

Design Note: Solving the Rails Tautology Problem

1. Problem Statement

In the current Tier-1 wiring, **rails echo the mapper** rather than independently validating. The flow looks like this:

- Mapper predicts a primitive (e.g., `deposit`).
- Synthetic latent generator fabricates a bump in the corresponding channel **because the mapper said so**.
- Rails run the matched filter, detect the fabricated bump, and "confirm" the mapper's guess.

This creates a **tautology**: - Mapper abstains → rails abstain. - Mapper says `deposit` → rails say `deposit`.

Rails add no independent evidence. Worse, if mapper hallucinates on nonsense (e.g., *"sing me a lullaby"*), rails will also hallucinate.

2. Why This is Dangerous

- **No safety guarantee**: Rails cannot suppress mapper hallucinations.
- **No true abstain**: Rails abstain only when mapper abstains.
- **Dead weight**: Rails become redundant; their matched filter and null calibration aren't being used as intended.

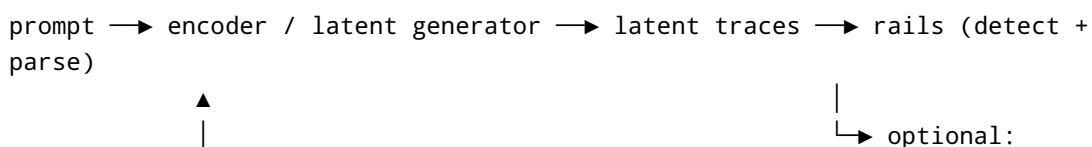
This undermines the core selling point of micro-LM: **0 hallucinations and deterministic abstains**.

3. Solution Strategy: Rails as Auditor

Rails must act as an **independent auditor** of mapper output. That requires:

1. **Independent latents**: Latent traces must be derived directly from text, *not* from mapper labels.
2. **Mapper as filter only**: Mapper narrows the candidate primitives (optional), but never injects energy.
3. **Matched filter + parser**: Rails validate bumps against relative + null thresholds, and order them in time.

Correct wiring



```
restrict_to(candidates from mapper)
    ↳ mapper (text → candidate set)
```

- **Rails decide from evidence.**
- **Mapper only filters channels.**
- **Nonsense prompts → noise-only traces → abstain.**

4. Implementation Plan

We trial this fix with a curated set of 200–500 prompts:

Train phase

1. **Train mapper:** Using (prompt, primitive) pairs, build a baseline embedding→label classifier.
2. **Harvest phrases:** From the same labeled set, mine n-grams that strongly align to primitives (e.g., deposit → {deposit, top up, add, put in}).
3. **Build prototype vectors:** Average embeddings of phrases per primitive to form prototype vectors $\{v_k\}$.

Test phase

For each test prompt: 1. **Mapper (optional):** Predict candidate primitives with confidence $\geq \tau_{\text{map}}$. 2. **Phrase spans:** Detect spans in the prompt that map to primitives (independent of mapper). 3. **Latent generator:** Encode prompt; compute similarity to prototypes $\{v_k\}$. Place shaped lobes in corresponding channels if above τ_{ood} ; otherwise generate **noise-only traces**. 4. **Rails:** Run matched filter + dual thresholds on these traces, restricted to mapper candidates if provided. 5. **Parser:** Sequence bumps into an ordered plan. 6. **Decision:** If no channel clears thresholds → abstain.

5. Expected Outcomes

- **OOD prompts:** Always abstain (no fabricated bumps).
- **Valid prompts:** Correct primitives and sequence detected.
- **Mapper hallucinations:** Caught and suppressed, since rails require independent bump evidence.
- **Metrics:**
 - Hallucination rate → 0%
 - Abstain rate → ~100% on junk prompts
 - Macro-F1 → ≥ 0.95 on curated test set

6. Key Takeaway

By decoupling rails from mapper labels, we restore rails as a **true auditor**: - Mapper guesses → rails validate. - No evidence → rails abstain. - Rails' matched filter + parser finally enforce the guarantees promised by the NGF design.

This eliminates the tautology and showcases the unique selling point of micro-LM: **deterministic, safe, 0-hallucination reasoning over a small set of primitives.**