# 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.

---

## Start Here

**The Political Operating System** is the main paper. It introduces the Governance Stack model, compares all four specifications structurally, and delivers the key findings. Read it first.

After reading it, load any individual OS specification into a fresh LLM session and ask it to evaluate a political phenomenon — you will see the constraint system in action.

---

## Documents

### Main Paper

| Document | Description |
|---|---|
| The Political Operating System | Entry point. Five-layer Governance Stack, structural comparison of all four OS, key findings (CJ-Theocratic isomorphism, evolved systems vs diagnostic fragments, provenance analysis). |

### OS Specifications

Each is a self-contained constraint system. Load one into a fresh LLM session — never two in the same session.

| Document | OS | Primary Unit | Nature |
|---|---|---|---|
| Classical Liberal | Classical Liberal | Individual | Full governance system — iterative |

| Document | OS | Primary Unit | Nature |
|---|---|---|---|
| | | | 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 interpretation gap |

## Supporting Documents

| Document | Purpose |
|---|---|
| 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 | 2026-02-16 | Labor government (2022-2026). Tests the prediction that tradition-carried invariants are more fragile than constitution-carried invariants. |
| United Kingdom | 2026-02-16 | Conservative and Labour governments (2019-2026). No written constitution — invariants carried entirely by convention and statute. Bi-directional degradation from both left and right. |
| Canada | 2026-02-16 | Trudeau government (2015-2025). Tests whether Charter rights with Section 1 and Section 33 overrides protect or enable invariant degradation. |
| Germany | 2026-02-16 | Scholz coalition (2021-2026). Strongest constitutional hardening (Grundgesetz + eternity clause + BVerfG) — but militant democracy doctrine creates a constitutional override mechanism. |
| United States | 2026-02-16 | Biden and Trump administrations (2020-2026). Symmetrical analysis — both sides degrade different invariants. Tests constitutional hardening under stress from competing directions. |
| California | 2026-02-16 | Democratic supermajority (2020-2026). Sub-national case study — tests whether state-level governance can degrade invariants despite federal constitutional protections. |

---

# Quick Start

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

For structured experiments, see the **Test Suite**.

---

# 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 (separate repo)