Supplementary Material

To Notch a Stone with Six Birds: Time as a Theory Artifact of Order, Measure, and Arrow

Ioannis Tsiokos

10 February 2026

S1 Reproducibility: regenerating artifacts and paper tables

Environment setup

The Python package requires Python ≥ 3.9. From the repository root:

```
cd python
python -m venv .venv
source .venv/bin/activate
pip install -e .
```

Dependencies are declared in `python/pyproject.toml`. The core runtime dependency is `numpy`; development dependencies include `pytest`, `ruff`, `mypy`, and `matplotlib`.

Regenerating all artifacts

All experiments write artifacts under `artifacts/` (JSON and CSV). The recommended end-to-end regeneration sequence from the repository root is:

```
python python/scripts/run_all_exhibits_smoke.py
```

This runner executes the exhibit scripts and verifies that the expected artifacts exist, including at minimum:

- `artifacts/exhibit_dpi_smoke/metadata.json`
- `artifacts/exhibit_clock_budget_smoke/metadata.json`
- `artifacts/exhibit_enablement_birth_smoke/metadata.json`
- `artifacts/exhibit_constraints_cones_smoke/metadata.json`
- `artifacts/exhibit_no_global_time_smoke/metadata.json`
- `artifacts/exhibit_no_signalling_toy/metadata.json`
- `artifacts/sweeps/sweep_smoke/results.csv`
- `artifacts/sweeps/sweep_smoke/summary.json`

Regenerating paper tables

Paper tables are generated from these artifacts by:

```
python python/scripts/paper/make_paper_tables.py
```

The generated tables are written to `docs/paper/tables/` and included in the manuscript via `\input{tables/...}`.

Building the PDF

After regenerating tables:

```
cd docs/paper
latexmk -pdf -interaction=nonstopmode \
  -halt-on-error to_notch_a_stone_with_six_birds.tex
```

Running tests

```
cd python && pytest
```

S2 Mechanized anchors (Lean)

We include lightweight Lean 4 formalizations as structural anchors for several claims. These files are intended to mechanize small algebraic skeletons cited in the narrative, not to verify the empirical audits end-to-end.

Holonomy obstruction (no global time). File: `lean/TimeWorld/HolonomyNoGlobalTime.lean`. Key identifiers: `triangle_sum_of_potential` and `no_global_potential_of_nonzero_triangle_holonomy`. These capture the telescoping identity for exact 1-forms and the obstruction implied by nonzero cycle sum.

Closure descent to fixed points. File: `lean/TimeWorld/DescentToFixpoints.lean`. Key identifiers: `map_fix_of_commute` and `restrictToFix`. These encode a basic "descent" fact: if an idempotent packaging map commutes with an update, then the update restricts to the packaged fixed-point subspace.

Ledger preorder anchor. File: `lean/TimeWorld/LedgerPreorder.lean`. Key identifiers: `ledgerPreorder` and `ledger_step_le_of_monotone`. These show how a monotone ledger induces a preorder compatible with an update rule.

No-signalling toy anchors. File: `lean/TimeWorld/NoSignallingToy.lean`. Key identifiers: `marginalB_uniform_of_xor_constraint` and `signalling_marginalB_depends_on_x`. These formalize, in a minimal Boolean setting, that constraint-mediated sharp conditionals do not imply a signalling channel.

S3 Code map (Python)

The main Python components are organized under `python/src/time_world/`:

- `model.py`: toy Markov world construction and simulation
- `audits_ep.py`: stationary distribution and entropy production
- `audits_path_kl.py`: DPI-safe path-reversal KL estimation under lenses
- `clock_audits.py`: clock viability metrics, including progress/anti-stall rates
- `enablement.py`: closure defect and forced theory extension
- `constraints_cones.py`: constraint masks and reachability cones
- `holonomy.py`: protocol holonomy measurement

- `no_signalling_toy.py`: constraint vs signalling boxes

Exhibit scripts live under `python/scripts/`. See `docs/experiments/index.md` for the internal runbook list.