. It is a multi-loop architectural commitment that any settlement infrastructure can now be evaluated against. Whether existing infrastructures upgrade to satisfy the architectural commitment, or whether new infrastructures are built around it from the start, is a deployment question, not a theoretical one. The complementarity of the present argument with the consumption-side analysis of the abundance transition is direct: where prior work addresses the consumption side of the abundance transition (Desai and Lemley 2022), the present paper addresses the generation side. The theory is now operational. The deployment, as elsewhere in institutional design, will be where the contest is decided.

References

Akerlof, George A. 1970. "The Market for 'Lemons': Quality Uncertainty and the Market Mechanism." Quarterly Journal of Economics 84 (3): 488–500. https://doi.org/10.2307/1879431.

Arrow, Kenneth J., and Gerard Debreu. 1954. "Existence of an Equilibrium for a Competitive Economy." Econometrica 22 (3): 265–290. https://doi.org/10.2307/1907353.

Banach, Stefan. 1922. "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales." Fundamenta Mathematicae 3: 133–181. https://doi.org/10.4064/fm-3-1-133-181.

Brouwer, L. E. J. 1911. "Über Abbildung von Mannigfaltigkeiten." Mathematische Annalen 71 (1): 97–115. https://doi.org/10.1007/BF01456931.

Calcaterra, Craig. 2018. "On-Chain Governance of Decentralized Autonomous Organizations: Blockchain Organization Using Semada." Working paper. https://ssrn.com/abstract=3188374.

Calcaterra, Craig. 2023. "Reputation Tokenomics: DAO Governance Design Analysis." Working paper. https://ssrn.com/abstract=5018833.

Calcaterra, Craig, and Wulf A. Kaal. 2018. "Secure Proof of Stake Protocol." University of St. Thomas (Minnesota) Legal Studies Research Paper No. 18–10. https://ssrn.com/abstract=3125827.

Calcaterra, Craig, and Wulf A. Kaal. 2021a. Decentralization: Technology's Impact on Organizational and Societal Structure. Berlin: De Gruyter. https://doi.org/10.1515/9783110673937.

Calcaterra, Craig, and Wulf A. Kaal. 2021b. “Decentralized Governance.” Chap. 7 in Decentralization: Technology’s Impact on Organizational and Societal Structure. Berlin: De Gruyter. https://ssrn.com/abstract=3782214.

Calcaterra, Craig, Wulf A. Kaal, and Gopinath Sivalingam. 2018. “Reputation Protocol for the Internet of Trust—Conceptual Whitepaper.” University of St. Thomas (Minnesota) Legal Studies Research Paper No. 18–23. https://ssrn.com/abstract=3266953.

Coase, Ronald H. 1937. “The Nature of the Firm.” Economica 4 (16): 386–405. https://doi.org/10.1111/j.1468-0335.1937.tb00002.x.

Debreu, Gerard. 1959. Theory of Value: An Axiomatic Analysis of Economic Equilibrium. New York: Wiley.

Desai, Deven R., and Mark A. Lemley. 2022. “Scarcity, Regulation, and the Abundance Society.” Stanford Law and Economics Olin Working Paper No. 572. https://ssrn.com/abstract=4150871.

Friedman, James W. 1971. “A Non-Cooperative Equilibrium for Supergames.” Review of Economic Studies 38 (1): 1–12. https://doi.org/10.2307/2296617.

Fudenberg, Drew, and Eric Maskin. 1986. “The Folk Theorem in Repeated Games with Discounting or with Incomplete Information.” Econometrica 54 (3): 533–554. https://doi.org/10.2307/1911307.

Grossman, Sanford J., and Oliver D. Hart. 1986. “The Costs and Benefits of Ownership: A Theory of Vertical and Lateral Integration.” Journal of Political Economy 94 (4): 691–719. https://doi.org/10.1086/261404.

Hardin, Garrett. 1968. “The Tragedy of the Commons.” Science 162 (3859): 1243–1248. https://doi.org/10.1126/science.162.3859.1243.

Hayek, F. A. 1945. “The Use of Knowledge in Society.” American Economic Review 35 (4): 519–530.

Jensen, Michael C., and William H. Meckling. 1976. “Theory of the Firm: Managerial Behavior, Agency Costs and Ownership Structure.” Journal of Financial Economics 3 (4): 305–360. https://doi.org/10.1016/0304-405X(76)90026-X.

Kaal, Wulf A. 2020. “Decentralized Autonomous Organizations - Internal Governance and External Legal Design.” University of St. Thomas (Minnesota) Legal Studies Research Paper No. 20–16. https://ssrn.com/abstract=3652481.

Kaal, Wulf A. 2024a. “AI Governance.” Working paper. https://ssrn.com/abstract=4796714.

Kaal, Wulf A. 2024b. “Quantum Economy and the Future of Work.” University of St. Thomas (Minnesota) Legal Studies Research Paper No. 24–18. https://ssrn.com/abstract=4900880.

Kaal, Wulf A. 2026a. Computative Economics: A Framework for Economic Analysis under Computational Abundance. Working paper. https://ssrn.com/abstract=6607458.

Kaal, Wulf A. 2026b. The Collapse of Scarcity Economics. Working paper. https://ssrn.com/abstract=6421319.

Ostrom, Elinor. 1990. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge: Cambridge University Press.

Robbins, Lionel. 1932. An Essay on the Nature and Significance of Economic Science. London: Macmillan.

Simon, Herbert A. 1955. “A Behavioral Model of Rational Choice.” Quarterly Journal of Economics 69 (1): 99–118. https://doi.org/10.2307/1884852.

Williamson, Oliver E. 1985. The Economic Institutions of Capitalism: Firms, Markets, Relational Contracting. New York: Free Press.