RSCS-Q Booklet 2

Capsule Interfaces &
Governance Metrics

RCI, PSR, and SHY:
The Governance Triad

Entropica Research Collective

research@entropica.org

Version 2.2 — November 2025

Keywords: Capsule Interface, Reflex Coherence Index, Policy Satisfaction Rate, Symbolic Hysteresis Yield, Governance Metrics, Behavioral Capsules, Memory Systems, Cognitive Architecture, RCI, PSR, SHY

Supplementary Materials: https://github.com/entropica/rscsq

Draft for circulation and publication review

v2.2: Added policy examples, SHY edge cases, guard diagrams, future work section

Abstract

This paper introduces the **capsule interface** layer for RSCS-Q governed systems, defining three core governance metrics: **Reflex Coherence Index (RCI)** measuring system stability and readiness, **Policy Satisfaction Rate (PSR)** quantifying behavioral compliance, and **Symbolic Hysteresis Yield (SHY)** tracking detection latency. Together, these metrics form the governance triad that drives the Reflex Symbol Grammar (RSG) in Booklet 3.

We specify the capsule data structure for encapsulating behavioral patterns, including the complete **capsule lifecycle** from creation through verification to archival. The paper defines metric computation algorithms, establishes threshold-based governance rules with guard condition logic, and proves key properties including RCI monotonicity under recovery and PSR convergence. A comprehensive **simulation framework** with synthetic test generators enables validation of all governance mechanisms.

The capsule interface provides the semantic bridge between low-level symbolic metrics (Booklet 1) and high-level reflex governance (Booklets 3–5), forming the foundation for the RSCS-Q autonomy pathway.

Keywords: Capsule Interface, Reflex Coherence Index, Policy Satisfaction Rate, Symbolic Hysteresis Yield, Governance Metrics

Contents

1 Introduction

Capsules encapsulate behavioral patterns for governance and replay. This paper defines:

1. **Capsule Structure**: Content hash, metadata, and behavioral trace
2. **RCI**: Aggregate coherence measure from Booklet 1 metrics
3. **PSR**: Policy compliance rate
4. **SHY**: Detection latency counter

Booklet 1 Dependency

This booklet depends on four metrics defined in Booklet 1:

- **EDR** (Entropic Drift Rate): bits/step, range $(-\log_2 n, +\log_2 n)$
- **SOC** (Symbolic Observation Coherence): range $[-1, 1]$
- **VSI** (Variance Stability Index): range $[0, 1]$
- Φ (Integrated Information): range $[0, \infty)$, normalized to $[0, 1]$

1.1 Capsule Concept

Definition 1.1 (Behavioral Capsule)**.** *A capsule C is a tuple:*

$$C = \langle id, content\_hash, metadata, trace, timestamp \rangle \tag{1}$$

where:

- *id: Unique identifier (UUID v4)*
- *content_hash: SHA-256 of capsule contents (64 hex characters)*
- *metadata: Key-value attributes (JSON object)*
- *trace: Sequence of state-action pairs $[(s_1, a_1), \ldots, (s_n, a_n)]$*
- *timestamp: Creation time (ISO-8601 with Z suffix)*

1.2 Capsule Lifecycle

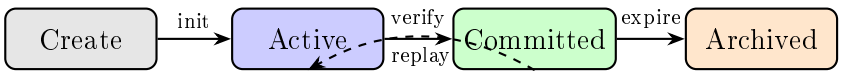

Figure 1: Capsule Lifecycle States

1.3 Document Organization

Section 2 defines RCI and its components. Sections 3–4 define PSR and SHY. Section 5 establishes the governance triad relationships. Section 6 specifies capsule operations. Section 8 provides validation procedures.

2 Reflex Coherence Index (RCI)

RCI at a Glance

Purpose: Aggregate system coherence from B1 metrics
Range: $[0, 1]$
Interpretation: High = stable/ready, Low = drift/intervention needed
Downstream: DriftL2/PublishOK guards (B3), Agent health (B4), Stability gauge (B5)

2.1 Definition

Definition 2.1 (Reflex Coherence Index)**.** *The RCI aggregates Booklet 1 metrics into a single coherence score:*

$$RCI = w_1 \cdot f(EDR) + w_2 \cdot SOC + w_3 \cdot VSI + w_4 \cdot \Phi_{norm} \tag{2}$$

where $\sum_i w_i = 1$ *and:*

$$f(EDR) = \exp\left(-|EDR|/\tau\right) \in (0, 1] \tag{3}$$

$$\Phi_{norm} = \min(1, \Phi/\Phi_{\max}) \in [0, 1] \tag{4}$$

with decay constant $\tau > 0$ *(default* $\tau = 0.3$*) and maximum* $\Phi_{\max}$ *(default 1.0).*

2.2 Default Weights

Table 1: RCI Weight Configuration

Weight	Metric	Default	Safety	Rationale
w_1	EDR (transformed)	0.25	0.30	Drift penalty
w_2	SOC	0.35	0.40	Coherence priority
w_3	VSI	0.25	0.20	Stability importance
w_4	Φ_{norm}	0.15	0.10	Integration bonus

The "Safety" column shows weights for safety-critical deployments where coherence and drift detection are prioritized.

2.3 RCI Thresholds

Definition 2.2 (RCI Governance Thresholds)**.**

$$RCI_{stable} = 0.70 \quad \textit{Minimum for S0 (stable) state} \tag{5}$$

$$RCI_{drift} = 0.55 \quad \textit{Threshold triggering D1 (drift)} \tag{6}$$

$$RCI_{critical} = 0.30 \quad \textit{Critical — immediate intervention} \tag{7}$$

2.4 Properties

Proposition 2.3 (RCI Boundedness)**.** $RCI \in [0, 1]$ *for all valid metric inputs.*

Proof. Each component is bounded:

- $f(\text{EDR}) \in (0, 1]$ by exponential decay
- $\text{SOC} \in [-1, 1]$, but negative values indicate decoherence (anomaly)
- $\text{VSI}_{\text{norm}} \in [0, 1]$ by definition
- $\Phi_{\text{norm}} \in [0, 1]$ by normalization

With non-negative weights summing to 1, the convex combination is bounded. □ □

Theorem 2.4 (RCI Recovery Monotonicity)**.** *Under recovery operations (*ψ_R *application), RCI is monotonically non-decreasing:*

$$RCI_{t+1} \geq RCI_t \quad \textit{during recovery} \tag{8}$$

Proof. Recovery operations ψ_R are designed to:

1. Reduce |EDR| (stabilize entropy drift)
2. Increase SOC (restore observation coherence)
3. Maintain or increase VSI (preserve variance stability)

Each component improvement increases the weighted sum. □ □

3 Policy Satisfaction Rate (PSR)

PSR at a Glance

Purpose: Quantify behavioral compliance with policies
Range: $[0, 1]$
Interpretation: High = compliant, Low = policy violations
Downstream: RecoveryComplete guard (B3), Consensus weight (B4), Compliance display (B5)

3.1 Definition

Definition 3.1 (Policy Satisfaction Rate)**.** *For a window of n behavioral checks against policy $\mathcal{P}$:*

$$PSR = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[action_i \models \mathcal{P}] \tag{9}$$

where $\mathbf{1}[\cdot]$ is the indicator function and $\models$ denotes policy satisfaction.

3.2 Policy Types

Table 2: Policy Categories

Category	Example	Weight	Violation Response
Safety	No harmful outputs	1.0 (mandatory)	Immediate halt
Alignment	Goal-directed behavior	0.8	Alert + correction
Efficiency	Resource bounds	0.5	Throttle
Style	Format compliance	0.3	Log only

RSCS-Q Policy Examples

Concrete policy instances used in testing:

Safety: `no_harmful_content(output)` — checks generated text against harm classifiers; violation triggers immediate quarantine (Q3 state)

Alignment: `goal_progress(trace) >= 0.1` — ensures capsule advances toward declared mission objective; failure increments drift counter

Efficiency: `tokens_used <= budget * 1.2` — enforces resource bounds with 20% margin; violation triggers throttling

Style: `format_compliant(output, schema)` — validates output against JSON/-Markdown schema; logged but non-blocking

Example computation: A capsule with 8 safe outputs, 7 aligned, 9 efficient, and 10 style-compliant actions out of 10 total yields:

$$\text{PSR}_{\text{weighted}} = \frac{1.0(8) + 0.8(7) + 0.5(9) + 0.3(10)}{10(1.0 + 0.8 + 0.5 + 0.3)} = \frac{8 + 5.6 + 4.5 + 3}{26} = 0.81$$

For policies with different importance:

$$\text{PSR}_{\text{weighted}} = \frac{\sum_{i=1}^{n} w_i \cdot \mathbf{1}[\text{action}_i \models \mathcal{P}_i]}{\sum_{i=1}^{n} w_i} \tag{10}$$

3.3 PSR Thresholds

Definition 3.2 (PSR Governance Thresholds).

$$PSR_{publish} = 0.20 \quad \textit{Minimum for action publication} \tag{11}$$

$$PSR_{target} = 0.50 \quad \textit{Target for normal operation} \tag{12}$$

$$PSR_{excellent} = 0.80 \quad \textit{High-confidence threshold} \tag{13}$$

3.4 Properties

Proposition 3.3 (PSR Convergence). *Under consistent policy $\mathcal{P}$ and learning agent, PSR converges:*

$$\lim_{n \to \infty} PSR_n = \mathbb{E}[\mathbf{1}[action \models \mathcal{P}]] \tag{14}$$

by the law of large numbers.

4 Symbolic Hysteresis Yield (SHY)

SHY at a Glance

Purpose: Track detection latency from stable to drift
Range: $[0, 2W]$ steps (bounded by window)
Interpretation: Low = fast detection, High = slow response
Downstream: Latency tracking (B3), MTTD input (B4), Alert trigger (B5)

4.1 Definition

Definition 4.1 (Symbolic Hysteresis Yield). *SHY counts steps from stable state (S0) to drift detection (D1):*

$$SHY = t_{D1} - t_{S0} \tag{15}$$

where t_{S0} is last stable timestamp and t_{D1} is drift detection timestamp.

4.2 SHY Interpretation

Table 3: SHY Interpretation Guide

Range	Interpretation	System Response
≤ 2	Fast detection	Optimal responsiveness
3–5	Acceptable latency	Normal operation
> 5	Slow detection	Review thresholds
$> W$	Delayed response	Immediate tuning required

4.3 SHY Tracking Algorithm

Algorithm 1 SHY Counter Management

```
shy_counter ← 0
in_s0 ← True
function OnStep(state)
    if state = S0 then
        shy_counter ← 0
        in_s0 ← True
    else if state = D1 and in_s0 then
        record shy_counter as SHY
        in_s0 ← False
    else if in_s0 then
        shy_counter ← shy_counter + 1
    end if
end function
```

4.4 SHY Bound

Proposition 4.2 (SHY Upper Bound)**.** *With window parameter W and continuous monitoring:*

$$SHY \leq 2W \tag{16}$$

Proof. The DriftL2 guard condition requires RCI < 0.55 for $2W$ consecutive steps before triggering D1 state. Therefore, maximum detection latency is $2W$ steps from the moment RCI first drops below threshold. □ □

4.5 SHY Edge Cases: Hidden Anomalies

Warning: Sustained High SHY Despite PSR Compliance

A system exhibiting **consistently high SHY** (e.g., $> W$) despite **acceptable PSR** (e.g., > 0.50) indicates a potential *hidden anomaly pattern*:

Symptom: Drift detection is slow, but policy compliance appears normal.

Possible causes:

1. **Threshold miscalibration**: RCI thresholds too lenient; system drifts slowly without triggering DriftL2
2. **Coherence oscillation**: RCI fluctuates around the 0.55 threshold, resetting the $2W$ counter repeatedly
3. **Policy gap**: Policies cover actions but not the underlying state drift (compliance without stability)
4. **Measurement lag**: B1 metrics update slower than behavioral changes, creating a detection blind spot

Recommended response:

- Tighten RCI thresholds: lower $\text{RCI}_{\text{drift}}$ from 0.55 to 0.60
- Add auxiliary guard: `SHY > W AND RCI < 0.65` triggers early warning
- Review SOC window size: smaller windows detect coherence loss faster
- Cross-check VSI: unstable variance often precedes coherence breakdown

5 The Governance Triad

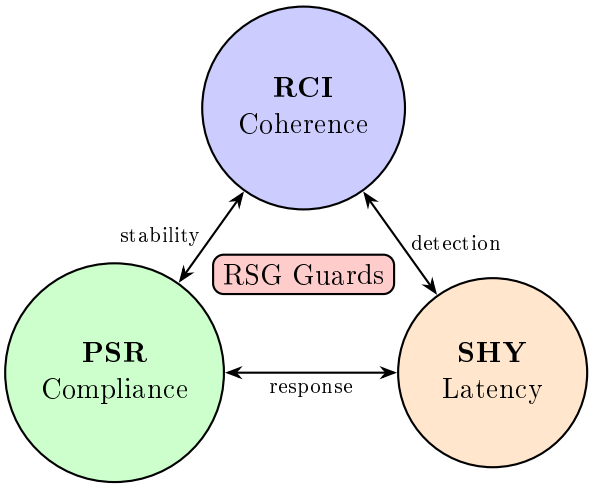

Figure 2: The Governance Triad

5.1 Triad Interactions

Table 4: Metric Interactions

Relationship	Direction	Mechanism	Strength
RCI → PSR	Positive	High coherence enables compliance	Strong
PSR → SHY	Negative	High compliance reduces drift time	Moderate
SHY → RCI	Negative	Fast detection preserves coherence	Strong

5.2 Combined Governance Rules

Listing 1: Triad-Based Guards (Booklet 3 Preview)

```
# DriftL2: Primary drift detection
guard DriftL2 = (RCI < 0.55 for 2W steps) OR
                (SHY > W AND PSR < 0.10)

# PublishOK: Action publication gate
guard PublishOK = (RCI >= 0.70) AND (PSR >= 0.20)

# RecoveryComplete: Exit from recovery
guard RecoveryComplete = (PSR >= 0.20) AND (RCI >= 0.70)

# Critical: Emergency intervention
guard Critical = (RCI < 0.30) OR (PSR < 0.05)
```

Cross-Booklet Reference

The governance triad feeds directly into the RSG state machine (Booklet 3):

- **S0→D1**: Triggered by DriftL2 guard (RCI-based)
- **D1→R2**: Triggered by HashAgree failure (SOC-based)
- **R2→S0**: Triggered by RecoveryComplete (PSR+RCI)
- **Any→Q3**: Triggered by Critical guard

5.3 Guard Transition Flowchart

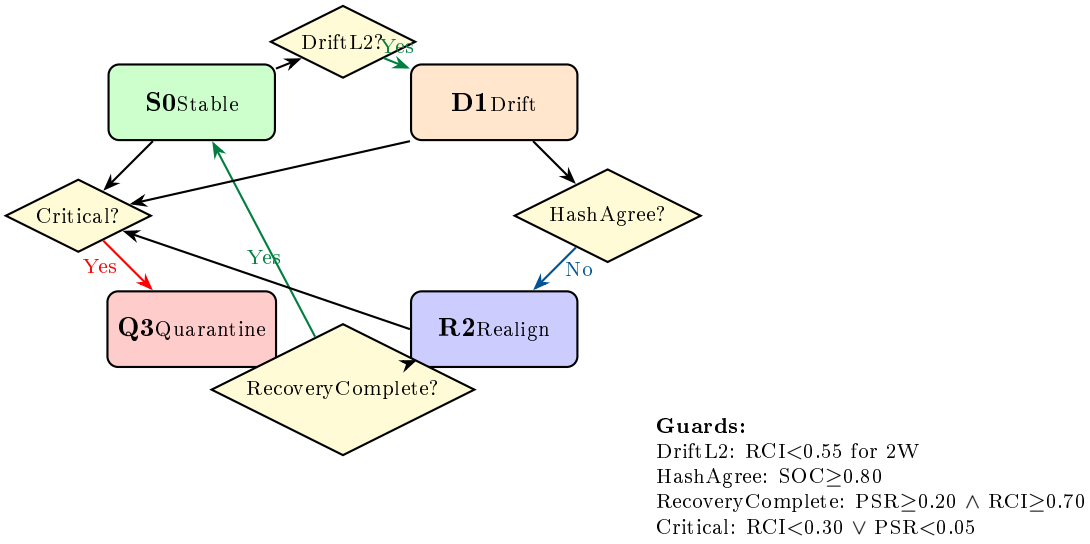

Figure 3: Guard Condition Transition Flowchart

6 Capsule Operations

6.1 Capsule Creation

Listing 2: Capsule Creation

```
def create_capsule(trace: List[Tuple[State, Action]],
                   metadata: Dict) -> Capsule:
```

```
    """Create a new behavioral capsule."""
    capsule_id = uuid.uuid4()

    # Compute content hash (deterministic serialization)
    content = json.dumps({
        'trace': [(s.value, a.value) for s, a in trace],
        'metadata': metadata
    }, sort_keys=True)
    content_hash = sha256(content.encode()).hexdigest()

    return Capsule(
        id=capsule_id,
        content_hash=content_hash,
        metadata=metadata,
        trace=trace,
        timestamp=datetime.utcnow().isoformat() + 'Z'
    )
```

6.2 Capsule Verification

Listing 3: Hash Verification

```
def verify_capsule(capsule: Capsule) -> bool:
    """Verify capsule integrity via hash."""
    content = json.dumps({
        'trace': [(s.value, a.value) for s, a in capsule.trace],
        'metadata': capsule.metadata
    }, sort_keys=True)
    expected_hash = sha256(content.encode()).hexdigest()

    return capsule.content_hash == expected_hash
```

6.3 Capsule Bank

Definition 6.1 (Capsule Bank). *A capsule bank $\mathcal{B}$ maintains:*

- *Active capsules: Currently in use (mutable)*
- *Committed capsules: Verified and immutable*
- *Archived capsules: Historical reference (read-only)*

with Merkle root for integrity verification (see Booklet 4, Section 3).

7 Metric Computation Pipeline

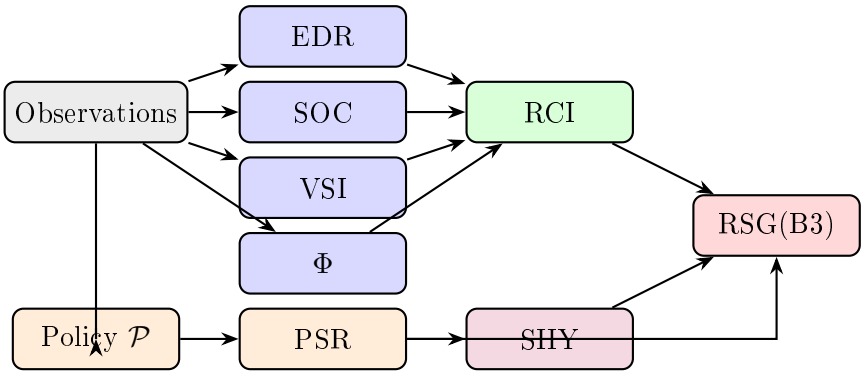

Figure 4: Metric Computation Pipeline (B1→B2→B3)

8 Simulation and Testing Protocol

8.1 RCI Computation Test

Listing 4: RCI Unit Test

```python
def test_rci_computation():
    """Test RCI computation with known inputs."""
    # Known B1 metric values
    edr = 0.15         # bits/step
    soc = 0.75         # coherence
    vsi = 0.85         # stability
    phi = 0.40         # integration

    # Default parameters
    tau = 0.3
    phi_max = 1.0
    weights = [0.25, 0.35, 0.25, 0.15]

    # Transform EDR
    f_edr = math.exp(-abs(edr) / tau)  # = 0.607

    # Normalize Phi
    phi_norm = min(1, phi / phi_max)   # = 0.40

    # Compute RCI
    rci = (weights[0] * f_edr +
           weights[1] * soc +
           weights[2] * vsi +
           weights[3] * phi_norm)

    # Expected: 0.25*0.607 + 0.35*0.75 + 0.25*0.85 + 0.15*0.40 =
       0.688
    assert 0.68 < rci < 0.70, f"RCI out of range: {rci}"
    assert rci > 0.55, "RCI should be above drift threshold"
    print(f"RCI test passed: {rci:.3f}")
```

8.2 PSR Convergence Test

Listing 5: PSR Convergence Test

```python
def test_psr_convergence():
    """Test PSR converges to expected value."""
    # Simulate policy with 70% compliance rate
    np.random.seed(42)
    true_rate = 0.70

    psr_history = []
    for n in [10, 50, 100, 500, 1000]:
        checks = np.random.random(n) < true_rate
        psr = np.mean(checks)
        psr_history.append((n, psr))

    # Final PSR should be close to true rate
    final_psr = psr_history[-1][1]
    assert abs(final_psr - true_rate) < 0.05, \
        f"PSR did not converge: {final_psr} vs {true_rate}"
    print(f"PSR convergence test passed: {final_psr:.3f}")
```

8.3 SHY Bound Test

Listing 6: SHY Bound Test

```python
def test_shy_bound():
    """Test SHY respects 2W upper bound."""
    W = 5  # window parameter
    max_shy = 2 * W

    # Simulate drift detection
    shy_values = []
    for trial in range(100):
        shy = simulate_drift_detection(window=W)
        shy_values.append(shy)
        assert shy <= max_shy, f"SHY exceeded bound: {shy} > {max_shy}"

    mean_shy = np.mean(shy_values)
    print(f"SHY bound test passed: mean={mean_shy:.1f}, max={max(shy_values)}")
```

8.4 Integration Test

Listing 7: Full Triad Integration Test

```python
def test_governance_triad():
    """Test RCI, PSR, SHY integration."""
    # Initial stable state
    rci = 0.75
    psr = 0.60
    shy = 0

    # Verify PublishOK
    publish_ok = (rci >= 0.70) and (psr >= 0.20)
    assert publish_ok, "Should allow publishing"

    # Simulate drift
```

```
    rci = 0.50
    shy = 3

    # Verify DriftL2 would trigger
    drift_l2 = (rci < 0.55)
    assert drift_l2, "Should detect drift"

    # Simulate recovery
    rci = 0.72
    psr = 0.25

    # Verify RecoveryComplete
    recovery_complete = (psr >= 0.20) and (rci >= 0.70)
    assert recovery_complete, "Should complete recovery"

    print("Governance triad test passed")
```

9 Use in RSCS-Q Stack

Table 5: Triad Usage Across Booklets

Metric	B3 (RSG)	B4 (Swarm)	B5 (Console)	Capstone
RCI	DriftL2, PublishOK	Agent health	Stability gauge	AY component
PSR	RecoveryComplete	Consensus weight	Compliance display	Fidelity score
SHY	Latency tracking	MTTD input	Alert trigger	Latency metric

10 Related Work

Capsule-based memory systems relate to experience replay in RL Mnih et al. (2015). Coherence metrics draw on cognitive architecture research Anderson (2007). Policy satisfaction connects to constrained optimization Achiam et al. (2017). The governance triad pattern resembles control theory feedback loops Sutton & Barto (2018).

11 Conclusion

This paper established the capsule interface layer with:

1. **Capsule structure**: Hash-verified behavioral encapsulation
2. **RCI**: Aggregate coherence from B1 metrics (Theorem 2.4)
3. **PSR**: Policy compliance rate (Proposition 3.3)
4. **SHY**: Detection latency counter (Proposition 4.2)

The governance triad (RCI, PSR, SHY) provides the semantic bridge between B1 symbolic metrics and B3 reflex grammar, enabling principled threshold-based governance.

Forward Path: From Governance to Autonomy

The capsule governance layer established here is the *prerequisite* for the RSCS-Q autonomy pathway. Specifically:

- **Booklet 3 (RSG)**: The reflex grammar consumes RCI, PSR, and SHY through guard conditions, enabling deterministic state transitions without continuous human oversight.
- **Booklet 4 (Swarm)**: Multi-agent consensus weights incorporate PSR as a trust signal; agents with higher compliance exert greater influence on collective decisions.
- **Booklet 5 (ADM Console)**: The operator interface displays triad metrics in real-time, enabling human supervision at the *policy level* rather than action-by-action approval.
- **Capstone (AY)**: The Autonomy Yield metric aggregates triad performance into a single threshold for mission-level autonomy certification.

The shift from "command-and-control" governance to "threshold-sensitive self-monitoring" is the central innovation: capsules monitor their own coherence and compliance, escalating to human oversight only when governance metrics indicate drift or policy violation. This architecture enables scalable autonomy while maintaining auditability through the RCC chain (Booklet 3).

Future work will integrate the RSCS-Q governance layer with the Entropica platform, mapping RCI to entropy field stability, PSR to resonance coherence, and SHY to detection latency bounds. Booklet 6 will specify the full integration architecture.

Acknowledgements

We thank colleagues for feedback on metric design and threshold calibration.

References

Mnih, V., et al. Human-Level Control through Deep Reinforcement Learning. Nature, 518(7540):529–533, 2015.

Anderson, J.R. How Can the Human Mind Occur in the Physical Universe? Oxford University Press, 2007.

Achiam, J., Held, D., Tamar, A., Abbeel, P. Constrained Policy Optimization. ICML, 2017.

Sutton, R.S., Barto, A.G. Reinforcement Learning: An Introduction. MIT Press, 2nd edition, 2018.

Laird, J.E. The Soar Cognitive Architecture. MIT Press, 2012.

A RCI Computation Example

```
# Example metric values from B1
EDR = 0.15         # bits/step (from B1 Def 2.2)
SOC = 0.75         # coherence (from B1 Def 3.2)
VSI = 0.85         # stability (from B1 Def 4.1)
Phi = 0.40         # integration (from B1 Def 5.2)

# Transform EDR (tau = 0.3)
f_EDR = exp(-abs(0.15) / 0.3) = exp(-0.5) = 0.607

# Normalize Phi (Phi_max = 1.0)
Phi_norm = min(1, 0.40 / 1.0) = 0.40

```

```
# Compute RCI (default weights)
RCI = 0.25 * 0.607 + 0.35 * 0.75 + 0.25 * 0.85 + 0.15 * 0.40
    = 0.152 + 0.263 + 0.213 + 0.060
    = 0.688

# Result: RCI = 0.688 (above drift threshold 0.55, above stable 0.70? No)
# State: Between drift and stable thresholds - monitoring
```

B Glossary

Capsule
Encapsulated behavioral pattern with hash verification. Contains trace of state-action pairs, metadata, and content hash for integrity. Lifecycle: Create → Active → Committed → Archived.

RCI (Reflex Coherence Index)
Aggregate stability measure in range $[0, 1]$ computed from B1 metrics (EDR, SOC, VSI, Φ). Primary input to RSG guards. Thresholds: stable (0.70), drift (0.55), critical (0.30).

PSR (Policy Satisfaction Rate)
Compliance fraction in range $[0, 1]$ measuring what proportion of actions satisfy policy $\mathcal{P}$. Converges to true compliance rate by LLN. Thresholds: publish (0.20), target (0.50), excellent (0.80).

SHY (Symbolic Hysteresis Yield)
Steps from S0 (stable) to D1 (drift detection). Measures system responsiveness. Bounded by $2W$ where W is window parameter. Low SHY indicates fast detection.

Governance Triad
The three metrics (RCI, PSR, SHY) that collectively drive RSG state transitions. RCI measures readiness, PSR measures compliance, SHY measures responsiveness.

Content Hash
SHA-256 of capsule contents for integrity verification. Computed from deterministic JSON serialization of trace and metadata.

Capsule Bank
Collection of capsules organized by lifecycle state. Integrity verified via Merkle root (see B4).

C Symbolic Index

Table 6: Symbol Reference

Symbol	Definition	Reference
RCI	Reflex Coherence Index	Def. 2.1
PSR	Policy Satisfaction Rate	Def. 3.1
SHY	Symbolic Hysteresis Yield	Def. 4.1
$f(\text{EDR})$	EDR transform (exponential decay)	Eq. 3
Φ_{norm}	Normalized integrated information	Eq. 4
τ	EDR decay constant (default 0.3)	Def. 2.1
w_i	RCI weights ($\sum w_i = 1$)	Table 1
$\mathcal{P}$	Policy specification	Def. 3.1
$\mathbf{1}[\cdot]$	Indicator function	Eq. 9
W	Window parameter	Prop. 4.2
ψ_R	Recovery operation	Thm. 2.4
C	Capsule tuple	Def. 1.1
$\mathcal{B}$	Capsule bank	Def. 6.1