BCQM VII Session Log
Stage–2 Cloth Bring-up and Provenance Verification (v0.1)

Peter M. Ferguson
Independent Researcher

29 January 2026

Timestamp

Session start reference: **Thursday 29/01/26 10:56** (Europe/London).

Purpose

Record Stage–2 bring-up actions in the new BCQM VII repo: (i) provenance verification against the Zenodo-extracted BCQM VI code; (ii) Stage–2 cloth objective selection and initial implementation strategy (bin-based concurrency); (iii) first cloth bring-up run(s) and diagnostics; (iv) refinement from "concurrent edges" to "edges used by the lockstep core"; and (v) the immediate next steps proposed for tomorrow.

1. Provenance verification (BCQM VI → BCQM VII)

Two folders were prepared locally:

- `bcqm_vi_spacetime`: extracted from the Zenodo archive of the BCQM VI reference implementation (release DOI 10.5281/zenodo.18403109).
- `bcqm_vii_cloth`: the trimmed Stage–2 working copy prepared for BCQM VII.

A "just-run" SHA256 comparison script (`prove_vi_lineage.sh`) was executed on a fixed list of 14 core files (CLI, runner, v_glue engine, event graph, schema, pipelines, and key analysis scripts). The result was:

- MATCH: 14
- DIFF: 0
- MISS_VI: 0
- MISS_VII: 0

This constitutes a cryptographic, file-by-file proof that the tested BCQM VII core runtime and pipeline files are byte-identical to the Zenodo-extracted BCQM VI release. The proof artefacts (script + timestamped report) were placed under `docs/provenance/` and committed to the new GitHub repository `PMF57/BCQM_VII` as the baseline state.

2. Stage–2 objective and design choice

Stage–2 begins from the Stage–1 conclusion that geometry on the short active slice $V_{\text{active}}(t)$ is "yarn-like" (finite, fast mixing) and that a persistent "cloth" geometry object must be constructed for metric/dimension tests. The chosen direction was:

- persistence via (3) lockstep-bundle support and (4) survival-under-perturbation validation;

- minimalism-preserving persistence proxy via *concurrency* without storing mutable weights on events/edges;
- **bin-based** (not tick-based) concurrency as the first implementation ("Stage–2a"), deferring sliding-window concurrency to a later robustness check.

3. Implementation bring-up (bin-based concurrency ledger)

A first Stage–2 patch introduced a *cloth ledger* and *cloth summary* in `RUN_METRICS`. An initial SyntaxError in `engine_vglue.py` (escaped docstring) was fixed (v0.1.1) and the bring-up run completed cleanly.

3.1 Bring-up run: concurrent-only persistence (baseline behaviour)

Config: W=100, N=8, n=0.8, seed 56796; bins=80; $w_{\text{lock}} = 0.10$; `min_concurrency`=2.

Two "persistence" thresholds were tested:

- **min_bin_hits=3** (original): core events were non-empty, but *core edges were empty*; ball growth on the edge core was degenerate. Interpretation: concurrent edges (two threads traversing the same transition within a bin) are real but too rare to repeat across bins at this scale.
- **min_bin_hits=1**: core edges became non-empty but appeared as disjoint two-node components (pair links), yielding trivial ball growth. Interpretation: concurrent edges exist but are not yet a connected backbone.

These results established a key Stage–2 diagnostic lesson: *edge concurrency alone is too sparse to define a connected cloth edge backbone* at the current run length/binning.

4. Refinement: "edges used by the lockstep core"

To obtain an edge-based cloth object without storing weights on primitives, a refinement was adopted:

- keep concurrent edges/events as a reinforcement diagnostic;
- additionally log *per-bin presence* of edges/events used by (i) all threads and (ii) the lockstep core members (based on overlap membership at w_{lock});
- define the persistent edge core from bin-hit counts of *used-by-core* edges rather than from concurrent edges only.

A first implementation of this refinement contained an indentation error and was corrected (v0.2.1). The corrected patch was then executed using the same bring-up configs.

4.1 Results under used-by-core persistence

Using the same physical regime (W=100, N=8, n=0.8, seed 56796):

- **min_bin_hits=2**: a non-zero persistent edge core appeared (dozens of edges), but the resulting edge-core component was still very small (ball-growth component size of only a few nodes), indicating that strict cross-bin repetition selects a tiny recurrent pocket at this scale.
- **min_bin_hits=1**: the used-by-core edge set formed a large connected component spanning the full core event set (non-degenerate ball growth), while the concurrent-edge diagnostic remained sparse. Interpretation: the used-by-core definition produces a meaningful "roads" cloth candidate, while concurrency remains useful as a reinforcement signal rather than the sole backbone criterion.

5. Immediate next steps proposed (for tomorrow)

The session concluded with two clear next actions:

1. **Ensemble stability for cloth:** repeat the used-by-core cloth extraction across a small seed set (e.g. 5 seeds) for N=8 at n=0.4 and n=0.8 (and then N=4), to assess the stability of cloth core size, ball-growth curves, and the concurrent-edge diagnostic.
2. **Survival metrics:** implement a lightweight survival measure across seeds (e.g. Jaccard overlap) for the cloth core. This requires either storing an explicit core edge list (small for $N \leq 8$) or storing a compact signature/hashing scheme to permit set comparison without bulky outputs. The preferred option will be chosen before coding.