Harmony from Coherence in Asynchronous MPST: A Minimal Erasure Invariant for Classical Dynamics

Sam Hart Berman

Abstract

We establish a Reconfiguration Harmony theorem for asynchronous MPST. Under well-formed `link` and delegation reconfiguration, projection commutes with reconfigured evolution. This gives a proof architecture for dynamic participant sets where topology change is a first-class semantic step. The inherited operational coherence invariant kernel is characterized in both directions and shown to admit an erasure characterization on active edges and to be the weakest admissible invariant for delegation safety (relative minimality). Safe delegation and composed-system conservation follow, including quantitative lift of *Computable Dynamics for Asynchronous MPST*'s Lyapunov framework with conservation for pure reconfiguration and descent or budget certificates for transition choreographies. The behavioral boundary is characterized by an exact determinism envelope with soundness, realizability, and maximality. Exchange-normalized determinism with spatial-subtyping monotonicity is established. Observational adequacy links abstract and protocol proofs to VM adherence modulo envelope. All results are mechanized in Lean 4.

1 Introduction

1.1 Background

Delegation and topology change are a primary failure point in MPST developments (Honda et al., 2008, and Tirore et al., 2025). Many systems exclude these operations, or admit them operationally without a theorem that connects choreography-level and local-level evolution.

In classical MPST, coherence primarily names global-type coherence and projectability conditions, with later refinements of projection criteria (Honda et al., 2008; Honda et al., 2016; Castagna et al., 2012; Majumdar et al., 2021). Logical lines also formulate coherence as n-ary duality or proof compatibility (Carbone et al., 2015; Carbone et al., 2016; Carbone et al., 2017), and data-certification extensions remain in the same global-projection discipline (Toninho and Yoshida, 2017). In this paper, as in *Coherence for Asynchronous Buffered MPST*, $\mathsf{Coherent}$ denotes the inherited operational invariant on local environments and buffered traces.

This manuscript is intended for companion review and publication with *Coherence for Asynchronous Buffered MPST* and *Computable Dynamics for Asynchronous MPST*. Cross-paper references in this text use companion-series dependencies.

1.2 Contribution

This paper addresses the reconfiguration gap with a commutation theorem, in the broader process-semantic tradition of operational correspondence and mobility (Hoare, 1985; Plotkin, 1981; Milner et al., 1992; Milner, 1999).

The central statement is Harmony under reconfiguration:

$$\mathsf{project} \circ \mathsf{step}_{\mathsf{reconfig}} = \mathsf{localStep}_{\mathsf{reconfig}} \circ \mathsf{project}. \tag{1.1}$$

for well-formed `link` and delegation operations with enabled post-reconfiguration steps. The theorem development proceeds from commutation to characterization and runtime adherence. It establishes erasure characterization of Coherence, safe delegation consequences, and relative minimality over admissible invariants. It then proves composed-system conservation, exact envelope characterization, exchange-normalized determinism with spatial monotonicity, and observational adequacy with VM adherence. The theorem order in this manuscript is dependency-driven rather than alphabetical.

The determinism-envelope layer follows a standard refinement-bound idea. One defines an admissible behavior relation that captures implementation freedom while preserving safety-visible observations. What is new here is an exact characterization for asynchronous MPST reconfiguration with soundness, realizability, and maximality proved in one theorem stack and connected directly to runtime admission and adherence evidence (Abadi and Lamport, 1991; Alpern and Schneider, 1985).

Equivalently, the envelope is a coarse-grained observational equivalence. It is the maximal admissible blur between executions that still preserves certified safety-visible meaning, in the spirit of noninterference and hyperproperty viewpoints (Goguen and Meseguer, 1982; Clarkson and Schneider, 2010).

Sharding correspondence follows a standard simulation view for distributed execution. A split execution should preserve the same safety-visible meaning as a reference execution under explicit compatibility assumptions (Lamport, 1978; Chandy and Lamport, 1985; Lynch, 1996). What is new here is an explicit envelope contract for local and cross-machine sharding profiles that makes the admissible difference class theorem-checkable and capability-gated at the VM bridge.

The envelope layer also admits a rate-distortion style reading. The reference semantics is the source, VM and sharded executions are channels, and admission profiles are distortion constraints on safety-visible observations (Shannon, 1948, and Cover and Thomas, 2006). Maximality then states that no strictly larger distortion class preserves the same certified safety guarantees.

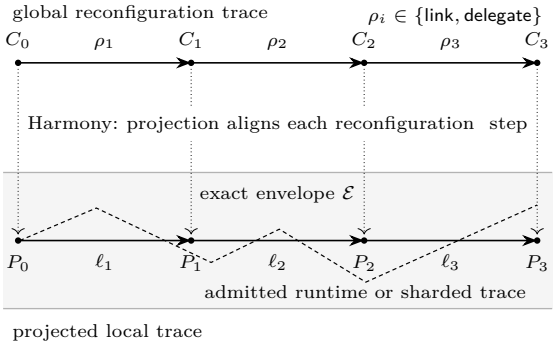

Figure 1.1: Dual-track trace view. Top track is the global reconfiguration trace $C_0 \to C_1 \to C_2 \to C_3$ labeled by ρ_i. Middle track is the projected local trace with corresponding labels ℓ_i. Bottom band is the admitted runtime trace corridor inside envelope $\mathcal{E}$. The figure intentionally scopes to $\rho \in \{\mathsf{link}, \mathsf{delegate}\}$; transition choreography is introduced later.

1.3 Scope

This paper closes the series architecture. *Coherence for Asynchronous Buffered MPST* supplies the invariant kernel. *Computable Dynamics for Asynchronous MPST* supplies bounds and decision procedures. This paper supplies reconfiguration commutation and envelope-level runtime adherence.

Scope is confined to asynchronous buffered semantics with explicit reconfiguration conditions, crash-stop fault assumptions where fault claims are stated, and declared envelope-admission profiles.

Result	**Imported Dependencies and Use**
Harmony	Imports the Paper 1 Coherence preservation and replacement stack, and Paper 2 computable-step premises, then proves projection commutation under link and delegation updates.
Minimality and Conservation	Imports the Paper 1 active-edge Coherence kernel, and Paper 2 transition-side quantitative premises, then fixes invariant minimality and conservation under declared reconfiguration premises.
Envelope family	Imports the Paper 1 quotient-facing Coherence boundary, and Paper 2 profile extraction interfaces, then lifts the safety core to envelope exactness, normalization, and VM adherence claims.
Byzantine family	Imports the Paper 2 Byzantine safety interface and converse packaging, then extends this interface to envelope-level maximality and adherence transport.

Table 1.1: Series dependency map for Paper 3.

In Table 1.1, Harmony corresponds to Theorem 4.1. Minimality and Conservation correspond to Theorems 7.1 and 8.1. Envelope family corresponds to Theorems 9.1, 10.1, and 11.1. Byzantine family corresponds to Theorem 11.5, Corollary 11.6, and Proposition 11.7.

Paper	Byzantine Layer Added
Paper 1	Interface obligations and bridge-ready Byzantine requirement classes.
Paper 2	Profile-level exact safety characterization plus converse witness packaging for dropped assumptions.
Paper 3	Reconfiguration-compatible exactness and capability-gated VM adherence/admission under the envelope boundary.

Table 1.2: Cross-paper Byzantine progression.

Our contributions are as follows:

1. A reconfiguration commutation theorem that covers both static composition and dynamic delegation.
2. An exact characterization of Coherence by active-edge erasure realizability and a relative-minimality theorem over admissible invariants.
3. A determinism-envelope theory with soundness, realizability, maximality, and exchange and spatial normalization consequences.
4. An abstract-to-VM adequacy bridge that ties profile claims to theorem-pack capability evidence.
5. A Byzantine envelope extension with exact characterization, converse counterexample families, and capability-gated VM adherence.

Main text proofs are proof sketches; full machine-checked derivations are provided by the Lean anchors indexed in Appendix E. Proof-sketch template used throughout the series: state proof mode (compositional/induction/coinduction/witness construction), give one concrete intermediate transformation, then conclude with the claim interface.

Symbol	Meaning
$\mathsf{H}_{\mathrm{byz}}$	Byzantine characterization bundle
$\mathsf{Obs}^{\mathrm{byz}}_{\mathrm{safe}}$	Byzantine safety-visible observation
$\mathsf{Eq}^{\mathrm{byz}}_{\mathrm{safe}}$	Byzantine safety-visible equality
$\mathcal{E}_{\mathrm{byz}}$	Byzantine determinism-envelope
Coherent	global active-edge coherence predicate
Harmony	reconfiguration commutation invariant
Admissible(I)	candidate invariant admissibility
W	weighted state function
$\mathcal{E}$	determinism-envelope trace set (admitted behaviors)
EnvelopeRel	envelope relation between VM and reference states
EnvelopeExact	exactness of envelope characterization
EnvelopeOK	envelope-admission predicate
$\mathsf{Cap}_{\mathcal{E}}$	capability set admitted by the envelope profile
ConfigEquiv	erasure-stability equivalence
Obs	observation map on executions/states
$C,\ C_i$	configuration state(s) used in reconfiguration traces
P_i	projected-local trace nodes in Figure 1.1
p, Π	runtime profile pair used in capability/adherence statements
ByzChar	Byzantine characterization formula
ByzSafe	Byzantine safety predicate
HasCaps(p, Π)	capability-evidence predicate for profile (p, Π)
VMAdheres	VM adherence predicate under admitted profile
N_s	explored state count in finite checker objects
N_e	explored transition-edge count in finite checker objects
$L_{\max}$	maximum active-edge buffered trace length
$W_{\max}$	maximum size of checked witness/certificate object
ℓ	projected local-step label for reconfiguration steps
ρ	reconfiguration step label (link/delegate/transition)

Table 1.3: Notation used in this paper.

2 Setup, Definitions, and Side Conditions

2.1 Base Recap

Assume the base model from *Coherence for Asynchronous Buffered MPST* and *Computable Dy-*

namics for Asynchronous MPST: asynchronous buffered steps, active-edge Coherence, fair scheduling assumptions stated per theorem, DeliveryModelparametricity, and crash-stop setting for fault claims.

Assumption	Status
async buffered semantics	required
active-edge Coherence	required
reconfig. conditions WF_ρ	required
fair scheduling profile	req. where stated
crash-stop fault model	req. for fault claims

Table 2.1: Model assumptions for exact claims in this paper.

Assumption Block 2.1. Core Model.
Core claims in this paper assume asynchronous buffered semantics, active-edge Coherence, reconfiguration side conditions through WF_ρ, and the stated fairness profile. Crash-stop premises apply where fault-side claims are stated.

Assumption-block inheritance policy. Assumption blocks do not inherit implicitly. If a theorem does not name a block, it uses Assumption Block 2.1 only.

Definition 2.1. Well-Typed Reconfiguration Configuration and Step.
Write $\Gamma \vdash C\ \mathsf{wf}$ for configuration typing and $\Gamma \vdash C \xrightarrow{\rho} C'$ for typed reconfiguration steps under the declared operator profile.

Role split. Well-typedness enforces operator and update side conditions for reconfiguration steps. Coherence enforces active-edge compatibility invariants transported through those steps.

2.2 Reconfiguration Operators

The paper distinguishes static composition (`link`) in deployment-level composition from dynamic delegation (endpoint/capability transfer during execution). Representative mechanized anchors are in the deployment-composition layer, the delegation-preservation layer, and the higher-order graph-delta layer. Appendix E provides the file-level mapping.

2.3 Dynamic Participant Sets

Participant sets are not fixed: participants may join through delegation, leave through crash-stop or transfer-and-exit, or change topology through composition.

The objective is to preserve Coherence/Harmony across these mutations.

2.4 Core Definitions

Write $\mathsf{envG}(C)$ and $\mathsf{envD}(C)$ for the global and delayed-trace environments extracted from configuration C, and Obs for the safety-visible observation map.

Definition 2.2. Lifted Coherence.

$$\mathsf{Coherent}(C) := \mathsf{Coherent}(\mathsf{envG}(C), \mathsf{envD}(C)). \tag{2.1}$$

Definition 2.3. Projection Operator.
$\mathsf{project}$ is the canonical projection from reconfiguration configurations to local-step configurations used by the Harmony square. The mechanized definition is `project` in `Runtime/Adequacy/EnvelopeCore/ReconfigurationBridge.lean`.

Definition 2.4. Harmony.
For reconfiguration operator ρ,

$$\begin{aligned}\mathsf{Harmony}(C,\rho) :\Longleftrightarrow\ & \mathsf{ProjStep}_\rho(C) \\ & = \mathsf{LocStep}_\rho(C).\end{aligned} \tag{2.2}$$

Here $\mathsf{ProjStep}_\rho(C)$ abbreviates $\mathsf{project}(\mathsf{step}_{\mathsf{reconfig}}(C,\rho))$, and $\mathsf{LocStep}_\rho(C)$ abbreviates $\mathsf{localStep}_{\mathsf{reconfig}}(\mathsf{project}(C),\rho)$.

Figure 2.1 gives the explicit commuting-square instance for Harmony.

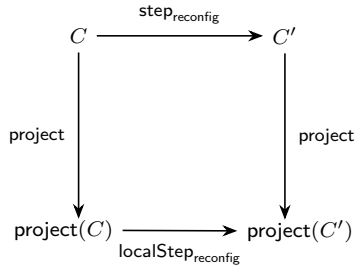

Figure 2.1: Commuting-square formulation of Harmony. Horizontal arrows are global reconfiguration and projected-local reconfiguration. Vertical arrows are projection. Commutation states that projecting after one reconfiguration step equals stepping in the projected local model.

Definition 2.5. Joint Realizability on Active Edges.
$\mathsf{JointRealizable}_{\mathsf{active}}(\mathsf{project}(C))$ holds when projected local views admit witnesses for all active edges.

Definition 2.6. Reconfiguration Well-Formedness.
Write $\mathsf{WF}_{\rho}(C)$ for the side-condition predicate used by reconfiguration theorems: `LinkWF(C,`ρ`)` for static linking, `DelegationWF(C,`ρ`)` for delegation, and `TransitionWF(C,`ρ`)` for transition choreographies.

Definition 2.7. Transition Well-Formedness Obligations.
$\mathsf{TransitionWF}(C, \rho_{\mathsf{tr}})$ holds when all of the following are discharged.

1. Structural typing and domain stability: source/target interfaces are well-typed, endpoint and edge domains are preserved on untouched components, and the declared transition footprint is well-formed.
2. Transition enabledness: the mode-transition operator is enabled at C under the declared operational side conditions.
3. Profile-premise package: $\mathsf{TrPrem}(C)$ is available, including scheduler/budget-side evidence used by transition descent arguments.
4. Trigger and guard discharge: declared transition trigger predicates and typed guard predicates are both satisfied.

Definition 2.8. TransitionWF Judgment Schema.
For transition choreography ρ_{tr}, define witness package

$$\omega_{\mathsf{tr}} := (\omega_{\mathsf{ty}}, \omega_{\mathsf{en}}, \omega_{\mathsf{pr}}, \omega_{\mathsf{gd}})$$

and judgment

$$\begin{aligned}\mathsf{TransitionWF}(C, \rho_{\mathsf{tr}}) : \Longleftrightarrow\ & \exists \omega_{\mathsf{tr}}.\, \mathsf{TyDomStable}(C, \rho_{\mathsf{tr}}; \omega_{\mathsf{ty}}) \\ & \wedge\ \mathsf{TrEnabled}(C, \rho_{\mathsf{tr}}; \omega_{\mathsf{en}}) \\ & \wedge\ \mathsf{TrPrem}(C; \omega_{\mathsf{pr}}) \\ & \wedge\ \mathsf{GuardedTr}(C, \rho_{\mathsf{tr}}; \omega_{\mathsf{gd}}).\end{aligned} \tag{2.3}$$

Items 1–4 of Definition 2.7 correspond to $\mathsf{TyDomStable}$, $\mathsf{TrEnabled}$, TrPrem, and $\mathsf{GuardedTr}$ respectively.

Obligation split. Items 1–3 in Definition 2.7 are structural proof obligations. Item 4 records application-level transition triggers together with typed admissibility guards.

Definition 2.9. Delegation Safety Predicates.

$\mathsf{SafeDelegation}(C, \rho_{\mathsf{del}})$ is the global delegation-safety judgment, and $\mathsf{SafeDelegationFootprint}(C, \rho_{\mathsf{del}})$ is its footprint-local form over edges touched by ρ_{del}.

Definition 2.10. $\mathsf{Admissible}(I)$**.**
$\mathsf{Admissible}(I)$ holds iff all of the following obligations are satisfied:

1. Locality: I is checkable from edge-local obligations on active edges.
2. Erasure stability: I is preserved under $\mathsf{ConfigEquiv}$classes.
3. Frame stability: untouched edges/endpoints preserve I under context extension.
4. Delegation-footprint adequacy: for each delegation step ρ_{del}, I entails the footprint-local update obligations used by Equation 2.4 together with the declared $\mathsf{WF}_{\rho_{\mathsf{del}}}$ side conditions.

Admissibility is defined by these obligations only. It does not assume $\mathsf{Coherent}$.

Definition 2.11. Envelope Predicates.
Write

$$\begin{aligned}\mathsf{EnvelopeRel}(S_{\mathrm{vm}}, S_{\mathrm{ref}}) := \mathsf{ObsEq}(S_{\mathrm{vm}}, S_{\mathrm{ref}}) \\ \wedge \mathsf{CertStepAlign}(S_{\mathrm{vm}}, S_{\mathrm{ref}}) \\ \wedge \mathsf{ProfileAlign}(S_{\mathrm{vm}}, S_{\mathrm{ref}}).\end{aligned}$$

ObsEq means safety-visible observations agree. CertStepAlign means every admitted VM step has the declared certified-step witness in the reference layer. ProfileAlign means both states satisfy the same profile-side admission obligations. $\mathsf{EnvelopeExact}(C)$ is exactness of the declared envelope (soundness and realizability and maximality) at configuration C. $\mathsf{EnvelopeOK}(C)$ means C satisfies the envelope-admission obligations used in Theorems 9.1–10.1.

Concrete example. Let S_{ref} be a reference execution prefix for the running example up to ρ_2. Let S_{vm} be a VM execution prefix with the same admitted external events, plus internal monitor bookkeeping steps. If the bookkeeping steps are silent at Obs, certified-step witnesses map each admitted VM step to the same reference prefix, and both prefixes satisfy the same profile obligations, then $\mathsf{EnvelopeRel}(S_{\mathrm{vm}}, S_{\mathrm{ref}})$ holds.

Construction recipe. To instantiate EnvelopeRel for a new protocol, first fix the safety-visible observation map Obs. Second, define the certified-step witness relation used by the admission checker. Third, define profile-side admission obligations and use their conjunction with observation equality and certified-step alignment.

Definition 2.12. Capability/Adherence Predicates.
$\mathsf{HasProfileCapabilities}(p, \Pi)$ means profile (p, Π) is backed by required theorem-pack capability evidence. $\mathsf{VMAdheres}(p, \Pi)$ means executions admitted under (p, Π) satisfy the corresponding adherence obligations.

Definition 2.13. Configuration Equivalence (ConfigEquiv).
ConfigEquiv is the least reflexive/symmetric/transitive closure of the base generator relation below, and is closed under context extension on untouched components (congruence at the frame boundary):

1. silent internal administrative steps that preserve safety-visible observations,
2. endpoint-preserving renamings on configuration identifiers,
3. frame-preserving bookkeeping rewrites on untouched components.

Cross-paper notation note. Paper 1 writes this same quotient relation as $\sim$. Canonical alias for coherent/admitted states is:

$$X \sim Y \iff X \ \mathsf{ConfigEquiv}\ Y.$$

Paper 1 uses $\sim$ for readability in early quotient equations, while this paper uses ConfigEquiv to align theorem and mechanization names.

Generator (paper)	Lean anchor/module mapping
silent admin steps	`config_equiv_iff_effect_bisim_silent` (`Runtime/Proofs/EffectBisim/ConfigEquivBridge.lea`
endpoint renamings	`Protocol/Coherence/ConfigEquivRenaming/*.lean`
frame bookkeeping rewrites	`Protocol/Coherence/ConfigEquiv.lean`

Table 2.2: Mapping from paper-level ConfigEquiv generators to mechanized anchors/modules.

Observation-preservation note. By construction, every generator in Definition 2.13 preserves the safety-visible interface Obs, so ConfigEquiv is an observational quotient.

Lemma 2.14. Generator Sanity for ConfigEquiv.
If $C \Rightarrow_{\mathsf{gen}} C'$ by one generator from Definition 2.13, then $\mathsf{Obs}(C) = \mathsf{Obs}(C')$.

Proof sketch. Proof mode: case analysis on generator family. Silent administrative steps are Obs-silent by definition, renamings preserve endpoint-level observations, and frame bookkeeping rewrites leave observed components untouched. Hence each generator preserves Obs. □

Definition 2.15. Normalization Relations.
ExchangeEq is the least equivalence relation generated by adjacent swaps of independent reconfiguration steps. A swap is admissible when two enabled steps ρ_a, ρ_b from the same pre-state C satisfy all of the following.

1. Footprints are disjoint:
   $\mathsf{Footprint}(\rho_a) \cap \mathsf{Footprint}(\rho_b) = \varnothing$.
2. Both orders are enabled from C under declared WF_ρ side conditions.
3. Diamond commutation holds at the update boundary:

$$C \xrightarrow{\rho_a} C_a \xrightarrow{\rho_b} C_{ab}, \qquad C \xrightarrow{\rho_b} C_b \xrightarrow{\rho_a} C_{ba},$$

   and C_{ab} ConfigEquiv C_{ba}.

Then $\mathsf{ExchangeEq}(C_1, C_2)$ holds when C_1 and C_2 are connected by a finite sequence of such admissible swaps plus ConfigEquivclosure.
$C \sqsubseteq_{\mathsf{sp}} C'$ is the spatial-subtyping preorder used in Theorem 10.1.

Relation inclusion and role split. Because ExchangeEq is closed under ConfigEquiv, we have

$$C_1 \text{ ConfigEquiv } C_2 \implies C_1 \text{ ExchangeEq } C_2.$$

ConfigEquiv is the observational quotient used by erasure and adequacy statements. ExchangeEq extends this with independent-step commutations used by exchange normalization.

Concrete exchange example. Suppose C has two enabled reconfiguration steps with disjoint footprints, one `link` step and one delegation step on a different session edge. If both orders are well-formed and produce ConfigEquivpost-states, then the two resulting configurations are exchange-equivalent.

Definition 2.16. Non-Interference Across Composition.
$\mathsf{NonInterf}(C_1, C_2)$ holds iff all of the following hold.

1. Footprint disjointness: boundary write footprints of enabled steps from C_1 and C_2 are disjoint.
2. Boundary commutation: if one enabled step from each component can fire, both execution orders are defined and preserve the same safety-visible observation.
3. Witness factorization: certified-step witnesses for composed traces factor into component witnesses without cross-boundary dependencies.

Checkability note. Under the same finite-reachability regime used in Paper 2, Clause 1 is decidable by finite footprint-set intersection. Clause 2 is decidable by bounded two-step exploration of enabled boundary transitions. Clause 3 is decidable when certified-step witnesses are represented by finite proof objects with total checkers. In that regime, checker/verifier complexity is $\mathrm{poly}(N_s + N_e)$ with witness-validation overhead $\mathrm{poly}(W_{\max})$. This is a validation bound, distinct from runtime transition-step cost in admitted executions.

Scope boundary. Outside finite-reachability or finite-witness encodings, this paper does not claim a total non-interference decider.

Equation 2.4 states the delegation-footprint update discipline.

$$\begin{aligned} C' &:= \mathsf{step}_{\mathsf{reconfig}}(C, \rho_{\mathsf{del}}), \\ F &:= \mathsf{Footprint}(\rho_{\mathsf{del}}), \\ D &:= \mathsf{envD}(C),\ D' := \mathsf{envD}(C'), \\ G &:= \mathsf{envG}(C),\ G' := \mathsf{envG}(C'), \\ \forall e \in F,&\quad D'(e) = \mathsf{rewrite}(D(e), \rho_{\mathsf{del}}), \\ \forall e \notin F,&\quad D'(e) = D(e), \\ &\quad G' = G \text{ on untouched endpoints.} \end{aligned} \tag{2.4}$$

3 Worked Example

We extend the baseline protocol through a sequence of reconfigurations, including federation, delegation, and mode transition, demonstrating how harmony, safety, and budget conservation compose at each step. The example carries the full structural theorem stack (Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, and 11.1) and shows that coherence-preserving reasoning from the first paper lifts through these dynamic changes.

Running example. Baseline choreography:

```
C → P : Request(n)
P → C : Grant(k)
C → M : Report(k)
M → P : Confirm
P → C : Token(t)
```

Extended system adds: `link` federation (`P1`, `P2`), delegation $C \to C'$, and optimistic-to-coordinated transition choreography.

Fix configurations and reconfiguration operators:

$$C_0 \xrightarrow{\rho_{\mathsf{link}}} C_1 \xrightarrow{\rho_{\mathsf{del}}} C_2 \xrightarrow{\rho_{\mathsf{tr}}} C_3. \tag{3.1}$$

Assume well-formedness judgments $\Gamma \vdash C_i$ wf for $i = 0, 1, 2, 3$. Here C_0 is the coherent single-pool state before federation, C_1 is the linked/federated state, C_2 is the post-delegation state where token capability is held by C', and C_3 is the coordinated-mode state after transition choreography.

The running obligations are used through these derived rule instances. Write $\mathsf{TrPrem}(C)$ for the transition-profile premise package at C, and $\mathsf{BDesc}(C, C')$ for a budget-certified descent witness from C to C'.

$$\begin{aligned} \mathsf{SD}(C,\rho) &:= \mathsf{SafeDelegation}(C,\rho);\\ \mathsf{SDF}(C,\rho) &:= \mathsf{SafeDelegationFootprint}(C,\rho);\\ \mathsf{EnvRel}^\star(C) &:= \mathsf{EnvelopeRel}(S_{\mathsf{vm}}(C), S_{\mathsf{ref}}(C));\\ \mathsf{ObsEq}^\star(C) &:= \mathsf{Obs}(S_{\mathsf{vm}}(C)) = \mathsf{Obs}(S_{\mathsf{ref}}(C)). \end{aligned} \tag{3.2}$$

Example derivation: link harmony.

$$\frac{\Gamma \vdash C_0 \ \mathsf{wf};\ \mathsf{WF}_{\rho_{\mathsf{link}}}(C_0)}{\mathsf{ProjStep}_{\rho_{\mathsf{link}}}(C_0) = \mathsf{LocStep}_{\rho_{\mathsf{link}}}(C_0)} \tag{3.3}$$

Example derivation: delegation safety.

$$\frac{\mathsf{Coherent}(C_1);\ \mathsf{DelegationWF}(C_1, \rho_{\mathsf{del}})}{\mathsf{SD}(C_1, \rho_{\mathsf{del}});\ \mathsf{SDF}(C_1, \rho_{\mathsf{del}})} \tag{3.4}$$

Example derivation: conservation.

$$\frac{\mathsf{WF}_{\rho_{\mathsf{link}}}(C_0);\ \mathsf{Coherent}(C_0)}{W(C_1) = W(C_0)} \tag{3.5}$$

Example derivation: transition.

$$\frac{\mathsf{WF}_{\rho_{\mathsf{tr}}}(C_2);\ \mathsf{Coherent}(C_2);\ \mathsf{TrPrem}(C_2)}{C_2 \to^* C_3;\ \mathsf{BDesc}(C_2, C_3)} \tag{3.6}$$

Example derivation: adequacy and adherence.

$$\frac{\mathsf{EnvRel}^\star(C_3);\ \mathsf{HasProfileCapabilities}(p, \Pi)}{\mathsf{ObsEq}^\star(C_3);\ \mathsf{VMAdheres}(p, \Pi)} \tag{3.7}$$

Table 3.1 gives the typed guard clauses used to discharge $\mathsf{WF}_{\rho_{\mathsf{tr}}}$ for the optimistic-to-coordinated step.

Guard	Source	Consequence
`conflict_detected`	app-level trigger	transition enabled
`reachable(Coord)`	crash decider	path preserved
$W \le$ budget	quant. bound	budget-feasible
$\lvert F\rvert \le f$	fault-budget decl.	bounded tolerance

Table 3.1: Typed guard clauses for transition well-formedness.

Guard-to-predicate link. The rows of Table 3.1 discharge Item 4 of Definition 2.7: `conflict_detected` is the application trigger, while `reachable(Coord)`, budget feasibility, and fault-budget bounds are typed/profile guards. At equation level, the same rows build the guard witness used in Equation 2.3:

$$\frac{\begin{array}{c}\mathsf{conflict\_detected}(C_2;\omega_{\mathsf{trg}}) \quad \mathsf{reachable}_{\mathsf{Coord}}(C_2;\omega_{\mathsf{reach}})\\ W(C_2) \le \mathsf{budget}(\omega_{\mathsf{budget}}) \quad |F(C_2)| \le f(\omega_{\mathsf{fault}})\end{array}}{\mathsf{GuardedTr}(C_2, \rho_{\mathsf{tr}}; \omega_{\mathsf{gd}})} \tag{3.8}$$

Combining (3.8) with $\omega_{\mathsf{ty}}, \omega_{\mathsf{en}}, \omega_{\mathsf{pr}}$ yields $\mathsf{TransitionWF}(C_2, \rho_{\mathsf{tr}})$ by Equation 2.3.

Concrete reading: C_0 pre-federation, C_1 linked, C_2 post-delegation handoff, C_3 coordinated mode with preserved envelope obligations.

4 Theorem B: Reconfiguration Harmony (Static + Dynamic)

Running-example thread. Section 3 instantiates this theorem on the sequence $C_0 \xrightarrow{\rho_{\mathsf{link}}} C_1 \xrightarrow{\rho_{\mathsf{del}}} C_2$.

Reading-order note. Theorem 4.1 is presented before Theorem 5.1 because later statements depend first on reconfiguration commutation and only then on erasure characterization. Appendix A Figure A.1 gives the full dependency graph for this ordering.

Assumption Block 4.1. Reconfiguration. The reconfiguration theorems assume typed global and local states, reconfiguration witnesses satisfying WF_ρ, enabled post-reconfiguration steps, and compatibility side conditions for unaffected edges.

Theorem 4.1. Reconfiguration Harmony. *For all configurations C and reconfiguration operators $\rho \in \{\mathsf{link}, \mathsf{delegate}\}$, if Assumption Block 4.1 holds for (C, ρ), then*

$$\mathsf{ProjStep}_\rho(C) = \mathsf{LocStep}_\rho(C). \tag{4.1}$$

Equivalently, the reconfiguration square commutes at the projection boundary.

Premise roles. In Theorem 4.1, typedness discharges WF_ρ obligations for `link`/`delegate`, while Coherence is the preserved safety invariant transported through commutation.

Running-example instantiation. For the example in Section 3, the link and delegation steps are exactly the two operators in this theorem.

Lemma 4.2. Delegation Projection Preservation Shape.
Under Assumption Block 4.1, let ρ_{del} be a well-formed delegation step from C with footprint F and post-state $C' = \mathsf{step}_{\mathsf{reconfig}}(C, \rho_{\mathsf{del}})$. If the update discipline of Equation 2.4 holds, then

$$\mathsf{project}(C') = \mathsf{localStep}_{\mathsf{reconfig}}(\mathsf{project}(C), \rho_{\mathsf{del}}). \tag{4.2}$$

The lemma packages the delegate case used in Theorem 4.1. On footprint edges $e \in F$, projection follows delegation update witnesses. On non-footprint edges $e \notin F$, projection follows frame preservation. These two clauses are exactly the commutation link from update discipline to projected-step equality.

The proof has two components: static Harmony via `link`, where composition-level commutation and coherence preservation are established in deployment theorems such as `link_harmony_through_link` and `link_preserves_coherent`; and dynamic Harmony via delegation, where delegation preservation and footprint lemmas establish commutation for topology-changing transfers. At the effect-observation boundary, the coinductive bridge supplies the composition-facing congruence and quotient lift used by this section: `effect_bisim_congr_link` and `config_equiv_iff_effect_bisim_silent`.

Side-condition necessity is explicit. Dropped-condition counterexample interfaces are part of the reconfiguration bridge layer.

Proof. Fix C and $\rho \in \{\mathsf{link}, \mathsf{delegate}\}$ satisfying Assumption Block 4.1. We prove

$$\mathsf{ProjStep}_\rho(C) = \mathsf{LocStep}_\rho(C)$$

by case analysis on ρ.

Case $\rho = \mathsf{link}$. From WF_ρ we obtain a well-formed composition object and enabled post-link step. Apply `link_preserves_coherent` to preserve coherence across the link update, then apply `link_harmony_through_link` to obtain commutation of projection with the link step:

$$\mathsf{project}(\mathsf{step}_{\mathsf{reconfig}}(C, \mathsf{link})) = \mathsf{localStep}_{\mathsf{reconfig}}(\mathsf{project}(C), \mathsf{link}).$$

This is exactly $\mathsf{ProjStep}_{\mathsf{link}}(C) = \mathsf{LocStep}_{\mathsf{link}}(C)$.

Case $\rho = \mathsf{delegate}$. Let $F := \mathsf{Footprint}(\rho_{\mathsf{del}})$ and write $C' := \mathsf{step}_{\mathsf{reconfig}}(C, \rho_{\mathsf{del}})$. Delegation well-formedness gives the side conditions required by the delegation-preservation layer. Apply Lemma 4.2 to the same $C, \rho_{\mathsf{del}}, F$, i.e. $\mathsf{ProjStep}_{\mathsf{delegate}}(C) = \mathsf{LocStep}_{\mathsf{delegate}}(C)$. In the running example, this is the $C_1 \to C_2$ delegation handoff.

Boundary transport. The above equalities are preserved at the observational/quotient interface by `effect_bisim_congr_link` and `config_equiv_iff_effect_bisim_silent`, so the same commutation statement holds at the composition-facing boundary used by later theorems.

Both cases are proved, therefore $\mathsf{ProjStep}_\rho(C) = \mathsf{LocStep}_\rho(C)$ for all $\rho \in \{\mathsf{link}, \mathsf{delegate}\}$ under the stated assumptions. □

Equation 4.3 separates static-link and dynamic-delegation commutation instances.
Commutation instances: link and delegate.

$$\begin{aligned} \mathsf{ProjLinkStep}(C) &= \mathsf{LocLinkStep}(C); \\ \mathsf{ProjDelegateStep}(C) &= \mathsf{LocDelegateStep}(C). \end{aligned} \tag{4.3}$$

5 Theorem A: Projection-Erasure Characterization of Coherence

Running-example thread. Section 3 provides the concrete projected traces used to read this equivalence operationally.

Erasure terminology. This paper uses three related terms. Projection erasure is the map $\mathsf{project}$. Class erasure is quotienting by $\mathsf{ConfigEquiv}$. Finite-state erasure transportability is the boundary property in Corollary 9.2.

Theorem 5.1. Projection-Erasure Characterization of Coherence.
Under Assumption Block 2.1, for all configurations C,

$$\mathsf{Coherent}(C) \iff \mathsf{JointRealizable}_{\mathsf{active}}(\mathsf{project}(C)). \tag{5.1}$$

That is, Coherence holds exactly when projected local views admit a compatible active-edge witness assignment.

Running-example instantiation. For $C_0, \ldots, C_3$ in Section 3, the theorem identifies Coherence with active-edge realizability of their projected local views.

This theorem gives the formal content of the quotient-first claim: erasure is not a heuristic post-processing step but the semantic object preserved by the theorem stack.

Proof sketch. The two directions are proved separately.

1. $\mathsf{Coherent} \to \mathsf{JointRealizable}_{\mathsf{active}}$:
   - from edge-local compatibility ($\mathsf{EdgeCoherent}$) build per-edge witnesses for projected local obligations,
   - combine witnesses over active edges to obtain joint realizability.
2. $\mathsf{JointRealizable}_{\mathsf{active}} \to \mathsf{Coherent}$:
   - invert witness assignment into edge-local consume compatibility,
   - reassemble edge obligations into global $\mathsf{Coherent}$.
3. Class-erasure stability:
   - use `coherent_erasure_stable` to show equivalence is preserved across configuration erasure classes.

Therefore Coherence is exactly characterized by active-edge realizability of projected views. □

6 Theorem C: Safe Delegation with Footprint Corollary

Running-example thread. Section 3 uses this theorem at the delegation step ρ_{del} from C_1 to C_2.

Theorem 6.1. Safe Delegation Sufficiency. *Under Assumption Block 4.1, for all configurations C and delegation operations ρ_{del},*

$$\mathsf{Coherent}(C) \wedge \mathsf{DelegationWF}(C, \rho_{\mathsf{del}}) \implies \mathsf{SafeDelegation}(C, \rho_{\mathsf{del}}). \tag{6.1}$$

Running-example instantiation. The delegation operation in Section 3 is a direct instance of this theorem's premise shape.

Corollary 6.2. Footprint Exactness Packaging. *Under Assumption Block 4.1, for all C, ρ_{del} satisfying the stated step and well-formedness side conditions,*

$$\mathsf{SafeDelegation}(C, \rho_{\mathsf{del}}) \iff \mathsf{SafeDelegationFootprint}(C, \rho_{\mathsf{del}}). \tag{6.2}$$

The converse direction is accompanied by strictness witnesses for dropped side conditions.

Proof sketch. Sufficiency follows by composing:

1. Coherence on active edges,
2. delegation well-formedness side conditions ($\mathsf{WF}_{\rho_{\mathsf{del}}}$),
3. delegation-preservation lemmas that update only the delegation footprint.

Together these imply safe delegation without introducing additional global invariants. □

Proof sketch. For the forward direction, apply `safe_delegation_to_footprint` to project $\mathsf{SafeDelegation}$ obligations to the declared delegation footprint. For the reverse direction, apply `footprint_to_safe_delegation` to reconstruct global safety from footprint obligations. Strictness witnesses from `safe_delegation_local_necessity` show necessity of the side conditions. The full equivalence is packaged as `safe_delegation_iff_footprint`. □

7 Theorem D: Relative Minimality

Running-example thread. Section 3 gives the concrete admissibility obligations that this minimality theorem abstracts.

Theorem 7.1. Relative Minimality. *Under Assumption Block 4.1, for all invariants I,*

$$\mathsf{Admissible}(I) \implies \forall C,\ I(C) \implies \mathsf{Coherent}(C). \tag{7.1}$$

Hence Coherence is the weakest admissible invariant that guarantees delegation safety under the stated model side conditions.

Running-example instantiation. For the running reconfiguration sequence, any candidate invariant

used to certify delegation safety must imply the same Coherence core.

This theorem prevents "stronger-than-needed" invariant inflation and pins down the architectural core. It is a characterization-style minimality theorem relative to the admissibility class above.

Proof sketch. Assume any invariant I satisfying $\mathsf{Admissible}(I)$.

1. Use admissibility clauses (locality, erasure stability, frame stability, delegation adequacy) to transport I along the same edge-local transformations used by Coherence.
2. Apply delegation-safety adequacy to derive local safety obligations on all active edges.
3. Convert these obligations into Coherence via erasure-characterization lemmas.

The full minimality statement is packaged as `relative_minimality`. Hence every admissible safety-guaranteeing invariant implies Coherence, making Coherence relatively minimal. □

Equation 7.2 summarizes the admissible-invariant order behind Theorem 7.1. The display below separates the invariant implication layer from its delegation safety consequence so the proof flow remains explicit.

$$\begin{array}{c} \mathsf{Admissible}(I) \\ \Downarrow \\ \forall C,\ I(C) \Rightarrow \mathsf{Coherent}(C) \\ \Downarrow \\ \text{delegation safety consequences.} \end{array} \tag{7.2}$$

8 Theorem E: Composed-System Conservation

Running-example thread. Section 3 instantiates the pure-link and delegation conservation path before transition choreography descent.

Theorem 8.1. Composed-System Conservation.
Under Assumption Block 4.1, for all configurations C and reconfiguration operators ρ, if $\mathsf{WF}_\rho(C)$ and $\mathsf{Coherent}(C)$ hold, then

$$\mathsf{Coherent}(\mathsf{step}_{\mathsf{reconfig}}(C,\rho)). \tag{8.1}$$

For all configurations C and pure reconfiguration operators ρ,

$$\begin{aligned} &\mathsf{WF}_\rho(C) \wedge \mathsf{Coherent}(C) \\ \Longrightarrow\quad & W(\mathsf{step}_{\mathsf{reconfig}}(C,\rho)) = W(C). \end{aligned} \tag{8.2}$$

For transition choreographies ρ_{tr} with stated budget and scheduler premises (fairness-required), descent and budget certificates are preserved with conservative profile-dependent bounds.

Fairness-required clause. The pure-reconfiguration conservation statements are fairness-independent. The transition-side bounded-dissipation clause uses the declared scheduler/-fairness premises imported with the quantitative profile.

Running-example instantiation. In Section 3, $C_0 \to C_1 \to C_2$ is the conservative branch and $C_2 \to C_3$ is the budgeted transition branch.

The theorem separates conservation and dissipation roles. Pure reconfiguration preserves the weighted quantity exactly. Transition choreographies consume certified progress budget and are therefore dissipative with explicit bounds. This split keeps structural rewiring distinct from progress-consuming work.

Under composition and delegation:

1. Coherence and Harmony are preserved.
2. Quantitative lift from *Computable Dynamics for Asynchronous MPST* holds:
   - pure reconfiguration conserves the weighted measure,
   - transition choreographies are governed by descent and budget certificates.

This theorem combines composition, delegation, and quantitative invariants in one package.

Proof sketch. Split reconfiguration into conservative and budgeted classes.

1. Pure reconfiguration (`link`/delegation):
   - preserve Coherence/Harmony by Theorem 4.1-side machinery,
   - show weighted measure W is invariant under balanced delegation/rewiring (`lyapunov_conserved_under_balanced_delegation`).
2. Transition choreographies:

- apply inherited quantitative descent/budget certificates from the dynamics layer,
- conclude bounded dissipation under declared scheduler premises.

3. Compose both cases via `flagship_composed_system_conservation` to obtain theorem statement and conservative profile-dependent quantitative side.

□

Corollary 8.2. Compositional Exactness Under Non-Interference.
For all compositions $C_1 \otimes C_2$, if Assumption Block 9.1 holds for both components and $\mathsf{NonInterf}(C_1, C_2)$ *holds, then envelope exactness is preserved by composition:*

$$\mathsf{EnvelopeExact}(C_1) \wedge \mathsf{EnvelopeExact}(C_2) \implies \mathsf{EnvelopeExact}(C_1 \otimes C_2). \tag{8.3}$$

Strictness witnesses are provided for dropped non-interference premises.

Proof sketch. $\mathsf{NonInterf}(C_1, C_2)$ gives a check criterion through footprint disjointness and boundary commutation, and a semantic criterion through witness factorization. These clauses factor composed traces into component traces with no cross-boundary interference steps. Apply envelope exactness to each component and recompose to obtain exactness of the composition. For strictness, use the dropped-assumption witness family to exhibit an interfering trace that violates factorization and therefore exactness. □

Interference example. If both components perform delegation rewrites that target the same boundary edge in one macro-step window, footprint disjointness fails. The two execution orders can then induce different certified boundary traces, so composed exactness can fail.

Equation 8.4 summarizes the quantitative split. The next display separates pure conservation from transition-side descent to match the two proof branches in Theorem 8.1. Write $S_\rho(C) := \mathsf{step}_{\mathsf{reconfig}}(C, \rho)$.

$$\begin{aligned} \mathsf{pure} &\Rightarrow W(S_\rho(C)) = W(C), \\ \mathsf{trans} &\Rightarrow \mathsf{descent\_budget}. \end{aligned} \tag{8.4}$$

Here pure abbreviates pure reconfiguration, and trans abbreviates transition choreography with inherited descent/budget certification under the stated profile assumptions.

9 Theorem F: Exact Characterization of Determinism Envelope

Running-example thread. Section 3 supplies the admitted execution prefixes used to read the envelope clauses.

Assumption Block 9.1. Envelope.
Envelope theorems assume the declared admissibility profile with trace well-formedness, certified-step obligations, and the realizability witness schema used by the admission checker.

Theorem 9.1. Exact Determinism Envelope.
Under Assumption Block 9.1, there exists an envelope relation $\mathsf{EnvelopeRel}$ *and an induced admitted trace set $\mathcal{E}$ such that:*

1. *Soundness:* $\forall t,\ \mathsf{Certified}(t) \implies t \in \mathcal{E}$.
2. *Realizability:* $\forall t \in \mathcal{E},\ \exists e,\ \mathsf{Trace}(e) = t$.
3. *Maximality: $\forall \mathcal{E}'$, if $\mathcal{E}' \supsetneq \mathcal{E}$, then $\exists t \in \mathcal{E}' \setminus \mathcal{E}$ that violates admissibility constraints.*

Running-example instantiation. The admitted traces around C_3 in Section 3 are the concrete profiles for soundness and realizability checks.

The envelope claim is exact under the stated model and assumption bundle:

1. soundness: certified executions lie within the envelope.
2. realizability and completeness: admitted envelope behaviors are witnessed.
3. maximality: strict supersets are rejected by witness counterexamples.

Conceptually, this is dual to Theorem 7.1:

- D: weakest admissible invariant core,
- F: maximal admissible behavior envelope.

Proof sketch. Exactness is obtained by packaging three envelope properties.

1. Soundness:

- derive Certified → Envelope from local/sharded envelope soundness definitions and profile extraction.

2. Realizability/completeness:
   - use realizability witness schemas to construct executions for admitted envelope behaviors.
3. Maximality:
   - show any strict envelope extension violates admissibility via explicit witness obligations.

Core formal structures come from envelope foundations (`LocalExactEnvelopeCharacterization`, `ShardedExactEnvelopeCharacterization`) and profile-level extraction theorems (`consensus_envelope_exact_of_profile`, `failure_envelope_maximality_of_profile`). Method split. Soundness/maximality steps are compositional consequence packaging over certified-step obligations, while realizability is a witness construction argument that builds executions from admitted traces. □

Corollary 9.2. Finite-State Erasure-Transportability Boundary.
Under Assumption Block 9.1,

$$\forall t,\ \mathsf{FiniteErasureTransportable}(t) \iff \mathsf{RationalFiniteState}(t). \tag{9.1}$$

Proof sketch. Use Theorem 9.1 to identify exactly admissible envelope traces. Then apply the finite-state/rational witness bridge in the same profile to show bi-implication between finite erasure transportability and rational finite-state representability. This is a compositional combination of exactness plus witness equivalence, not an additional fixed-point proof. □

Corollary 9.3. Strict Boundary Witness.
Under Assumption Block 9.1, there exists t such that

$$\neg\mathsf{RationalFiniteState}(t) \wedge \neg\mathsf{FiniteErasureTransportable}(t). \tag{9.2}$$

Proof sketch. From maximality in Theorem 9.1, any strict extension beyond the admitted class must violate an admissibility clause. Instantiate that violation with the strict-boundary witness family to obtain t outside rational finite-state transportability. This is a witness-construction argument. □

Corollary 9.4. Inductive-Embedding Exactness.
Under Assumption Block 9.1, for all inductive states x, exact envelope characterization is preserved under `toCoind`:

$$\mathsf{EnvelopeExact}_{\mathrm{ind}}(x) \iff \mathsf{EnvelopeExact}_{\mathrm{coind}}(\mathsf{toCoind}(x)). \tag{9.3}$$

Proof sketch. Apply envelope exactness on the inductive representation, then transport along `toCoind` and its inverse correspondence laws. The proof is compositional: representation conversion preserves the same admission/equality obligations on both sides, so exactness is equivalent after embedding. □

Appendices A and C provide the detailed witness constructions and transport lemmas for these boundary corollaries.

10 Theorem G: Exchange-Normalized Determinism and Spatial Monotonicity

Running-example thread. Section 3 gives independent reconfiguration steps whose admissible swaps are quotiented by this theorem.

Theorem 10.1. Exchange-Normalized Determinism and Spatial Monotonicity.
Under Assumption Block 9.1, for all configurations, the following properties hold:

1. Exchange normalization:

$$\mathsf{ExchangeEq}(C_1, C_2) \implies \mathsf{Obs}(C_1) = \mathsf{Obs}(C_2). \tag{10.1}$$

2. Spatial monotonicity:

$$C \sqsubseteq_{\mathsf{sp}} C' \wedge \mathsf{EnvelopeOK}(C) \implies \mathsf{EnvelopeOK}(C'). \tag{10.2}$$

Running-example instantiation. In Section 3, swapping independent certified updates while preserving observation is an exchange-normalization instance.

Theorem 10.1 combines two normalization properties.

1. Exchange normalization: admissible reorderings collapse to equivalent safety-visible outcomes.
2. Spatial monotonicity: envelope and safety obligations are preserved under admissible spatial refinement.

This is a symmetry-reduction statement. Different concrete traces can represent the same observable behavior class once exchange symmetries are quotiented. The theorem makes that collapse explicit and checkable at the envelope layer.

This theorem sharpens what "determinism modulo envelope" means operationally.

Proof sketch. Instantiate reconfiguration-bridge theorem forms with exchange and spatial premises.

1. Exchange normalization follows from bridge theorem `exchange_normalization_of_e1_bridge_premises`.
2. Spatial monotonicity follows from `spatial_subtyping_monotonicity_of_e1_bridge_premises`.
3. Combine both to obtain determinism modulo envelope classes rather than raw traces.

□

11 Theorem H: Observational Adequacy and VM Adherence Modulo Envelope

Running-example thread. Section 3 provides the VM/reference paired prefixes used by adequacy and adherence.

Assumption Block 11.1. Adequacy. Adequacy and adherence theorems assume envelope-related VM and reference states, trace-consistency premises for observable events, and profile extraction rules for capability reports.

Theorem 11.1. Observational Adequacy and VM Adherence Modulo Envelope. ***(Adequacy).*** *Assume Assumption Block 11.1. Let states S_v, S_r satisfy*

$$\mathsf{EnvelopeRel}(S_v, S_r). \tag{11.1}$$

Then

$$\mathsf{Obs}(S_v) = \mathsf{Obs}(S_r). \tag{11.2}$$

(Adherence). *For all profiles (p, Π), if the profile-side premises of Assumption Block 11.1 hold and the capability premise below holds,*

$$\mathsf{HasProfileCapabilities}(p, \Pi). \tag{11.3}$$

then

$$\mathsf{VMAdheres}(p, \Pi). \tag{11.4}$$

Local and sharded variants are recovered as profile-specific instances under explicit collapse assumptions.

Proof sketch. The adequacy/adherence statement is obtained by composition of three bridges.

1. Protocol outcome bridge:
   - connect observational equality and effect-bisim (`protocol_outcome_effect_bisim_of_observational_eq`).
2. Compile/runtime refinement bridge:
   - lift effect-bisim to compile-time observational refinement (`compile_refines_observational_eq_of_effect_bisim`).
3. VM view bridge:
   - connect VM configuration equivalence and effect-bisim (`vm_view_effect_bisim_of_vmc_equiv`, `vm_c_equiv_of_vm_view_effect_bisim`, `topology_change_preserves_vmc_equiv_via_effect_bisim`).

Adequacy follows from trace-consistency theorem (`vm_adequacy_of_trace_consistency`). Adherence follows from profile-capability extraction and theorem-pack gating. Method split. The bridge composition itself is compositional. The trace-consistency discharge uses the existing finite-trace induction packaged in `vm_adequacy_of_trace_consistency`. □

Running-example instantiation. The adequacy derivation for C_3 in Section 3 is a direct instance of this theorem.

Configuration-equivalence link. The quotient used by this section is ConfigEquiv, connected to effect-bisimulation through `config_equiv_iff_effect_bisim_silent`. This is why adequacy and adherence transport can be stated at the same observational boundary.

VM instruction-layer note. This section reuses the instruction-form interface and `WellTypedInstr` mapping from the Effect-Typed Bridge in *Coherence for Asynchronous Buffered MPST* (Section 7, Proposition 7.1). At this interface, `send`/`select` carry sender-continuation obligations, `recv`/`branch` carry consume-alignment obligations, and `invoke` carries handler-fragment lift obligations. The novelty here is envelope-indexed adequacy and adherence transport over that shared VM typing layer.

Proposition 11.2. Capability-Bit Soundness. *Under Assumption Block 11.1, for all runtime profiles q and capability bits b, if q advertises b, then there exists a corresponding theorem-pack object a_b such that a_b satisfies the same envelope-premise profile used by Theorem 11.1.*

Proof sketch. Proof mode: witness extraction. From capability extraction, each advertised bit b yields an intermediate typed gate fact $\mathsf{cap\_entry\_typed}(q, b)$ validated by theorem-pack checking. Soundness of extraction maps this fact to witness object a_b with matching profile premises. Therefore each advertised capability is justified by a corresponding theorem-pack witness under Assumption Block 11.1. □

Corollary 11.3. Principal Capability and Admission Completeness. *Under Assumption Block 11.1, let D_{user} be a requested envelope capability and let D_{prog} be inferred from program and deployment profile. We have*

$$\mathsf{Cap}_{\mathcal{E}} := \{\, d \mid \mathsf{CapOK}_{\mathsf{EnvelopeRel}}(d) \,\},$$

where $\mathsf{CapOK}_{\mathsf{EnvelopeRel}}(d)$ means that d is admitted by the envelope profile induced by EnvelopeRel*.*

$$D_{user} \subseteq D_{prog} \subseteq \mathsf{Cap}_{\mathcal{E}}. \tag{11.5}$$

Admission is complete with a principal inferred capability.

Assumption Block 11.2. Byzantine. Byzantine characterization assumes explicit bundle $\mathsf{H}_{\mathrm{byz}}$with fault-model, authentication and evidence-validity, conflict-exclusion primitive consistency, and adversarial-budget side conditions.

Shared Byzantine notation block (series baseline). This section reuses ByzChar, ByzSafe, $\mathsf{Eq}^{\mathrm{byz}}_{\mathrm{safe}}$, and $\mathcal{E}_{\mathrm{byz}}$from Papers 1–2 and adds envelope adherence/admission integration at the A–H boundary.

Definition 11.4. Byzantine Bundle Component Meanings.
At the envelope interface in this paper:

1. ByzQuorumOK: quorum and intersection constraints hold for the declared fault budget. The standard threshold instantiation is $n > 3f$ with quorum intersection witnesses.
2. ByzAuthEvidenceOK: accepted Byzantine-relevant actions carry valid authentication and evidence for envelope admission.
3. ByzBudgetOK: adversarial actions satisfy the declared budget envelope used by profile admission.
4. ByzPrimitiveConsistent: primitive-level conflict exclusion and consistency obligations hold for certified transitions.

ByzSafe is the operational safety predicate at this interface. It means admitted executions preserve non-conflicting safety-visible outcomes in $\mathsf{Obs}^{\mathrm{byz}}_{\mathrm{safe}}$modulo $\mathcal{E}_{\mathrm{byz}}$.

Layering note. Paper 2 establishes profile-level exact safety characterization for the static and decision layer. The theorem below lifts that characterization to the reconfiguration and VM-adherence boundary used in this paper.

Define

$$\begin{aligned} \mathsf{ByzChar} := {} & \mathsf{ByzQuorumOK} \\ & \wedge\ \mathsf{ByzAuthEvidenceOK} \\ & \wedge\ \mathsf{ByzBudgetOK} \\ & \wedge\ \mathsf{ByzPrimitiveConsistent}. \end{aligned} \tag{11.6}$$

where each conjunct names the corresponding requirement class in Assumption Block 11.2.

New in this paper: Byzantine characterization is lifted to the envelope and VM adherence boundary, so the exact safety interface composes directly with Theorem 11.1.

Theorem 11.5. Exact Byzantine Safety Characterization. *Under Assumption Block 11.2, profile extraction*

yields an exact-characterization package for the declared Byzantine model. At this paper interface, the package gives

$$\mathsf{ByzChar} \iff \mathsf{ByzSafe}. \tag{11.7}$$

with relative maximality for the same characterization relation.

Proof sketch. First instantiate the profile package under Assumption Block 11.2. Second, extract soundness, completeness, and maximality components from `byzantine_safety_exact_of_profile`. Third, reinterpret those components through the same safety-visible interface used by Theorem 11.1. This yields the exact iff characterization. The argument is compositional package projection rather than a fresh induction. □

Corollary 11.6. Converse Counterexample Families.
If any required class in Assumption Block 11.2 is dropped, there exists a counterexample family violating ByzSafe. *Covered dropped classes are quorum or intersection obligations, authentication or evidence-validity obligations, adversarial-budget obligations, and primitive-consistency obligations.*

Proposition 11.7. Byzantine VM Adherence and Admission Gating.
Under Assumption Blocks 11.1 and 11.2, for runtime profile (p, Π), if theorem-pack capability bits include Byzantine safety characterization and VM envelope adherence evidence, then admitted executions satisfy $\mathsf{Eq}^{\mathrm{byz}}_{\mathrm{safe}}$ modulo $\mathcal{E}_{\mathrm{byz}}$ and rejected requests correspond to failed assumptions or missing evidence.

12 Supporting Formal Layer

Three support packages make the main theorem stack executable and compositional.

1. Higher-order extension: channel-carrying payloads and graph-delta semantics.
2. Conservative extension: collapse back to first-order when no channel payloads are present.
3. Delegation-preservation microkernel: redirected, unrelated, and other-session edge coherence lemmas.

Concrete modules for these packages:

1. Higher-order/graph-delta semantics: `Protocol/Coherence/GraphDelta.lean`, `Protocol/Coherence/GraphDeltaHO.lean`.
2. First-order collapse and equivalence/renaming support: `Protocol/Coherence/ConfigEquiv.lean`, `Protocol/Coherence/ConfigEquivRenaming/*.lean`, `Protocol/Coherence/RoleSwap/*.lean`.
3. Delegation microkernel: `Protocol/Coherence/Delegation/Core/*.lean`, `Protocol/Coherence/Delegation/Preservation.lean`.

Channel-payload cross-reference. The formal channel-payload objects and rewrite rules are the definitions in `GraphDelta.lean` and `GraphDeltaHO.lean`. The anchor mapping is listed in Appendix E.

Definition 12.1. Payload-Carrying Edge.
An active edge $e = (u, v)$ is payload-carrying when the message payload on e includes transferable channel/capability references (not only first-order data values). In that case, one-step semantics includes both control-trace update and capability-graph update obligations.

Higher-order Coherence lift. For a payload-carrying edge, coherence requires two checks. The control trace must satisfy the same `Consume`-style alignment obligation as the first-order case. The transferred payload-channel capability graph must satisfy the graph-delta compatibility obligations used by the delegation-preservation layer. Projected local types therefore mention payload channels only through admitted capability transfers. If no channel payloads are present, the payload clause is vacuous and the theorem statements collapse to the first-order form.

Graph-delta example. Suppose a delegation message on edge (C, C') transfers a capability token for channel χ. The graph-delta obligations require (i) removal of χ from the sender-side capability map, (ii) insertion at the receiver side, and (iii) preservation of declared disjointness/ownership invariants on unrelated edges. Control alignment without this capability rewrite is insufficient for higher-order coherence.

Mini-trace (end-to-end). A payload transfer with graph-delta rewrite is:

$$C_2 \xrightarrow{\rho_{\mathsf{del}}[\chi]} C_2'$$

$$\Gamma^{C}_{\mathsf{cap}}(\chi) = 1,\ \Gamma^{C'}_{\mathsf{cap}}(\chi) = 0 \xRightarrow{\Delta_{\mathsf{graph}}} \Gamma^{C}_{\mathsf{cap}}(\chi) = 0,\ \Gamma^{C'}_{\mathsf{cap}}(\chi) = 1$$

$$\mathsf{Obs}(C_2') = \mathsf{Obs}(C_2) \text{ on unrelated edges}$$

$$C_2' \xrightarrow{\rho_{\mathsf{tr}}} C_3.$$

Failure if graph-delta obligations are dropped: if sender-side removal is not enforced, both endpoints can retain χ, producing duplicated capability ownership and violating delegation-preservation invariants; such steps are not admitted by the higher-order coherence checker.

When to use which layer. First-order coherence is sufficient when payloads are data-only and no channel/capability references are transferred. Higher-order coherence is required when delegation or migration carries channel capabilities whose ownership/footprint must be rewritten soundly.

13 Worked Transport in Main Body

This section formalizes two transport instances from the generic schema in Appendix B. Connection to the theorem stack. Theorem 11.1 provides the observational/adherence boundary that lets structural executions be identified with analytic observables. The transport statements below are therefore stack-enabled applications: they use the structural interface discharged by Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, and 11.1 plus extra analytic package premises. They are not corollaries of Theorem 11.1 alone, because each transport bound additionally requires an external analytic package.

Assumption Block 13.1. Transport.
For each target claim, assume:

1. the structural theorem interface required by Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, and 11.1 at the relevant profile,
2. an external analytic package for the target claim,
3. interface compatibility between structural observables and analytic variables.

Definition 13.1. Transport Package Well-Formedness.
$\mathsf{TailPkgWF}(\theta)$ holds iff the transport interface verifies: (i) target-signature compatibility for $\Psi_{\mathrm{tail}}(\cdot;\theta)$, (ii) observable-domain compatibility with transported variable T_{comp}, and (iii) checker-side certificate object well-typing. $\mathsf{MixPkgWF}(\eta)$ is defined analogously for $\Psi_{\mathrm{mix}}(\cdot;\eta)$ and transported law sequence $(\mu_n)_n$. Write

$$\mathrm{TailPkg}(\theta) := \mathsf{TailPkgWF}(\theta) \wedge \mathsf{TailAnalyticCert}(\theta),$$

$$\mathrm{MixPkg}(\eta) := \mathsf{MixPkgWF}(\eta) \wedge \mathsf{MixAnalyticCert}(\eta),$$

where $\mathsf{TailAnalyticCert}$ and $\mathsf{MixAnalyticCert}$ are external analytic certificates not derived by structural protocol proofs.

Package existence guidance. Typical sufficient conditions are as follows. For a tail package, one may use bounded-increment or bounded-moment assumptions that yield concentration bounds for the transported observable. For a mixing package, one may use a contractive/ergodic hypothesis (for example, geometric contraction in the chosen metric) that yields an explicit rate function. These are analytic assumptions external to the structural theorem stack.

Validation boundary. The transport interface validates structural compatibility and the declared package shape (target function/signature compatibility), but it does not derive analytic inequalities from protocol structure alone. Analytic package obligations are consumed as external proved certificates.

Non-example. If the transported observable has heavy tails with no admissible concentration inequality in the declared family, or if the induced process is non-mixing (for example periodic without contraction), then no admissible TailPkg or MixPkg exists for those targets.

Proposition 13.2. Transport Package Validation Sufficiency.
Under Assumption Block 13.1, if $\mathsf{TailPkgWF}(\theta)$ *(resp.* $\mathsf{MixPkgWF}(\eta)$*) then the transport interface can safely consume the package: shape/domain checks and witness typing checks succeed, and all remaining obligations are exactly the external analytic certificate clauses* $\mathsf{TailAnalyticCert}(\theta)$ *(resp.* $\mathsf{MixAnalyticCert}(\eta)$*).*

Proof sketch. Proof mode: witness construction. Package well-formedness provides concrete interface witnesses for signature, domain, and typing compatibility. The interface checker consumes

these witnesses and reconstructs the structural substitution used by transport theorems. No additional structural premise is needed; only the analytic inequality clause remains external by definition. □

Fix the source execution family to the running-example transition slice from Section 3:

$$C_2 \xrightarrow{\rho_{\mathrm{tr}}} C_3 \to \cdots . \tag{13.1}$$

and let transported observables be computed from the safety-visible projection Obs. Concretely for this running slice, we use

$$\begin{aligned} X_n &:= \mathsf{Obs}(C_n), \\ T_{\mathrm{comp}} &:= \inf\{n \mid \mathsf{Complete}(C_n)\}, \\ \mu_n &:= \mathcal{L}(X_n), \end{aligned} \tag{13.2}$$

where $\mathsf{Complete}(C_n)$ abbreviates "C_n satisfies the coordinated-mode completion guard," and μ_n is the transported law at step n.

Concrete package instances. One admissible tail-package instance is a sub-gamma style bound

$$\Psi_{\mathrm{tail}}(t;\sigma,\nu) = \exp\Big(-\frac{t^2}{2(\nu+\sigma t)}\Big), \quad t \geq 0.$$

One admissible mixing-package instance is geometric decay

$$\Psi_{\mathrm{mix}}(n;\beta,\rho) = \beta\,\rho^n, \quad n \geq 0,\ 0 < \rho < 1.$$

These are examples of the package interface, not additional semantic claims.

Admissible-rate side conditions used by this section are: non-negativity on the declared domain, monotonic non-increase in the step or deviation parameter, and convergence to zero in the asymptotic limit where the bound is intended to vanish.

Theorem 13.3. Tail-Bound Transport. *Let T_{comp} be completion time for the declared execution family. Under Assumption Block 13.1, with interface-verified premise* $\mathsf{TailPkgWF}(\theta)$ *and external analytic premise* $\mathsf{TailAnalyticCert}(\theta)$*,*

$$\forall t \geq 0,\ \Pr\big[T_{\mathrm{comp}} - \mathbb{E}[T_{\mathrm{comp}}] \geq t\big] \leq \Psi_{\mathrm{tail}}(t;\theta). \tag{13.3}$$

The bound is exact with respect to the imported analytic package and does not strengthen structural premises.

Proof sketch. Proof mode: compositional transport. Instantiate Theorem B.1 with target claim equal to the tail inequality. Intermediate step:

Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, *and* 11.1∧$\mathsf{TailPkgWF}(\theta)$ =

Then combine $\mathsf{IfaceTailReady}(\theta)$ with $\mathsf{TailAnalyticCert}(\theta)$ and observable compatibility to obtain the stated bound. □

Theorem 13.4. Mixing-Rate Transport. *Let μ_n be the transported observable distribution at step n and μ_* its reference limit. Under Assumption Block 13.1, with interface-verified premise* $\mathsf{MixPkgWF}(\eta)$ *and external analytic premise* $\mathsf{MixAnalyticCert}(\eta)$*,*

$$\forall n \geq 0,\ \|\mu_n - \mu_*\|_{\mathrm{TV}} \leq \Psi_{\mathrm{mix}}(n;\eta). \tag{13.4}$$

Again, the conclusion imports analytic constants/rates without adding new semantic assumptions to the theorem stack.

Proof sketch. Proof mode: compositional transport. Apply Theorem B.1 with target claim equal to the mixing inequality. Intermediate step:

Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, *and* 11.1∧$\mathsf{MixPkgWF}(\eta)$ =

Use $\mathsf{IfaceMixReady}(\eta)$ plus $\mathsf{MixAnalyticCert}(\eta)$ to derive the contraction estimate on the transported process, yielding the stated total-variation bound. □

14 Discussion: The Classical Boundary

This section is interpretive and does not add theorem premises. Theorems 4.1 through 11.1 remain valid without this interpretation. The discussion explains why the proof architecture has its current shape. It also explains where the current model intentionally stops.

The formal layer establishes exchangeability through delegation-compatible symmetries, local compatibility through active-edge Coherence, and well-posed quotient dynamics through Harmony and envelope laws. These results define an erasure-safe regime for the model class in this paper. Within that regime, safety-visible behavior is invariant under the declared symmetries. This is the

precise sense in which identity details are treated as gauge.

The boundary claim is exact for the stated assumptions. Safe erasures are exactly those justified by Coherence-preserving symmetries and envelope admissibility laws. Behaviors that depend on non-erasable state update mechanisms require a different semantic model. Examples include measurement backaction and entanglement-sensitive observables.

This interpretation is consistent with established MPST coherence lines: global coherence and projectability with projection refinement (Honda et al., 2008; Honda et al., 2016; Castagna et al., 2012; Majumdar et al., 2021), logical n-ary coherence formulations (Carbone et al., 2015; Carbone et al., 2016; Carbone et al., 2017), and data-certification lines that remain projection-based (Toninho and Yoshida, 2017), together with session-typing correspondences in Girard (1987), Caires and Pfenning (2010), and Wadler (2012). The paper does not claim model identity with those frameworks. It claims an operational correspondence at the level of compatibility structure. That correspondence is sufficient for the theorem program in this manuscript.

15 Related Work

Work on reconfiguration and delegation established important safety baselines for dynamic session topologies. These lines often prove preservation-style properties for specific operators. They usually do not package commutation, minimality, and exact envelope bounds in one theorem stack. This paper targets that combined package.

Mechanized MPST developments established high-confidence metatheory and exposed proof fragility under richer operational features. The robustness analysis in Tirore, Bengtson, and Carbone (2025) is especially relevant to this pressure point. The present work responds by factoring the development into reusable assumption blocks and bridge theorems. This choice reduces repetition across reconfiguration, exactness, and adequacy layers. Concretely, their robustness analysis highlights brittleness when adding new operational constructors (for example richer delegation/update features) and when transporting equivalence claims across presentation layers. In this series, assumption-block factoring localizes those obligations: constructor-sensitive premises are isolated in reconfiguration blocks, while observational transport obligations are isolated in bridge/adequacy blocks, reducing cross-theorem proof breakage under feature growth.

Brittleness Pattern (Tirore et al.)	Mitigation in This Series
New operational constructors destabilize proof reuse	Isolate constructor-sensitive obligations in explicit reconfiguration assumptions and `TransitionWF` judgments (Section 2; Theorems 4.1–6.1).
Transporting equivalence claims across layers requires re-proving glue lemmas	Factor bridge composition once at adequacy boundary, then reuse in envelope and transport interfaces (Section 11; Section 13).

Table 15.1: Concrete mapping from reported robustness pressure points to mitigations in this series.

Session foundations from Honda et al. (2008, 2016) and global coherence and projectability refinements (Castagna et al., 2012; Majumdar et al., 2021) remain the base for local and global consistency claims, alongside logical coherence accounts (Carbone et al., 2015; Carbone et al., 2016; Carbone et al., 2017), data-certification extensions (Toninho and Yoshida, 2017), and logical correspondences (Caires and Pfenning, 2010; Wadler, 2012). Program-logical lines such as Hinrichsen et al. (2020) provide strong local reasoning for implementations. Event-structure and partial-order models such as Castellan et al. (2023) provide alternate macro semantics for concurrency. This paper works at the semantic-commutation layer and connects that layer to runtime adherence evidence.

We refine established MPST coherence with an operational invariant over local environments and buffered traces. The novelty here is theorem-level integration of Harmony, relative minimality, exact envelope characterization, and VM adherence over that refined kernel.

Our refinement and adequacy framing is also consistent with classical correspondence and distributed-system baselines (Abadi and Lamport, 1991; Lamport, 1978; Chandy and Lamport, 1985), while Byzantine-facing interfaces inherit the standard adversarial formulation lineage (Lamport et al., 1982).

The main difference of this paper is theorem-level integration of reconfiguration Harmony, relative minimality, exact determinism-envelope characterization, and VM adherence modulo envelope in one proof program.

16 Limitations and Scope

Reconfiguration and delegation claims require the premises of Assumption Block 4.1: typed global and local states, reconfiguration witnesses satisfying WF_ρ, enabled post-reconfiguration steps, and compatibility side conditions for unaffected edges.

Envelope exactness requires the premises of Assumption Block 9.1, including trace well-formedness, certified-step obligations, and the realizability witness schema. VM adherence requires the premises of Assumption Block 11.1, including envelope-related VM and reference states, trace-consistency obligations, and profile-extraction rules for capability reports.

Crash-stop and Byzantine safety claims are scoped to their explicit fault-model bundles. Byzantine liveness beyond the stated timing and fairness assumptions is out of scope.

Higher-order results depend on channel-payload semantics. First-order collapse is a projection result and does not replace higher-order proofs.

Quantitative constants and base decidability primitives are reused assumptions from earlier results in the series rather than re-derived in this paper.

Excluded protocol classes in this paper include non-admitted envelope extensions, non-checkable non-interference witnesses outside finite-witness encodings, and Byzantine liveness claims beyond the declared timing and fairness regimes.

Operational failure mode. If reconfiguration well-formedness or envelope admission obligations fail, the runtime gate should reject the requested link, delegation, or mode transition. If execution proceeds outside admitted envelope premises, the model no longer guarantees safety-visible equivalence, and strict counterexample families from the boundary theorems become reachable. Concrete diagnostic. Expected runtime output is an admission failure citing the first failing `TransitionWF` guard (Table 3.1), or failed $\mathsf{EnvelopeOK}/\mathsf{HasProfileCapabilities}$checks at the theorem-pack gate.

17 Target Application: Unified Distributed Protocol Stacks

The target application is a unified typed protocol stack where networking, coordination, and state evolution are expressed in one choreographic VM framework. The stack uses proof-carrying conformance so capability claims are tied to theorem-pack evidence. This removes informal trust boundaries between layers. It also makes admission failures diagnosable as missing assumptions or missing evidence.

In this setting, typed mode transitions and decidable guard obligations become first-class runtime checks. Delegation supports typed handoff and failover without leaving the theorem scope. Composition theorems provide interoperability guarantees across linked subsystems. Upgrade choreography supports no-downtime migration with typed phase boundaries.

Concrete scenario (running example). Start from $C_0 \xrightarrow{\rho_{\mathsf{link}}} C_1 \xrightarrow{\rho_{\mathsf{del}}} C_2 \xrightarrow{\rho_{\mathsf{tr}}} C_3$. If a primary coordinator fails after linking, delegation from C to backup C' is justified by Theorem 6.1 and Corollary 6.2, preserving delegation safety obligations. The coordinated-mode transition ρ_{tr} is admitted only when the typed guards in Table 3.1 discharge $\mathsf{TransitionWF}(C_2, \rho_{\mathsf{tr}})$, after which Theorem 11.1 and Proposition 11.2 justify runtime adherence/capability reporting for the upgraded mode.

The engineering consequence is a single semantic contract across protocol lifecycle operations. Design-time proofs, deployment-time profile checks, and runtime adherence checks are aligned to the same assumption blocks. This alignment is the main practical payoff of the A through H theorem stack.

Implementation pointer. Reconfiguration and envelope interfaces are implemented in `Runtime/Adequacy/EnvelopeCore/*.lean`; theorem-pack capability surfaces are exported from `Runtime/Proofs/TheoremPack/API.lean`; and VM admission and adherence checks are consumed by runtime gates in the `rust/vm` layer.

18 Conclusion

This paper completes the paper-series theorem arc by proving that reconfiguration can be first-class without sacrificing compositional safety. Harmony supplies commutation, Coherence supplies minimal invariant structure, envelope theorems supply exact behavioral boundaries, and adequacy supplies runtime adherence.

Open item: Byzantine liveness under weaker timing assumptions remains future work beyond this paper's exact safety characterization.

Works Cited

Abadi, M., and Lamport, L. (1991). The Existence of Refinement Mappings. Theoretical Computer Science, 82(2), 253–284.

Alpern, B., and Schneider, F. B. (1985). Defining Liveness. Information Processing Letters, 21(4), 181–185.

Caires, L., and Pfenning, F. (2010). Session Types as Intuitionistic Linear Propositions. CONCUR 2010.

Carbone, M., Lindley, S., Montesi, F., Schürmann, C., and Wadler, P. (2016). Coherence Generalises Duality: A Logical Explanation of Multiparty Session Types. CONCUR 2016.

Carbone, M., Montesi, F., Schürmann, C., and Yoshida, N. (2015). Multiparty Session Types as Coherence Proofs. CONCUR 2015.

Carbone, M., Montesi, F., Schürmann, C., and Yoshida, N. (2017). Multiparty Session Types as Coherence Proofs. Acta Informatica, 54(3), 243–269.

Castagna, G., Dezani-Ciancaglini, M., Gesbert, N., and Padovani, L. (2012). On Global Types and Multi-Party Sessions.

Castellan, S., et al. (2023). Event-structure and partial-order semantics for session-based concurrency. Journal of Logic and Algebraic Methods in Programming.

Chandy, K. M., and Lamport, L. (1985). Distributed Snapshots: Determining Global States of Distributed Systems. ACM Transactions on Computer Systems, 3(1), 63–75.

Clarkson, M. R., and Schneider, F. B. (2010). Hyperproperties. Journal of Computer Security, 18(6), 1157–1210.

Cover, T. M., and Thomas, J. A. (2006). Elements of Information Theory (2nd ed.). Wiley.

Girard, J.-Y. (1987). Linear Logic. Theoretical Computer Science, 50(1), 1–101.

Goguen, J. A., and Meseguer, J. (1982). Security Policies and Security Models. IEEE Symposium on Security and Privacy.

Hinrichsen, J., et al. (2020). Actris: Session-type based reasoning in separation logic. POPL 2020.

Hoare, C. A. R. (1985). Communicating Sequential Processes. Prentice Hall.

Honda, K., Yoshida, N., and Carbone, M. (2008). Multiparty Asynchronous Session Types. POPL 2008.

Honda, K., Yoshida, N., and Carbone, M. (2016). Multiparty Asynchronous Session Types. Journal of the ACM, 63(1), Article 9.

Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7), 558–565.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), 382–401.

Lynch, N. A. (1996). Distributed Algorithms. Morgan Kaufmann.

Majumdar, R., Mukund, M., Stutz, F., and Zufferey, D. (2021). Generalising Projection in Asynchronous Multiparty Session Types. CONCUR 2021.

Milner, R. (1999). Communicating and Mobile Systems: The Pi-Calculus. Cambridge University Press.

Milner, R., Parrow, J., and Walker, D. (1992). A Calculus of Mobile Processes, I and II. Information and Computation, 100(1), 1–77.

Plotkin, G. D. (1981). A Structural Approach to Operational Semantics. DAIMI FN-19.

Shannon, C. E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), 379–423 and 27(4), 623–656.

Companion Series Paper I. (2026). *Coherence for Asynchronous Buffered MPST*. Companion manuscript reviewed with this paper.

Companion Series Paper II. (2026). *Computable Dynamics for Asynchronous MPST*. Companion manuscript reviewed with this paper.

Tirore, L., Bengtson, J., and Carbone, M. (2025). Mechanized MPST metatheory with subject-reduction robustness analysis. ECOOP 2025.

Toninho, B., and Yoshida, N. (2017). Certifying Data in Multiparty Session Types. Journal of Logical and Algebraic Methods in Programming, 90, 61–83.

Wadler, P. (2012). Propositions as Sessions. ICFP 2012.

Appendix

A Assumption Regimes and Dependency Shape

This appendix records regime decomposition and theorem dependencies.

A.1 Regime Classes

Define four premise classes.

1. R_{struct}: reconfiguration well-formedness + coherence transport assumptions.
2. R_{quant}: R_{struct} plus scheduler/budget assumptions for quantitative statements.
3. R_{env}: envelope admissibility + admission/adherence assumptions.
4. R_{byz}: Byzantine assumption bundle layered on R_{env}.

Write $\text{Prem}(R)$ for the premise set of regime R, and define regime order by

$$R \preceq R' :\iff \text{Prem}(R) \subseteq \text{Prem}(R'). \tag{A.1}$$

A.2 Dependency Graph

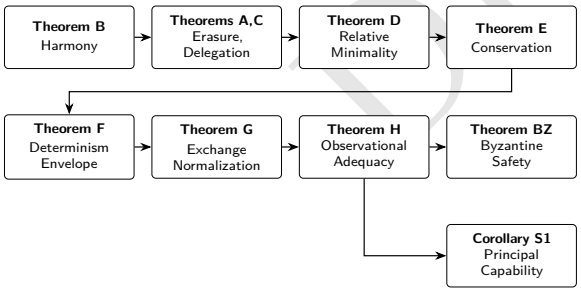

Figure A.1: Theorem dependency graph

Proposition A.1. Regime Monotonicity. *If a theorem requires regime R, then any stronger regime R' with $R \preceq R'$ preserves theorem validity.*

Proof sketch. Assume $R \preceq R'$, so $\text{Prem}(R) \subseteq \text{Prem}(R')$ by definition. Any derivation under R uses only premises from $\text{Prem}(R)$, hence all of its premises are available under R'. Therefore theorem validity is monotone along the regime order. □

A.3 Consequence of the Dependency Graph

Structural theorems (A–D) are insensitive to quantitative profile constants; quantitative and envelope/byzantine claims are regime-indexed exactly as declared in main text assumption blocks.

B Deferred Transport Schemas

Transport results in Section 13 factor into structural and analytic layers.

B.1 Generic Transport Rule

Theorem B.1. Transport by Premise Separation. *If structural premises of Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, and 11.1 hold and analytic package A_T holds for target claim T, then T follows without strengthening structural premises.*

Proof sketch. Proof mode: interface composition. Map the structural side of Theorems 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, 10.1, and 11.1 into the interface required by A_T and obtain intermediate judgment $\mathsf{IfaceReady}_T$. Apply the analytic theorem under $\mathsf{IfaceReady}_T$ to derive claim T in interface form, then transport it back to protocol form through the same map. No additional structural premise is introduced. □

B.2 Interface Discipline

Transport statements must expose only:

1. structural interface obligations (coherence/harmony/envelope/adherence),
2. analytic assumptions external to this manuscript,
3. resulting transported consequence.

This prevents hidden strengthening of either side.

B.3 Representative Schemas

1. Tail-bound transport schema:

$$\text{Struct}_{B..H} \wedge \text{ConcentrationPkg}(\theta) \Longrightarrow \forall t \geq 0,\ \Pr[X - \mathbb{E}X \geq t] \leq \Psi_{\text{tail}}(t;\theta). \tag{B.1}$$

2. Mixing/convergence transport schema:

$$\begin{aligned}&\mathrm{Struct}_{B..H} \wedge \mathrm{MixPkg}(\eta) \Longrightarrow \\ &\quad \forall n \geq 0,\ \|\mu_n - \mu_*\|_{\mathrm{TV}} \leq \Psi_{\mathrm{mix}}(n;\eta).\end{aligned} \tag{B.2}$$

3. Compositional exactness transport schema:

$$\begin{aligned}\mathrm{Struct}_{B..H} \wedge \mathrm{NonInterference} &\Longrightarrow \\ \mathsf{EnvelopeExact}(C_1) \wedge \mathsf{EnvelopeExact}(C_2) &\Rightarrow \\ \mathsf{EnvelopeExact}(C_1 \otimes C_2).&\end{aligned} \tag{B.3}$$

4. Byzantine adherence transport schema:

$$\begin{aligned}\mathrm{Struct}_{B..H} &\wedge \mathrm{ABlockD} \\ &\wedge \mathrm{WitnessPkg} \\ &\Longrightarrow \\ \mathsf{VMByzAdheres} &\\ &\wedge \mathsf{EqsafeConformance}.\end{aligned} \tag{B.4}$$

C Deferred Consequence Statements

C.1 Reconfiguration and Minimality

Proposition C.1.
Under Assumption Block 4.1, Harmony (Theorem 4.1) and Relative Minimality (Theorem 7.1) imply that any admissible safe-reconfiguration invariant implies Coherent *and therefore factors through the same canonical safety core.*

Proof sketch. Theorem 7.1 shows that every admissible invariant factors through Coherent. Theorem 4.1 then gives commutation of reconfiguration steps over that common core. Chaining these two implications yields the proposition. □

C.2 Envelope and Determinism Boundary

Proposition C.2.
Under Assumption Block 9.1, Theorem 9.1 and Theorem 10.1 jointly characterize determinism modulo exchange/spatial normalization and identify strict boundary witnesses for inadmissible envelope extensions.

Proof sketch. Apply Theorem 9.1 to obtain exact envelope characterization via soundness, realizability, and maximality. Apply Theorem 10.1 to identify the normalized observational equalities that hold inside that exact envelope. Combining both gives the claimed determinism boundary statement. □

C.3 Runtime Adequacy and Admission

Proposition C.3.
Under Assumption Block 11.1, Theorem 11.1 plus Corollary 11.3 yields principal-capability admission completeness with explicit rejection diagnostics when premise evidence is missing.

Proof sketch. Theorem 11.1 yields adequacy and adherence under Assumption Block 11.1. Corollary 11.3 then refines this with principal-capability inference and admission completeness. Together they imply the proposition. □

C.4 Byzantine Scope Boundary

Proposition C.4.
Under Assumption Block 11.2, Theorem 11.5 and Corollary 11.6 establish exact safety characterization and converse counterexample packaging, but do not establish Byzantine liveness outside declared timing/fairness assumptions.

Proof sketch. Theorem 11.5 and Corollary 11.6 provide exact Byzantine safety characterization and converse counterexample packaging under Assumption Block 11.2. Section 16 explicitly restricts scope away from weaker timing and fairness assumptions required for Byzantine liveness. Therefore the claimed boundary follows directly. □

D Reproducibility

Reproduction workflow and expected outputs are documented in `ARTIFACT.md`. The one-command supplement check is `just artifact-check`. Pinned-commit, DOI, and Lean-statistics rows are synchronized via `bash scripts/paper_repro_rows.sh -sync papers/paper1.tex papers/paper2.tex papers/paper3.tex`. Artifact semantics note. The Lean artifact exposes an explicit trace-level strong fairness predicate (`StrongFairEnv`) refining weak fairness, and the default in-memory runtime transport implements minimal per-edge FIFO enqueue/dequeue behavior.

Artifact Field	Value
Repository	`https://github.com/hxrts/telltale`
Pinned commit	`ccddc8ea843684c4d2d018f673c4861d0664e008`
Archival DOI	`DOI-UNSET`
Lean source statistics	1303 files, 206913 LOC, axioms: 0, unresolved proof holes (`sorry`): 0

Table D.1: Artifact identity and verification statistics for this draft snapshot.

1. Run `just artifact-check`. This runs reproducibility row checks, `just escape`, `just verify-protocols`, paper builds, and manifest generation.
2. Before camera-ready submission, set DOI in `papers/artifact_metadata.env` and run `just paper-repro-check-strict`.

E Anchor Index

Table E.1 maps paper claims to their Lean anchors and source files. Anchors marked with — are infrastructure lemmas used in proofs but not tied to a single claim. Claim entries include assumption-block tags in square brackets. Assumption-tag legend for claim cells: [A1] = Assumption Block 4.1, [A2] = Assumption Block 9.1, [A3] = Assumption Block 11.1, [A4] = Assumption Block 11.2. Functional categories in the table are: Theorems and Runtime Gates. Reading guide for long names: `exchange_normalization_of_e1_bridge_premises` packages the exchange-normalization bridge used in Theorem 10.1; `consensus_envelope_exact_of_profile` lifts envelope exactness to profile-indexed distributed settings; and `topology_change_preserves_vmc_equiv_via_effect_bisim` gives the topology-change preservation law at the VM observational quotient. Nickname note. For scanning, read `exchange_normalization_of_e1_bridge_premises` as "ENorm-bridge" and `topology_change_preserves_vmc_equiv_via_effect_bisim` as "Topo-VMC-preservation." Long anchors generally follow `object_claim_via_bridge` naming.

Anchor	Path (relative to lean/)	Sec.	Claims
Theorems			
link_harmony_through_link	Protocol/Deployment/LinkingTheorems.lean	§4	Thm. 4.1 [A1]
link_preserves_coherent	Protocol/Deployment/LinkingCore.lean	§4	Thm. 4.1 [A1]
effect_bisim_congr_link	Runtime/Proofs/EffectBisim/Congruence.lean	§4	—
config_equiv_iff_ effect_bisim_silent	Runtime/Proofs/EffectBisim/ ConfigEquivBridge.lean	§4	—
coherent_erasure_stable	Protocol/Coherence/ ErasureCharacterization.lean	§5	Thm. 5.1 [A1]
safe_delegation_local_necessity	Protocol/Coherence/ ErasureCharacterization.lean	§6	—
safe_delegation_to_footprint	Protocol/Coherence/ ErasureCharacterization.lean	§6	—
footprint_to_safe_delegation	Protocol/Coherence/ ErasureCharacterization.lean	§6	Cor. 6.2 [A1]
safe_delegation_iff_footprint	Protocol/Coherence/ ErasureCharacterization.lean	§6	Cor. 6.2 [A1]
relative_minimality	Protocol/Coherence/ ErasureCharacterization.lean	§7	Thm. 7.1 [A1]
flagship_composed_ system_conservation	Protocol/Deployment/LinkingTheorems.lean	§8	Thm. 8.1 [A1]
lyapunov_conserved_under_balanced_ delegation	Runtime/Proofs/Lyapunov.lean	§8	—
consensus_envelope_exact_of_profile	Runtime/Proofs/Adapters/Distributed/ EnvelopeTheorems.lean	§9	Thm. 9.1, Cors. 9.2–9.4 [A2]
failure_envelope_ maximality_of_profile	Runtime/Proofs/Adapters/Distributed/ EnvelopeTheorems.lean	§9	—
exchange_normalization_ of_e1_bridge_premises	Runtime/Adequacy/EnvelopeCore/ ReconfigurationBridge.lean	§10	Thm. 10.1 [A2]
spatial_subtyping_ monotonicity_of_e1_bridge_premises	Runtime/Adequacy/EnvelopeCore/ ReconfigurationBridge.lean	§10	Thm. 10.1 [A2]
vm_adequacy_of_trace_consistency	Runtime/Adequacy/Adequacy.lean	§11	Thm. 11.1 [A3]
protocol_outcome_effect_bisim_of_ observational_eq	Runtime/Proofs/HandlerEquivalence.lean	§11	—
compile_refines_observational_eq_ of_effect_bisim	Runtime/Adequacy/ CompileRefinesEffectBisim.lean	§11	—
vm_view_effect_bisim_of_vmc_equiv	Runtime/Proofs/ ObserverProjectionEffectBisim.lean	§11	—
vm_c_equiv_of_vm_view_effect_bisim	Runtime/Proofs/ ObserverProjectionEffectBisim.lean	§11	—
topology_change_ preserves_vmc_equiv_ via_effect_bisim	Runtime/Proofs/ ObserverProjectionEffectBisim.lean	§11	—
byzantine_safety_exact_of_profile	Runtime/Proofs/Adapters/Distributed/ EnvelopeTheorems.lean	§11	Thm. 11.5, Cor. 11.6 [A4]
Runtime Gates			
canOperateUnderByzantineEnvelope	Runtime/Proofs/TheoremPack/API.lean	§11	Prop. 11.2, Cor. 11.3, Prop. 11.7 [A3, A4]

Table E.1: Anchor index.