Political OS Suite

Four competing political philosophies expressed as formal constraint specifications

Worked examples of Logical Encapsulation from the Constraint-Emergence Ontology. Each document defines axioms, invariants, and an evaluation algorithm that programs an LLM to reason within a specific political framework. The political domain demonstrates the method; the contribution is the logical architecture.

The Four Operating Systems

Document	OS	Primary Unit	Nature
Classical Liberal	Classical Liberal	Individual	Full governance system — iterative optimization loop protecting four invariants (Agency, Information, Alternatives, Revocability)
Marxist	Marxist	Class	Diagnostic with governance gap — strong diagnosis of exploitation, vanguard gap at implementation
Critical Justice	Critical Justice	Intersectional identity group	Diagnostic program — analytical invariants for identifying structural power, no governance model
Theocratic	Theocratic	Divine order	Full governance system — divine authority with

Document	OS	Primary Unit	Nature
			interpretation gap

Supporting Documents

Document	Purpose
The Governance Stack	Layered governance model (hardware → OS → runtime → program → bootstrap) and cross-OS structural analysis. Includes completeness-as-discovery methodology — the analysis discovers whether each candidate is a full OS or a fragment.
US Democratic Political OS	The US Constitutional system mapped as an implementation of the Classical Liberal OS — where invariants are constitutionally hardened and where they depend on corruptible programs.
Political OS Test Suite	15 test cases with predicted results across all four OS. Six evaluation methods including hosted execution (fragment testing).

Reports

Real-world analyses applying the framework to current political events.

Report	Date	Subject
Australia Invariant Analysis	2026-02-16	Australian Labor government legislative programme (2022-2026) analysed through Liberal OS invariants. Tests the prediction that tradition-carried invariants are more fragile than constitution-carried invariants.

How to Use

Quick start — see it work

1. Load **Classical Liberal Political OS** into a fresh LLM session (paste the entire document as context)
2. Ask it to evaluate a political phenomenon: *"Evaluate mandatory digital identity systems using the framework's evaluation algorithm"*
3. The LLM will reason within the constraints — triage first, then invariant test, then system state classification
4. Load a different OS (e.g., Critical Justice) in a **new session** with the same question
5. Compare the mechanically divergent results

Run the test suite

1. Read the **Test Suite** for full instructions
2. **Critical rule**: Each LLM session gets **exactly one OS document**. Never load two in the same session.
3. Start with Method 1 (single case) to familiarize yourself
4. Method 6 (Hosted Execution) tests the claim that CJ/Marxist are diagnostic fragments by loading them as programs on a host OS

Understand the comparative analysis

1. Read at least two OS documents (Classical Liberal + one other)
2. Read **The Governance Stack** for the structural comparison
3. Key findings: CJ self-discovers as a diagnostic program; CJ and Theocratic OS are structurally isomorphic (interpretive priesthood, injected consciousness, unfalsifiability); the same diagnostic produces opposite outcomes at different stack levels

Write a real-world analysis

1. Choose a political situation
2. Run it through the Liberal OS evaluation algorithm: Pre-Evaluation Triage (4 steps) → Invariant Test → System State Classification
3. Note which invariants are affected and whether constitutional protections exist
4. Save to `reports/` with date prefix: `YYYY-MM-DD-subject.md`

Key Concepts

- **Invariants**: Hard constraints that must never be violated. Each OS defines four.
- **Programs**: Policies, laws, institutions — they run on the OS and can be changed. Programs can corrupt the OS.
- **Pre-Evaluation Triage** (Liberal OS): 4-step filter before invariant testing — formal encoding → enforcement asymmetry → runtime distortion → distributional artifact. Prevents both denialism and presumptive encoding.
- **Completeness-as-discovery**: The framework doesn't assume whether a candidate is a full OS or a fragment — the analysis discovers it.

- **Iterative design**: The Liberal OS invariants protect an optimization loop, not single-step aggregation. Arrow's Theorem explains the design rather than wounding it.

Parent Framework

This suite is a worked example of: - **Constraint-Emergence Ontology** — the philosophical framework - **Ontology Templates** — the Logical Encapsulation method used to build each OS - **Emergent Reasoning** — the formal model of how LLMs process constraint specifications