EGWM Part 8: Experiment 8 – Frustration and Tool-Augmented World Bank

Mel Howard

December 2025

1 Introduction

The EGWM framework manages a bank of specialist world models using value channels ("feelings") such as confusion, competence, and elegance. These feelings govern structural actions like SPAWN, UPDATE, MERGE, and FORGET. In previous parts, the agent relied entirely on parametric world models: any failure was treated as a signal to grow or reshape the world bank.

However, not all problems are best solved by internal pattern-matching models. Some classes of tasks are inherently symbolic or deterministic (e.g., exact arithmetic, sorting, database lookups) and are more naturally handled by external tools. A general agent must therefore be able to recognise when structural edits are no longer the right solution and when it should instead acquire or invoke a specialised tool.

This part introduces a slow "frustration" channel and a tool-augmented world bank. Frustration measures the long-term cost of repeatedly failing to solve a task with structural edits alone. When frustration rises, the agent can trigger a new structural action, TOOL-ACQUIRE, that integrates an external non-parametric tool into its repertoire.

2 Experiment 8: Frustration-Driven Tool Acquisition

2.1 Setup

We consider an environment that mixes two families of tasks:

- **Pattern tasks**: two-dimensional classification problems that can be effectively solved by small parametric heads (as in previous EGWM experiments).
- **Symbolic tasks**: simple arithmetic or list-processing problems that are difficult for the small heads to learn reliably, but trivial for an external

tool (for example: computing $a \bmod b$ for integers, or returning the length of a sequence).

At each phase t, the environment samples a task instance from either the pattern family or the symbolic family. The agent maintains:

- A **world bank** $\mathcal{H}_t$ of small parametric heads (logistic or small MLPs) trained with gradient descent.
- A **tool bank** $\mathcal{T}_t$ of non-parametric tools. In this experiment we include a single perfect tool that can solve the symbolic tasks exactly (e.g., a calculator function).

Initially, the tool bank is empty and the agent does not know that an external tool exists. It only has the ability to allocate and train new heads in $\mathcal{H}_t$.

2.2 Feelings, frustration, and structural actions

For each pattern or symbolic task we compute the usual fast feelings: confusion, competence, and complexity, derived from the accuracy of scratch models and existing heads on the current phase. As before, a fast governor uses these feelings to choose between SPAWN and UPDATE on the world bank.

To detect when structural edits are not sufficient, we introduce a slow *frustration* channel F_t. Intuitively, frustration should rise when the agent repeatedly fails on a task family despite trying to allocate and train heads.

We define frustration as an exponential moving average of structural failure:

$$F_{t+1} = (1 - \alpha)F_t + \alpha\, \ell_t^{\text{struct}},$$

where $\alpha \in (0, 1)$ is a smoothing parameter and ℓ_t^{struct} is a scalar that captures how badly the current structural strategy is failing. For example,

$$\ell_t^{\text{struct}} = \max\{0, \ell_t - \ell_{\text{target}}\},$$

where ℓ_t is the loss (or $1-$ accuracy) on the current task and ℓ_{target} is a small tolerance. When the agent solves tasks well, ℓ_t^{struct} is near zero and frustration decays. When the agent repeatedly fails, ℓ_t^{struct} is positive and frustration accumulates.

We extend the structural action set with a new slow action:

- TOOL-ACQUIRE: integrate a new external tool into $\mathcal{T}_t$ and allow the routing policy to invoke this tool on future tasks of the same family.

The slow reflective layer monitors F_t and triggers TOOL-ACQUIRE when frustration crosses a threshold F_{thresh} and the agent has already attempted multiple SPAWN/UPDATE cycles without improvement.

At inference time, the routing problem is extended from “which world head?” to:

> Which combination of world head and tool should be used for this task?

For pattern tasks, the agent will typically route to heads in $\mathcal{H}_t$. For symbolic tasks, once a suitable tool has been acquired, the agent should learn to route directly to that tool and avoid wasting structural edits.

2.3 Conditions and baselines

We compare three regimes:

1. **No tools**: the agent only has access to the world bank $\mathcal{H}_t$ and never uses external tools. All failures must be addressed by structural edits (SPAWN, UPDATE).
2. **Always-tools**: the agent is given access to the tool bank from the start and may freely call the symbolic tool. There is no frustration signal; the routing policy is trained to pick heads or tools directly from experience, but there is no notion of structural failure accumulating over time.
3. **Frustration-driven tool acquisition**: the agent starts with no tools, attempts to solve tasks with structural edits, and uses the slow frustration channel to decide when to trigger TOOL-ACQUIRE. After acquisition, the routing policy can choose between heads and tools for each task.

We measure:

- Overall competence (mean accuracy across both task families).
- Pattern-task competence (accuracy restricted to pattern tasks).
- Symbolic-task competence (accuracy restricted to symbolic tasks).
- Structural complexity (mean number of heads in $\mathcal{H}_t$).
- Tool usage rate (fraction of symbolic tasks routed to the tool).

2.4 Result template

Once the experiment is implemented and run, results can be summarised in a table of the form:

Qualitatively, we expect the following pattern of behaviour:

- In the **no-tools** setting, the agent repeatedly spawns and updates heads on symbolic tasks, but competence remains limited. Frustration (if computed) would grow, but there is no mechanism to act on it. The world bank becomes unnecessarily large and symbolic accuracy lags behind the tool-based ceiling.

Regime	Overall acc.	Pattern acc.	Symbolic acc.	Mean # heads	Tool use
No tools	–	–	–	–	–
Always-tools	–	–	–	–	–
Frustration-driven	–	–	–	–	–

Table 1: Experiment 8: Frustration and Tool-Augmented World Bank. Numbers to be filled in after running the implementation. We expect the no-tools regime to struggle on symbolic tasks and compensate with excessive head growth, the always-tools regime to overuse the tool even when heads suffice, and the frustration-driven regime to approach tool-level performance on symbolic tasks while maintaining a compact world bank and using the tool primarily when structural edits fail.

- In the **always-tools** setting, the agent can achieve high symbolic accuracy by routing directly to the tool, but may overuse the tool even for tasks that could be solved by heads. Structural complexity may remain low, but the agent has no notion of *when* tools are actually needed.
- In the **frustration-driven** setting, the agent initially behaves like the no-tools regime, but as frustration accumulates on symbolic tasks it triggers TOOL-ACQUIRE. After acquisition, routing for symbolic tasks shifts to the tool, while pattern tasks continue to be handled by heads. The result should be high competence on both task families, a compact world bank, and tool usage that is concentrated on the genuinely hard symbolic cases.

2.5 Discussion

This experiment extends EGWM from a purely parametric architecture to a hybrid system that can recognise when structural edits are no longer the right response to failure. By introducing a slow frustration channel and a TOOL-ACQUIRE action, the agent gains the ability to:

- Distinguish between "I need a new world" and "I need a new kind of capability".
- Offload intrinsically symbolic or deterministic subtasks to specialised tools, while keeping its world bank focused on pattern recognition.
- Learn, over time, which task families benefit from tool use and which are better handled by internal models.

Together with the adaptive inner voice from Experiment 7, frustration and tool acquisition move EGWM closer to an architecture that not only manages its internal structure, but also understands when to extend that structure outward by integrating new tools when its own patterns are not enough.