Behavior Theory: The Static Collapse Theorem

Heath Fortin

May 29, 2025

1 Definitions

- **Task Universe:** $\mathcal{P} = \{P_1, P_2, \dots\}$ (unbounded problem set)
- **Solution Space:** For each task P_i, valid outputs lie in $\mathcal{S}_i \subseteq \mathcal{O}$
- **Agent:** System with parameters ω where $|\omega| \leq K < \infty$ (finite capacity)
- **Static System:** Parameters ω fixed after deployment

2 Core Axioms

1. **A1. Task-Oblivious Execution:** Agent processes fixed input x_* with *no task-identifying information*
2. **A2. Monolithic Output Requirement:** Agent emits single output $y \in \mathcal{O}$ satisfying *all* tasks:
$$y \in \bigcap_{i=1}^{n} \mathcal{S}_i$$
3. **A3. Independent Solution Sets:** $\mu(\mathcal{S}_i) \leq \rho < 1$ and μ-independent:
$$\mu\left(\bigcap_{i=1}^{n} \mathcal{S}_i\right) \leq \prod_{i=1}^{n} \mu(\mathcal{S}_i) \leq \rho^n$$
Note. This assumption applies to solution **sets** $\mathcal{S}_i$, not to individual solutions $y \in \mathcal{O}$. Solutions may overlap or be structurally related even when the sets are measure-independent.

3. Interpretation Notes

- **Task Information:** Input x may include task-identifying information. However, the agent's behavior is fully determined by fixed parameters ω. Thus, inclusion of task data does not affect the static nature of execution.

- **Monolithic Output:** The requirement that $y \in \bigcap \mathcal{S}_i$ reflects the limitation of fixed-capacity systems rather than a restriction on how tasks are presented. Even with full task context in x, the agent cannot dynamically reconfigure its behavior.
- **Worst-Case Assumption:** Axiom A3 models the exponential collapse under task-separation. If solution sets are not independent, collapse may occur more slowly, but it remains inevitable in static systems without capacity growth.

3 Main Theorem

[Static Collapse] For any static agent facing n tasks under A1–A3:

$$\Pr\left[y \in \bigcap_{i=1}^{n} \mathcal{S}_i\right] \leq \rho^n \quad \text{(single agent)}$$

and for the entire capacity-K agent class:

$$\Pr\left[\exists\, \text{successful agent}\right] \leq 2^K \rho^n \xrightarrow{n\to\infty} 0$$

Single-agent bound follows from A3. Class bound: 2^K agents via pigeonhole, union bound over ρ^n.

4 Recursive Collapse Corollary

Let $h : \mathcal{X} \to \mathcal{P}$ be a static task-classifier with $|\omega_h| \leq K_h$. Under analogous axioms:

$$\Pr\left[h(x_*) \in \bigcap_{k=1}^{n} \mathcal{T}_k\right] \leq 2^{K_h} \tau^n \to 0$$

where $\mathcal{T}_k$ are valid ID sets. *No static subsystem escapes collapse.*

5. Broader Implications

The collapse theorem demonstrates that static systems cannot scale indefinitely across unbounded tasks. Even in optimistic scenarios where task overlap exists, performance degrades without growth. True generalization thus requires agents whose internal structure or memory can change over time.

This implies a boundary between shallow pattern-matching and intelligence:

Static systems simulate intelligence; dynamic systems embody it.

5 Dynamic Intelligence Principle

- **Escape Hatch:** Agents avoiding collapse *must* grow capacity after deployment:

Memory-augmented networks	(e.g., external memory)
Runtime module synthesis	(e.g., neural weight growth)
On-the-fly architecture change	(e.g., attention pathway creation)

- **Collapse Test:**

$$\text{Does } |\omega_t| \uparrow \text{ after deployment?} \begin{cases} \text{Yes} & \Rightarrow \text{Dynamic} \Rightarrow \text{No collapse} \\ \text{No} & \Rightarrow \text{Static} \Rightarrow \text{Inevitable collapse} \end{cases}$$

Edit notes : There seems to be confusion relating to 3 key facts

1:

The importance of independence of solution sets is to not deal with the trivial case that all problems are the same

2:

This theory states ALL Static systems will fail; metalearning, hypernetworkers anything thats not continuosly learning will face an ever increasing set of solutions to which the agent cannot distinguish

3:

I refute task information as it assumes an objective truth to all problems. Godel's should be the obvious example