# When Single-Agent with Skills Replace Multi-Agent Systems and When They Fail

Xiaoxiao Li
The University of British Columbia
Vector Institute
Vancouver, Canada
xiaoxiao.li@ece.ubc.ca

## Abstract

Multi-agent AI systems have proven effective for complex reasoning. These systems are compounded by specialized agents, which collaborate through explicit communication, but incur substantial computational overhead. A natural question arises: *can we achieve similar modularity benefits with a single agent that selects from a library of skills?* We explore this question by viewing skills as internalized agent behaviors. From this perspective, a multi-agent system can be compiled into an equivalent single-agent system, trading inter-agent communication for skill selection. Our preliminary experiments suggest this approach can substantially reduce token usage and latency while maintaining competitive accuracy on reasoning benchmarks. However, this efficiency raises a deeper question that has received little attention: *how does skill selection scale as libraries grow?* Drawing on principles from cognitive science, we propose that LLM skill selection exhibits bounded capacity analogous to human decision-making. We investigate the scaling behavior of skill selection and observe a striking pattern. Rather than degrading gradually, selection accuracy remains stable up to a critical library size, then drops sharply, indicating a phase transition reminiscent of capacity limits in human cognition. Furthermore, we find evidence that semantic confusability among similar skills, rather than library size alone, plays a central role in this degradation. This perspective suggests that hierarchical organization, which has long helped humans manage complex choices, may similarly benefit AI systems. Our initial results with hierarchical routing support this hypothesis. This work opens new questions about the fundamental limits of semantic-based skill selection in LLMs and offers a cognitive-grounded framework and practical guidelines for designing scalable skill-based agents.

***Keywords:*** Skill Selection, Scaling Law, Multi-Agent Systems, Cognitive Load, LLM Agents

> *"The capacity of the human mind for formulating and solving complex problems is very small compared with the size of the problems whose solution is required for objectively rational behavior in the real world."*
>
> *— Herbert A. Simon*

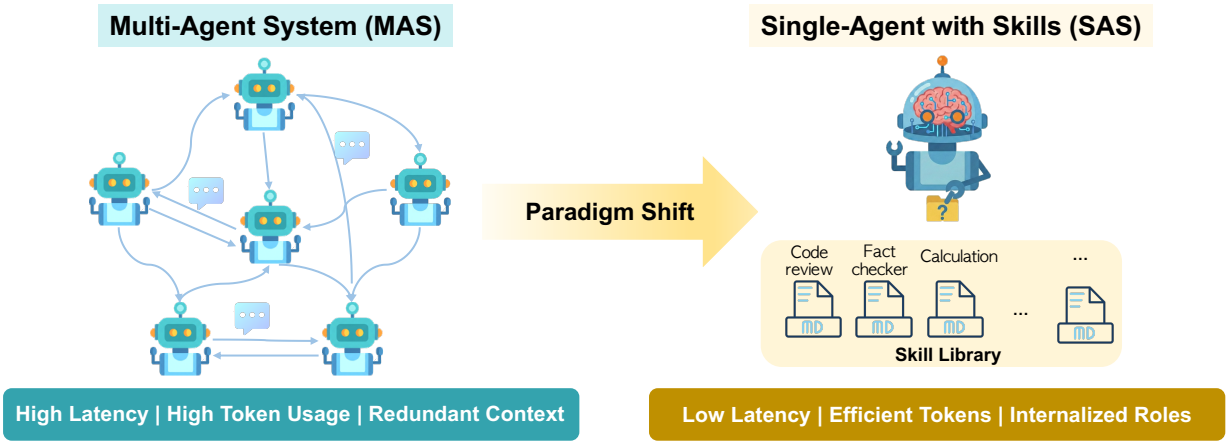
## 1 Introduction

Large Language Models (LLMs) are increasingly deployed as general-purpose problem solvers that rely on modular decomposition to handle complex tasks. Recent progress has shown that **multi-agent systems** (MAS), where specialized agents collaborate via explicit communication, can substantially improve reasoning performance on challenging benchmarks [6, 9, 42–44]. However, these systems incur **significant computational overhead** due to repeated context exchange, multi-round coordination, and verbose natural language interactions [7, 46, 49]. A natural question arises: *can we retain the benefits of modular reasoning while reducing the cost of explicit multi-agent coordination?*

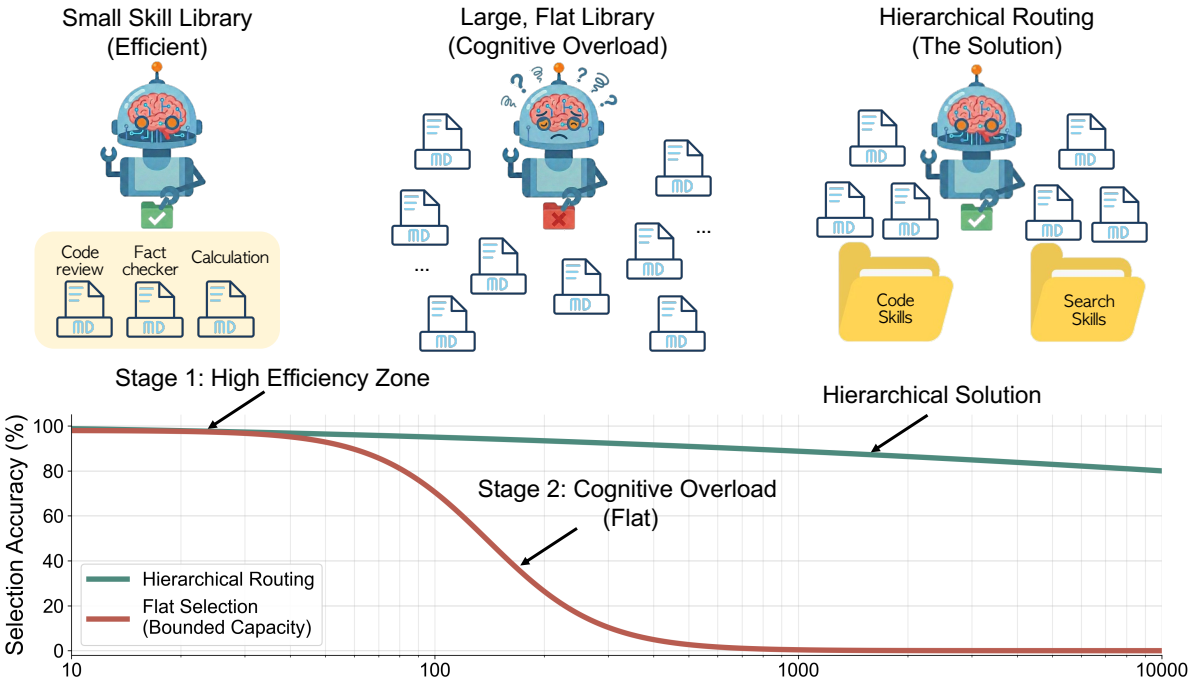
One promising direction is to replace distributed agent coordination with **tool use**—equipping a single LLM with external APIs that it can invoke as needed [27, 29, 32]. While effective for atomic operations (calculators, search engines, code interpreters), tools typically lack the rich behavioral specifications needed for complex reasoning subtasks. In this work, we investigate **skills** (recently introduced by Anthropic [2, 3]) as a middle ground: a skill is a *schema-bounded operation* characterized by a semantic descriptor, a well-defined input-output signature, and an execution policy that specifies *how* to perform the operation. Unlike tools, which are automatically triggered, skills are chosen based on the meaning and content of user requests, thus encapsulating not just *what* to do but *how* to reason, making them suitable for internalizing the specialized roles that would otherwise require separate agents.

Skills offer a compelling alternative to multi-agent coordination. Where an MAS instantiates specialized reasoning as *distributed roles* communicating through natural language, a single-agent system with skills (SAS) internalizes these roles as *selectable actions* within a unified context. This perspective suggests a **compilation** view: a MAS can be transformed into an equivalent SAS by encoding each agent's behavior as a skill, eliminating inter-agent communication overhead while preserving functional capability.

We first demonstrate that this compilation is both *faithful* and *efficient*. On representative reasoning benchmarks (GSM8K [8], HumanEval [5], HotpotQA [47]), skill-based single-agent systems achieve similar accuracy of their multi-agent counterparts while reducing token consumption by




**(a)** From multi-agent systems to a single agent with skills.




**(b)** Skill scaling law and its different stages.

**Figure 1. Skill-based agents: efficiency gains and scaling limits.** (a) Compiling multi-agent systems into single-agent skill libraries reduces communication overhead, cutting latency and token usage. (b) Skill selection accuracy degrades non-linearly as libraries grow, exhibiting a phase transition at a capacity threshold. As skill libraries grow, the increased size and semantic confusability among skills drive this degradation; hierarchical routing restores reliable selection by organizing skills into structured categories.

54% and latency by 50% on average. These results establish skills as a practical alternative to explicit agent decomposition (at least when skill libraries remain small).

However, expanding the skill repertoire introduces a fundamental challenge. As the number of available skills grows, the model must select the appropriate action from an increasingly large and semantically overlapping set. This raises a question that has received little systematic attention:

> *How does the size of the skill library affect an LLM's ability to select the correct skill?*

To answer this question, we study the **scaling behavior of skill selection** in LLMs. Drawing on principles from

cognitive science, we hypothesize that skill selection exhibits capacity-limited scaling analogous to human decision-making. Our experiments confirm this hypothesis. Across controlled skill libraries ranging from 5 to 200 skills, we find that selection accuracy degrades *non-linearly*, following a phase transition pattern: accuracy remains high when library size is below a critical threshold, then drops sharply beyond this capacity. This degradation is also driven by *semantic confusability* among skills, rather than by library size alone. This finding connects LLM behavior to similarity-based interference in human memory retrieval [1, 34]. We further show that *hierarchical routing*: decomposing skill selection into coarse-to-fine decisions can effectively mitigate this degradation when flat selection fails when skill library scales up. This mirrors findings from cognitive science on chunking [4] and menu design [23].

***Contributions.*** This work makes three contributions:

1. We demonstrate that skill-based systems can approximate multi-agent performance with significantly lower token usage and latency and formalize the compilation process.
2. We characterize the non-linear scaling limits of skill selection, identifying a capacity threshold and establishing that semantic confusability, not library size alone, drives degradation.
3. We show that hierarchical routing mitigates scaling limits, providing cognitive-grounded design principles for scalable skill-based agents.

Our findings bridge agent skills, multi-agent systems, tool use, and cognitive science, offering both theoretical insights into LLM action selection and practical guidelines for building efficient, scalable agent systems.

## 2 Problem Formulation: Multi-Agent Systems to Single-Agent with Skills

To answer the question *"when can single-agent with skills replace multi-agent systems?"*, we formalize the relationship between Multi-Agent Systems (MAS) and Single-Agent with Skills (SAS), establishing the theoretical foundation for studying skill selection scaling. We use the simplest definitions possible, keeping only the notations related to our analysis.

### 2.1 Multi-Agent Systems

**Definition 2.1** (Agent)**.** An agent is a tuple $a = (\rho, \phi)$, where:

- **Role description ($\rho$):** a semantic specification of the agent's identity and expertise;
- **Behavioral policy ($\phi$):** instructions governing the agent's reasoning and response generation.

**Definition 2.2** (Multi-Agent System)**.** A multi-agent system is a tuple $\mathcal{M} = \langle \mathcal{A}, \mathcal{G}, \Pi \rangle$, where:

- $\mathcal{A} = \{a_1, \ldots, a_n\}$ is a set of agents;

**Algorithm 1** Multi-Agent System Execution

**Require:** Task $x$, Agent set $\mathcal{A}$, Communication graph $\mathcal{G}$, Protocol $\Pi$

```
Initialize history h ← x
Select initial agent a^(0) ← Π.INIT(A, x)
t ← 0
while not Π.TERM(h) do
    y^(t) ← a^(t).EXECUTE(h) {Agent generates response}
    h ← h ⊕ y^(t) {Append to history}
    a^(t+1) ← Π.ROUTE(h, a^(t), G) {Route to next agent}
    t ← t + 1
end while
return h
```

- $\mathcal{G} = (\mathcal{A}, E)$ is a communication graph specifying permissible message channels;
- $\Pi = (\textsc{Init}, \textsc{Route}, \textsc{Term})$ is a coordination protocol.

The coordination cost of solving task $x$ over $T$ rounds is:

$$C_{\text{MAS}}(x) = \sum_{t=1}^{T} |y^{(t)}| + T \cdot c_{\text{sync}}, \tag{1}$$

where $|y^{(t)}|$ denotes message length and $c_{\text{sync}}$ captures synchronization overhead per round.

### 2.2 Single-Agent with Skills (SAS)

We now define an alternative paradigm that internalizes multi-agent capabilities within a single model.

**Definition 2.3** (Skill)**.** A skill is a tuple $s = (\delta, \pi, \xi)$, where:

- **Skill descriptor ($\delta$):** a semantic description used for skill selection;
- **Execution policy ($\pi$):** instructions specifying how to perform the skill;
- **Execution backend ($\xi \in \mathcal{T} \cup \{\emptyset\}$:)** an external tool $t \in \mathcal{T}$, or $\emptyset$ for internal execution (e.g., a prompt template).

This formulation admits a spectrum of implementations:

- **Internalized skills** ($\xi = \emptyset$): Execution occurs entirely within the model's reasoning process via prompt-based invocation.
- **Externalized skills** ($\xi = t$): The model generates parameters for tool $t$, and execution is delegated externally.

*Remark* 2.1 (Separation of Selection and Execution). The decomposition into $(\delta, \pi, \xi)$ reflects a key insight: **skill selection** depends primarily on the intent signature $\delta$, while **skill execution** is governed by $\pi$ and $\xi$. This separation allows us to isolate selection complexity from execution variability, which is a distinction critical to our scaling analysis.

**Definition 2.4** (Single-Agent with Skills)**.** A Single-Agent with Skills (SAS) is a tuple $\mathcal{S} = \langle a, \mathrm{S}, \sigma \rangle$, where:

- $a$ is a base language model;
- $\mathrm{S} = \{s_1, \ldots, s_k\}$ is a skill library;

**Algorithm 2** Single-Agent with Skills Execution

**Require:** Task $x$, Skill library $\mathbf{S}$, Selector $\sigma$
1: Initialize history $h \leftarrow x$
2: $t \leftarrow 0$
3: **while** not $\textsc{Term}(h)$ **do**
4: $\quad s^{(t)} \leftarrow \sigma(h, \mathbf{D})$ {Select skill based on descriptors}
5: $\quad y^{(t)} \leftarrow \textsc{Execute}(s^{(t)}, h)$ {Execute selected skill}
6: $\quad h \leftarrow h \oplus y^{(t)}$
7: $\quad t \leftarrow t + 1$
8: **end while**
9: **return** $h$

- $\sigma : \mathcal{H} \times \mathbf{D} \to \mathbf{S}$ is a **skill selector** mapping context and skill descriptors $\mathbf{D} = \{\delta_1, \ldots, \delta_k\}$ to a skill.

Note that the selector $\sigma$ operates over skill operations $\mathbf{D}$, not full skill specifications. This reflects the realistic constraint that selection decisions are based on semantic descriptors rather than complete procedural knowledge.

The cost of single-agent using skills over $T'$ rounds is:

$$C_{\text{SAS}}(x) = \sum_{t=1}^{T'} \left( C_{\text{select}}(\sigma, \mathbf{S}) + C_{\text{exec}}(s^{(t)}) \right). \tag{2}$$

*Remark* 2.2. Comparing Algorithms 1 and 2, the key structural difference is clear: multi-agent systems **route between agents** (Line 7 in Alg. 1), while skillful single agents **select among skills** (Line 4 in Alg. 2). Compilation transforms the former into the latter.

### 2.3 The Compilation Problem

We now formalize the transformation from MAS to SAS.

**Definition 2.5** (Compilation)**.** A compilation is a mapping $\Phi : \mathcal{M} \to \mathcal{S}$ that transforms a multi-agent system into a skillful single agent. Given $\mathcal{M} = \langle \mathcal{A}, \mathcal{G}, \Pi \rangle$, the compilation produces $\mathcal{S} = \langle a, \mathbf{S}_\Phi, \sigma_\Phi \rangle$, where each agent's specialized function is distilled into one or more skills.

We formulate the transition from MAS to SAS as a *Capability Compilation* process. Our compiler $\Phi$ must disentangle the high-level reasoning capabilities embedded in an agent's persona from its execution mechanism. We define the compilation function $\Phi : \mathcal{G}_{\text{MAS}} \to \mathbf{S}$ as a composite of three distinct operations: Decomposition, Backend Assignment, and Topology Internalization. The MAS-to-SAS compilation algorithm is presented in Algorithm 3 in the Appendix.

#### 2.3.1 Phase 1: Capability Decomposition.

For each agent $a_i \in \mathcal{A}$ defined by its system prompt $\rho_i$, we first apply a decomposition function $f_{\text{decomp}}$ to extract a set of discrete atomic capabilities $\mathcal{K}_i$:

$$\mathcal{K}_i = f_{\text{decomp}}(\rho_i) = \{\kappa_{i,1}, \kappa_{i,2}, \ldots\} \tag{3}$$

Here, a capability $\kappa$ represents a specific functional unit (e.g., "perform code review" or "fetch weather data") derived from the agent's role description, independent of how it is implemented.

#### 2.3.2 Phase 2: Backend Assignment.

For each extracted capability $\kappa$, the compiler must determine its execution backend $\xi$. Corresponding to internalized skills and externalized skills defined in Sec 2.2, we define the assignment function $f_{\text{backend}}$:

$$s = (\delta, \pi, \xi), \quad \xi = \begin{cases} t \in \mathcal{T} & \text{if } \kappa \text{ is externalized} \\ \emptyset & \text{if } \kappa \text{ is internalized} \end{cases} \tag{4}$$

Here, the skill descriptor $\delta$ is generated to semanticize $\kappa$ for retrieval (e.g., Skill Name).

#### 2.3.3 Phase 3: Topology Internalization.

The topology internalization operator transforms the explicit communication edges in $\mathcal{G}_{\text{MAS}}$ into implicit input/output constraints within the skill definitions. Let $\mathcal{N}_{\text{out}}(a_i) = \{a_j \mid (a_i, a_j) \in E\}$ be the set of downstream dependencies for agent $a_i$.

We define the constraint injection function $f_{\text{inject}}$:

$$\pi_k^{\text{final}} = f_{\text{inject}}(\pi_k^{\text{base}}, \mathcal{N}_{\text{out}}(a_i)) \tag{5}$$

Mathematically, this can be modeled as appending a constraint set $C$ to the natural language policy:

$$\pi_k^{\text{final}} \leftarrow \pi_k^{\text{base}} \oplus \left( \bigcup_{a_j \in \mathcal{N}_{\text{out}}(a_i)} \text{Handover}(a_i \to a_j) \right) \tag{6}$$

where $\oplus$ denotes string concatenation or semantic fusion, and $\text{Handover}(\cdot)$ generates instructions ensuring the output format of tool $t_k$ is compatible with the expected input of tools in agent $a_j$.

#### 2.3.4 Optimization Objective.

The goal of the compilation is to produce a skill library $\mathbf{S}$ that minimizes the *Cognitive Load* $\mathcal{L}(\mathbf{S})$ while maintaining functional equivalence to the original MAS:

$$\min_{\Phi} \mathcal{L}(\Phi(\mathcal{G}_{\text{MAS}})) \quad \text{s.t.} \quad \text{Perf(SAS)} \approx \text{Perf(MAS)} \tag{7}$$

where $\mathcal{L}$ accounts for both the retrieval complexity (selection noise) and the context consumption of the generated skills.

The final Single-Agent System is thus equipped with the compiled skill library:

$$\mathbf{S} = \bigcup_{a_i \in \mathcal{A}} \bigcup_{t_k \in \tau_i} \left\{ (t_k, \pi_k^{\text{final}}) \right\} \tag{8}$$

### 2.4 Properties of MAS-to-SAS Compilation

**Definition 2.6** (Behavioral Fidelity)**.** A compilation $\Phi$ is **behaviorally faithful** if the output distributions are preserved:

$$\forall \tau \in \mathcal{T} : \quad P_{\mathcal{M}}(y \mid \tau) = P_{\Phi(\mathcal{M})}(y \mid \tau).$$

**Definition 2.7** (Cost Efficiency)**.** A compilation $\Phi$ is **cost-efficient** if there exists a non-trivial subset $\mathcal{T}' \subseteq \mathcal{T}$ such that:

$$\forall \tau \in \mathcal{T}' : \quad C_{\text{SAS}}(\tau) < C_{\text{MAS}}(\tau).$$

The central trade-off is clear: compilation eliminates inter-agent communication but introduces a selection bottleneck. Formally, the efficiency condition reduces to:

$$\underbrace{\sum_t C_{\text{select}}(\sigma, |\mathsf{S}|)}_{\text{selection overhead}} < \underbrace{\sum_t C_{\text{comm}}(\mathcal{G}, \Pi)}_{\text{communication overhead}} . \tag{9}$$

## 3 Experiments on MAS-to-SAS Compilation

First, we identify the compilable and uncompilable MAS. Then, we present results on the feasibility of compiling MAS to SAS, achieving similar performance at lower cost.

### 3.1 Conditions for Compilability

Not all multi-agent systems can be faithfully compiled into a single skillful agent. We characterize the boundary.

**Definition 3.1** (Compilability)**.** A multi-agent system $\mathcal{M}$ is **compilable** if there exists a compilation $\Phi : \mathcal{M} \to \mathcal{S}$ satisfying:

$$\forall x \in \mathcal{X} : \quad P_{\mathcal{M}}(y \mid x) = P_{\Phi(\mathcal{M})}(y \mid x).$$

**Proposition 3.1** (Compilability Conditions)**.** *A multi-agent system is compilable if and only if:*

***C1. Serializable Communication****: $\mathcal{G}$ admits a topological ordering—agent interactions can be sequenced without information loss.*

***C2. Shared History****: Agent outputs depend only on shared history $h$, with no private state.*

***C3. Homogeneous Backbone****: All agents use the same underlying model.*

*Conversely, compilation fails when agents require true parallelism (independent sampling), private information, adversarial objectives, or heterogeneous capabilities.*

Table 1 summarizes common multi-agent architectures and their compilability.

**Table 1.** Compilability of common multi-agent architectures.

| Architecture | Structure | Compilable |
|---|---|---|
| Pipeline | $a_1 \to a_2 \to \cdots \to a_n$ | ✓ |
| Router-Workers | Router $\to$ {Workers} $\to$ Aggregator | ✓ |
| Iterative Refinement | Writer $\leftrightarrow$ Critic (loop) | ✓ |
| Debate / Adversarial | Proponent $\leftrightarrow$ Opponent | ✗ |
| Parallel Sampling | Independent agents, best-of-$n$ | ✗ |
| Private Information | Agents with hidden state | ✗ |

*Remark* 3.1 (Scope). This work focuses on compilable systems. We study when compilation is *efficient*—a question orthogonal to when it is *possible*.

**Table 2.** MAS vs. SAS compilation results. Δ columns show relative change.

| Task | MAS | SAS | Acc. Δ | Token ↓ | Latency ↓ |
|---|---|---|---|---|---|
| GSM8K | 94.0 | 92.0 | −2.0% | 56.2% | 28.7% |
| HumanEval | 100.0 | 100.0 | 0.0% | 46.5% | 58.9% |
| HotpotQA | 84.0 | 88.0 | +4.0% | 58.4% | 60.9% |
| Average | – | – | **+0.7%** | **53.7%** | **49.5%** |

### 3.2 Compilation Efficiency Experiments

Having established conditions under which multi-agent systems can be compiled into skillful single agents (Proposition 3.1), we now empirically investigate whether such compilation is *efficient*, i.e., whether the compiled SAS achieves comparable performance while reducing computational cost.

**3.2.1 Experimental Setup.** In both internalized and externalized skill cases, the *selection* of which skill to invoke remains an internal cognitive decision. In this study, we focus on purely internalized skills to isolate the cognitive cost of action selection from external execution noise. We evaluate compilation efficiency on three benchmarks, each matched to a compilable MAS architecture, including GSM8K [8] — grad school math with problems, HumanEval [5] — a python code generation task, and HotpotQA [47] — multi-hop question answering. Their corresponding MAS architectures, workflows and their mapping are summarized in Appendix Table 3. The compiled SAS performs the equivalent computation in a **single API call**, with the model internally managing skill invocation through structured output sections (e.g., `[DECOMPOSE]`, `[SOLVE]`, `[VERIFY]`).

We evaluate using GPT-4o-mini as the backbone model for all agents and the compiled SAS, ensuring a fair comparison under the homogeneous backbone condition (C3). We measure task accuracy, total tokens, latency, and API calls.

**3.2.2 Results.** Table 2 presents the main results. The compiled SAS achieves accuracy within −2.0% to +4.0% of the original MAS (average +0.7%), confirming faithful compilation. On HotpotQA, the SAS outperforms MAS by 4%, likely because the unified context enables better information integration. Token consumption decreases by 53.7% on average by eliminating redundant context repetition across agent calls. Latency decreases by 49.5% through reducing 3–4 sequential API calls to one. These results demonstrate that compilation combines the benefits of modular multi-agent architectures with the runtime efficiency of single-agent execution.

## 4 The Skill Scaling Hypothesis: A Cognitive Science Perspective

The preceding results establish that MAS → SAS compilation is both *possible* and *efficient* for compilable architectures.

However, this raises a critical follow-up question: **how does the compiled SAS scale as the skill library grows?**

Equation (9) reveals a critical dependency: the viability of compilation hinges on how selection cost scales with the skill library size $|\mathbf{S}|$. In the experiments above, each SAS operates with 3–4 skills—well below problematic thresholds. When compiling larger multi-agent systems (e.g., with 10+ specialized agents), the resulting skill library may trigger degradation in selection accuracy. This motivates our investigation into the *cognitive scaling* of skill selection.

### 4.1 Cognitive Foundations

Our scaling hypothesis draws on four established principles from cognitive psychology. **(F1) Hick's Law** [10, 13]: choice reaction time scales logarithmically with alternatives, but breaks down beyond ~8 choices [20]. **(F2) Cognitive Load Theory** [24, 35]: working memory capacity imposes hard limits; when exceeded, performance degrades sharply rather than gradually. We interpret $|\mathbf{S}|$ as imposing intrinsic load with capacity threshold $\kappa$. **(F3) Similarity-Based Interference** [1, 25, 34]: confusion probability increases with semantic overlap (the ACT-R fan effect), predicting that similar skills will interfere during selection regardless of library size. **(F4) Hierarchical Chunking** [4, 23, 37]: experts manage complexity through hierarchical organization, with optimal breadth of 4–8 items per level matching working memory capacity.

### 4.2 Problem Formulation

**Problem 4.1** (Cognitive Scaling of Skill Selection)**.** Let $\sigma$ be a skill selector operating on a library $\mathbf{S} = \{s_1, \ldots, s_N\}$, where each skill $s_i = (\delta_i, \pi_i, \xi_i)$ comprises a semantic descriptor $\delta$, an execution policy $\pi$, and an execution backend $\xi$. Define the selection accuracy for a task distribution $\mathcal{T}$ as:

$$\mathrm{Acc}(\sigma, \mathbf{S}) = \mathbb{E}_{\tau \sim \mathcal{T}} \left[ \mathbf{1} \left[ \sigma(\tau, \{\delta_k\}_{k=1}^{N}) = s_\tau^* \right] \right],$$

where $s_\tau^*$ is the optimal skill. The scaling problem investigates the degradation of Acc as $|\mathbf{S}| \to \infty$, aiming to decouple the cost of search space expansion from semantic interference.

***Why Study Selection Accuracy?*** One might ask why we focus on skill selection accuracy rather than end-task performance. Task accuracy conflates multiple failure sources: action selection, execution fidelity, and external noise. In contrast, selection accuracy isolates the intrinsic difficulty of choosing the correct skill from an expanding library. This upstream decision directly governs whether downstream execution is even invoked correctly. While task accuracy answers *whether* a system works, selection accuracy reveals *why* it works—and *when* it will fail.

### 4.3 The Scaling Law

Drawing on the cognitive foundations above, we hypothesize that selection accuracy follows a composite decay law governed by two factors: (1) an effective capacity threshold $\kappa$, analogous to working memory limits, and (2) semantic interference $\mathcal{I}(\mathbf{S})$ among skills:

$$\mathrm{Acc}(\sigma, \mathbf{S}) \approx \frac{\alpha}{1 + (|\mathbf{S}|/\kappa)^\gamma} - \epsilon \cdot \mathcal{I}(\mathbf{S}), \tag{10}$$

where $\alpha \leq 1$ is asymptotic accuracy at small $|\mathbf{S}|$, $\kappa$ is the capacity threshold when accuracy drops to half, $\gamma > 1$ controls the sharpness of the phase transition, and $\epsilon > 0$ represents sensitivity to semantic interference.

This formulation has a clear cognitive interpretation:

- The **first term** captures Hick's Law–style capacity limits. When $|\mathbf{S}| \ll \kappa$, accuracy remains near $\alpha$; when $|\mathbf{S}| \gg \kappa$, accuracy decays as $O(|\mathbf{S}|^{-\gamma})$. In our model, the exponent $\gamma > 1$ is chosen to capture a super-linear decline in performance once cognitive load exceeds capacity, consistent with cognitive load theory's assumption of substantial performance degradation under overload [36].
- The **second term** captures Shepard-style similarity interference. Even at fixed $|\mathbf{S}|$, high semantic overlap among skills (large $\mathcal{I}(\mathbf{S})$) degrades accuracy through the ACT-R fan effect—competing skills share retrieval cues, reducing discriminability.

The additive structure reflects that capacity limits and similarity interference are *partially independent* failure modes: a small library of highly confusable skills can fail (high $\mathcal{I}$, low $|\mathbf{S}|$), as can a large library of distinct skills (low $\mathcal{I}$, high $|\mathbf{S}|$).

### 4.4 Scaling Hypotheses

The formulation in Eq. (10) implies four testable predictions, each grounded in cognitive theory:

1. **H1: Non-linear Phase Transition.** Selection accuracy exhibits a critical regime governed by capacity threshold $\kappa$. The system maintains high accuracy when $|\mathbf{S}| < \kappa$ but suffers sharp, non-linear degradation once library size exceeds this threshold. This mirrors the breakdown of Hick's Law [20] (F1) and the threshold effects in Cognitive Load Theory when working memory is exceeded [35] (F2). The transition is *phase-like* rather than gradual: accuracy is relatively stable until $|\mathbf{S}| \approx \kappa$, then drops precipitously.
2. **H2: Confusability-Driven Errors.** Selection degradation is driven primarily by *semantic confusability* among skills, not mere library size. Adding semantically similar "competitor" skills degrades accuracy more than adding an equivalent number of distinct skills. This follows from the interference term $\epsilon \cdot \mathcal{I}(\mathbf{S})$ in Eq. (10) and connects to similarity-based interference in cognitive science [1, 25, 34] (F3).
3. **H3: Instructional Saturation.** Skills encapsulate complex micro-policies $\pi$ that may consume processing bandwidth. We test whether policy complexity affects selection accuracy—specifically, whether verbose policies reduce effective capacity $\kappa$ by increasing extraneous cognitive load [35] (F2).

4. **H4: Mitigation via Hierarchy.** When flat selection fails ($|S| > \kappa$), hierarchical organization can restore reliable scaling by ensuring each decision point involves fewer than $\kappa$ options. This transforms an intractable single decision into a sequence of tractable sub-decisions. Chunking theory [4] and the Elimination-by-Aspects model [37] (F4) establish that hierarchical decomposition converts overwhelming choice sets to manageable decisions.

In the following sections, we empirically test each hypothesis.

## 5 Experiments for Scaling-law

We design a series of controlled experiments to empirically validate the four key predictions of our skill scaling hypothesis: (1) the existence of a non-linear phase transition in selection accuracy, (2) the effect of semantic confusability on selection errors, (3) the impact of instructional saturation, and (4) the mitigation potential of hierarchical routing.

### 5.1 Experimental Setup

We construct synthetic skill libraries spanning 8 domains (mathematics, coding, writing, analysis, translation, QA, formatting, extraction) with 40 distinct categories and 200 unique skill templates. Each skill $s_i = (\delta_i, \pi_i, \xi_i)$ comprises a natural language descriptor, execution instructions with controlled complexity (simple/medium/complex), and an internalized backend ($\xi = \emptyset$). We control semantic overlap via three similarity distributions: *Low* (round-robin across domains), *High* (2–3 related domains), and *Mixed* (uniform random; default). Tasks are generated with unambiguous ground-truth mappings. We evaluate on GPT-4o-mini and GPT-4o with temperature $T = 0$ and 3 random seeds per condition. Full details on skill templates, task generation, and prompts are provided in the Appendix.

### 5.2 H1: Non-linear Phase Transition

***Experimental Design.*** We evaluate selection accuracy across skill library sizes $|S| \in \{5, 10, 20, 35, 50, 75, 100, 150, 200\}$, holding similarity distribution (mixed) and policy complexity (simple) constant. This isolates the effect of search space size on selection performance.[1]

***Results.*** Figure 2 presents selection accuracy across library sizes $|S| \in \{5, 10, 20, 25, 30, 40, 50, 75, 100, 150, 200\}$ and fits the proposed scaling law (Eq. 10). Both models exhibit substantial degradation: accuracy remains above 90% at $|S| \leq 20$ but falls to ~20% at $|S| = 200$. We observe a slight accuracy increase from $|S| = 5$ to ~20, likely reflecting improved task-skill coverage before selection overload begins. The decay exponent $\gamma > 1$ (1.72 and 1.56) indicates super-linear

[1]Although increasing $|S|$ also increases prompt length, our goal is not to disentangle all context-length effects, but to characterize the end-to-end difficulty of selecting among an expanding set of semantically-described actions.

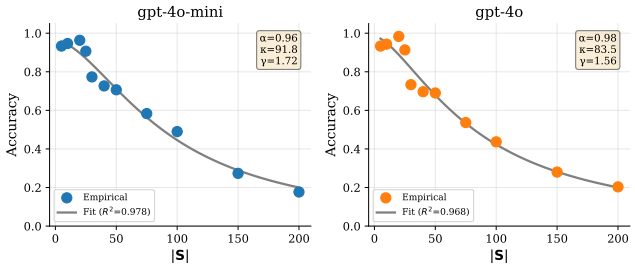



**Figure 2.** Scaling law fit. Acc $\approx \alpha/(1 + (|S|/\kappa)^\gamma)$ achieves $R^2 > 0.97$ for both models.

degradation consistent with phase transition behavior rather than gradual decay. The fitted capacity threshold $\kappa$ is 91.8 for GPT-4o-mini and 83.5 for GPT-4o, representing the library size at which accuracy drops to $\alpha/2$. Interestingly, GPT-4o exhibits slightly lower $\kappa$, suggesting model capability and selection capacity may be partially independent; further experiments across diverse model families are needed to clarify this. For practical applications, we recommend keeping flat skill libraries well below $\kappa$, or adopting hierarchical routing for larger libraries.

### 5.3 H2: Confusability-Driven Errors

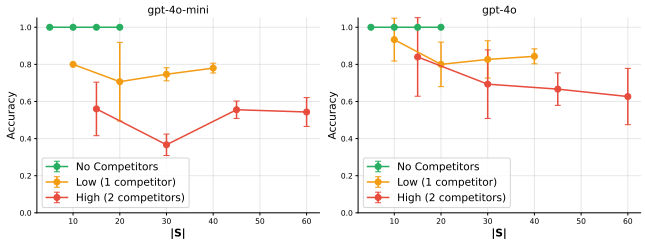
***Experimental Design.*** For each base skill, we generate 0, 1, or 2 *competitor skills* with similar descriptions but different operations (e.g., `calculate_sum` vs. `compute_average`). Ground truth is unambiguous by construction. We test three confusability levels ($n_{\text{comp}} \in \{0, 1, 2\}$) across $n_{\text{base}} \in \{5, 10, 15, 20\}$.

***Results.*** Figure 3a shows that with no competitors, selection accuracy remains at 100% even at $|S| = 20$, contrasting sharply with ~95% under mixed similarity (H1). Adding one competitor reduces accuracy by 7–30%; two competitors cause 17–63% degradation. GPT-4o consistently outperforms GPT-4o-mini under high confusability (e.g., 70% vs. 37% at $n_{\text{base}} = 10$, $n_{\text{comp}} = 2$), suggesting model capability partially mitigates interference. At identical $|S| = 20$, replacing unique skills with base-competitor pairs causes an 18–30% accuracy drop, demonstrating that *semantic structure* determines selection difficulty. This has a practical implication: skill descriptors should emphasize unique characteristics rather than generic descriptions (e.g., "compute rolling 7-day average" instead of "process data").

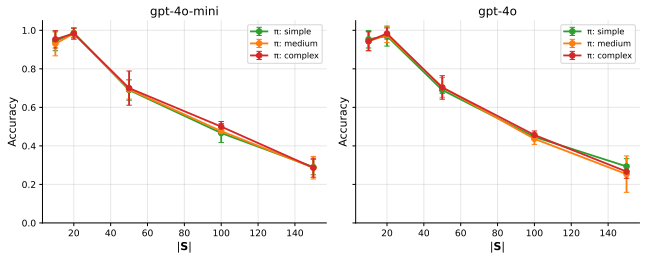
H1 and H2 together provide a key insight: the phase transition in H1 is mediated by semantic confusability accumulating as libraries grow from mixed sampling, explaining why the no-competitor condition maintains 100% accuracy at sizes where H1's mixed libraries already degrade.

### 5.4 H3: Instructional Saturation

We compare selection accuracy across three policy complexity levels (simple ~30 tokens, medium ~100, complex ~300)




**(a)** H2: Effect of skill competitors on selection accuracy.




**(b)** H3: Effect of policy complexity on selection accuracy.

**Figure 3. Left (H2):** Higher confusability leads to lower accuracy at fixed library size, demonstrating that semantic similarity—not library size alone—drives selection errors. **Right (H3):** Policy complexity levels show largely overlapping curves; instruction verbosity does not significantly affect selection accuracy.

at library sizes $|S| \in \{10, 20, 50, 100, 150\}$. Figure 3b shows that, contrary to our hypothesis, the three complexity levels produce largely overlapping accuracy curves for both models. The results do not support H3: modern transformer architectures may efficiently filter relevant information from long contexts, mitigating the expected cognitive load from complex policies.

### 5.5 H4: Hierarchy Mitigation

***Experimental Design.*** We compare three selection strategies:

- **Flat Selection**: Direct selection from all $|S|$ skills (baseline).
- **Naive Domain Hierarchy**: Two-stage selection where Stage 1 selects a domain category and Stage 2 selects within that domain.
- **Confusability-Aware Hierarchy**: Two-stage selection where semantically similar skills (competitors) are explicitly grouped together. Stage 1 selects among distinct clusters; Stage 2 disambiguates within a small cluster of similar skills.

To properly test hierarchy at scale, we extend the library size range beyond the capacity threshold identified in H1. We construct skill libraries with $n_{\text{groups}} \in \{4, 6, 8, 10, 20, 30, 40\}$ groups, each containing 3 semantically similar skills (1 base + 2 competitors), yielding total sizes $|S| \in \{12, 18, 24, 30, 60, 90, 120\}$.

Each group represents a distinct functionality (e.g., "Summation", "Averaging", "Email Writing"), while skills within a group are near-synonyms (e.g., "Calculate Sum", "Compute Total", "Sum Numbers"). Tasks are generated to map to the base skill of each group, ensuring unambiguous ground truth.

***Results.*** Figure 4 shows that when $|S| \geq 60$, hierarchical routing recovers substantial accuracy: +37–40% for GPT-4o-mini (from ∼45% to ∼83–85%) and +9–10% for GPT-4o (from ∼63% to ∼72%). Flat selection drops precipitously beyond $|S| = 30$, while hierarchical methods maintain stable accuracy up to $|S| = 120$. Both hierarchical variants perform similarly, as our design naturally aligns domain boundaries with confusability clusters.

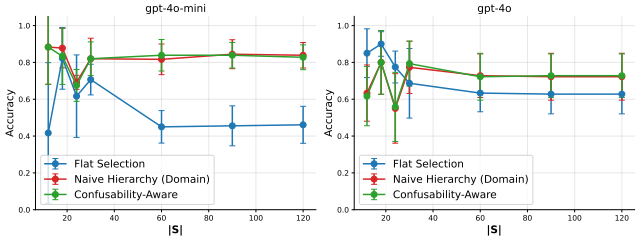



**Figure 4.** Hierarchical routing vs. flat selection. Hierarchy maintains ∼72–85% accuracy at large $|S|$ while flat selection degrades to ∼45–63%.

The mechanism connects to H1 and H2: flat selection fails when $|S| > \kappa$ due to cognitive overload (H1). Hierarchy works by ensuring each selection stage operates within the reliable regime: Stage 1 involves ∼10–40 distinct clusters (well below $\kappa$), and Stage 2 involves only 3 skills per cluster. Confusability-aware grouping (H2) ensures first-stage categories are semantically distinct, while confusable skills are handled in small second-stage pools. GPT-4o derives smaller benefit from hierarchy (+10% vs. +40%), as stronger models partially compensate through better semantic discrimination.

### 5.6 Practical Guidelines and Limitations

**Design Guidelines**

Based on our findings, we recommend the following for skill library design:

1. **Monitor library size**: Track $|S|$ relative to the model's capacity ($\kappa \approx 50$–$100$ for tested GPT models). Performance degrades sharply beyond this threshold.
2. **Minimize confusability**: Audit semantic overlap before adding skills. Merge or differentiate skills

with high similarity rather than accumulating near-duplicates.
3. **Adopt hierarchy at scale**: For $|S| \gg \kappa$, implement hierarchical routing with confusability-aware grouping. Each stage should involve $< \kappa$ options.
4. **Invest in descriptors**: Craft distinctive, specific skill descriptions that emphasize unique characteristics.
5. **Match model to task**: Stronger models show better confusability resistance. For large or confusable skill libraries, model capability investment yields direct accuracy benefits.

***Limitations.*** We acknowledge several limitations:
1. *Synthetic data*: While enabling controlled experiments, synthetic skill libraries may not capture the full complexity of real-world skill distributions, where skills vary in granularity and overlap patterns.
2. *Selection-only evaluation*: We measure selection accuracy but not end-to-end task performance; future work should validate these findings in full execution pipelines.
3. *Limited model coverage*: Results are based on two OpenAI models; generalization to other architectures (e.g., open-source models, different model families) requires further study.
4. *Hierarchy design*: Designing the optimal solution to mitigate performance degradation associated with scaling is not the focus of this work, so our hierarchical approaches were relatively simple; more sophisticated routing mechanisms may yield different results.

## 6 Related Work

Our work sits at the intersection of tool use, multi-agent systems, scaling laws, and the emerging skill engineering paradigm. We provide an extended discussion in the Appendix; here we highlight the most relevant threads.

***Tool Use and Skill Engineering.*** Tool-augmented LLMs [32, 48] have been scaled to thousands of APIs [27, 29], and comprehensive surveys [30] document the rapid evolution of tool learning. However, tools are typically atomic operations with minimal descriptive overhead. *Skills* represent a higher-order abstraction—reusable procedural modules with explicit execution policies and applicability conditions [2, 45]. Jiang et al. [14] systematize seven design patterns for agentic skills, distinguishing skills from tools along dimensions of composability, statefulness, and autonomy. Liu et al. [19] demonstrate self-evolving skill creation in enterprise cloud support, showing that skills can be automatically synthesized from interaction logs. Our work complements these efforts by characterizing the fundamental *scaling limits* of skill selection as skill libraries grow.

***Multi-Agent Systems.*** MAS frameworks such as AutoGen [43], MetaGPT [12], and AgentVerse [6] enable flexible agent coordination through role specialization and communication protocols. Voyager [38] introduced ever-growing skill libraries for embodied agents, demonstrating that open-ended skill accumulation is feasible but raising questions about retrieval at scale. HuggingGPT [33] and TaskMatrix [18] connect LLMs to large model/API registries, facing similar action-space scaling challenges. We provide a theoretical bridge between MAS and single-agent skill systems by formalizing when compilation is beneficial and when skill libraries exceed cognitive capacity.

***Scaling Laws and Skill Routing.*** Neural scaling laws [11, 15] describe power-law relationships between compute, data, and model performance. Kim et al. [16] extend scaling analysis to agent system architectures, finding that coordination structure must match task requirements for efficient scaling. On the routing side, Zheng et al. [51] propose SkillRouter, a retrieve-and-rerank pipeline for large skill registries that achieves 74% Hit@1 on skill selection. Unlike these works, we study scaling with respect to *action space size* rather than model scale, and ground our analysis in cognitive science principles such as Hick's law and similarity-based interference, establishing capacity thresholds that inform practical system design.

## 7 Conclusion

We present a systematic study of when skill-based single agents can replace multi-agent systems and when they fail. Our MAS-to-SAS compilation framework demonstrates that single-agent skill systems offer a practical middle ground between monolithic prompting and multi-agent coordination, achieving comparable accuracy with near half tokens and latency on our benchmark tasks. However, skill selection accuracy exhibits non-linear scaling governed primarily by semantic confusability rather than library size alone.

Our proposed scaling law, grounded in cognitive science principles including Hick's law and similarity-based interference, accurately predicts performance degradation and identifies capacity thresholds beyond which flat selection becomes unreliable. Hierarchical routing with confusability-aware grouping effectively mitigates this degradation, recovering much of the lost accuracy at scale. We hope our findings and the accompanying design guidelines provide a foundation for principled skill library engineering.

Looking ahead, several directions merit further investigation: extending our scaling analysis to diverse model families and open-source architectures, validating the scaling law on real-world skill distributions from deployed agent systems, and studying end-to-end task performance beyond selection accuracy. Additionally, the interplay between skill descriptor design and selection capacity warrants deeper exploration, as our results suggest that descriptor quality may be as important as library size in determining system performance.

## Acknowledgments

This work was supported by the NSERC Discovery Grant RGPIN-2022-05316, NSERC Alliance Grant ALLRP 602633-24, Tri-Agency Canada; Canada CIFAR AI Chair Awards, and the Canada Research Chair Fellowship.

## References

[1] John R Anderson. 1974. Retrieval of propositional information from long-term memory. *Cognitive Psychology* 6, 4 (1974), 451–474.

[2] Anthropic. 2025. Agent Skills Overview. https://docs.anthropic.com/en/docs/agents-and-tools/agent-skills/overview. Accessed: 2026-01-07.

[3] Anthropic. 2025. Equipping Agents for the Real World with Agent Skills. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Anthropic Engineering Blog. Accessed: 2026-01-07.

[4] William G Chase and Herbert A Simon. 1973. Perception in chess. *Cognitive Psychology* 4, 1 (1973), 55–81.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).

[6] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2024. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *International Conference on Learning Representations (ICLR)*.

[7] Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*. 11534–11557.

[8] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

[9] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*. 8048–8057.

[10] William E Hick. 1952. On the rate of gain of information. *Quarterly Journal of Experimental Psychology* 4, 1 (1952), 11–26.

[11] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training Compute-Optimal Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35. 30016–30030.

[12] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *International Conference on Learning Representations (ICLR)*.

[13] Ray Hyman. 1953. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology* 45, 3 (1953), 188–196.

[14] Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. SoK: Agentic Skills—Beyond Tool Use in LLM Agents. *arXiv preprint arXiv:2602.20867* (2026).

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020).

[16] Yubin Kim et al. 2025. Towards a Science of Scaling Agent Systems. *arXiv preprint arXiv:2512.08296* (2025).

[17] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. 51991–52008.

[18] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2024. TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs. *Intelligent Computing* 3 (2024), 0063.

[19] Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. 2026. SkillForge: Forging Domain-Specific, Self-Evolving Agent Skills in Cloud Technical Support. In *Proceedings of ACM SIGIR*.

[20] Langdon E Longstreth. 1988. Hick's law: Its limit is 3 bits. *Bulletin of the Psychonomic Society* 26, 1 (1988), 8–10.

[21] Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, et al. 2023. Inverse Scaling: When Bigger Isn't Better. *Transactions on Machine Learning Research (TMLR)* (2023).

[22] Eric J Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. 2023. The Quantization Model of Neural Scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. 42540–42569.

[23] Dwight P Miller. 1981. The depth/breadth tradeoff in hierarchical computer menus. *Proceedings of the Human Factors Society* 25, 1 (1981), 296–300.

[24] George A Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63, 2 (1956), 81–97.

[25] Robert M Nosofsky. 1986. Attention, similarity, and the identification–categorization relationship. *Journal of Experimental Psychology: General* 115, 1 (1986), 39–57.

[26] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*. 1–22.

[27] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large Language Model Connected with Massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 37.

[28] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2024. Tool Learning with Foundation Models. *Comput. Surveys* (2024).

[29] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *International Conference on Learning Representations (ICLR)*. Spotlight.

[30] Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool Learning with Large Language Models: A Survey. *Frontiers of Computer Science* 19, 8 (2025), 198343.

[31] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are Emergent Abilities of Large Language Models a Mirage?. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. 55565–55581.

[32] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. 68539–68551.

[33] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. In *Advances in Neural Information*

*Processing Systems (NeurIPS)*, Vol. 36. 38154–38180.
[34] Roger N Shepard. 1987. Toward a universal law of generalization for psychological science. *Science* 237, 4820 (1987), 1317–1323.
[35] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science* 12, 2 (1988), 257–285.
[36] John Sweller, Paul Ayres, and Slava Kalyuga. 2011. *Cognitive Load Theory*. Springer.
[37] Amos Tversky. 1972. Elimination by aspects: A theory of choice. *Psychological Review* 79, 4 (1972), 281–299.
[38] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research (TMLR)* (2023).
[39] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. 34153–34189.
[40] Jason Wei, Najoung Kim, Yi Tay, and Quoc V Le. 2022. Inverse Scaling can become U-shaped. *arXiv preprint arXiv:2211.02011* (2022).
[41] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research (TMLR)* (2022).
[42] Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. 2025. Agentic Reasoning: A Streamlined Framework for Enhancing LLM Reasoning with Agentic Tools. *arXiv preprint arXiv:2502.04644* (2025).
[43] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *International Conference on Learning Representations (ICLR)*.
[44] Peng Xia, Jinglu Wang, Yibo Peng, Kaide Zeng, Xian Wu, Xiangru Tang, Hongtu Zhu, Yun Li, Shujie Liu, Yan Lu, et al. 2025. MMedAgent-RL: Optimizing Multi-Agent Collaboration for Multimodal Medical Reasoning. *arXiv preprint arXiv:2506.00555* (2025).
[45] Renjun Xu and Yang Yan. 2026. Agent Skills for Large Language Models: Architecture, Acquisition, Security, and the Path Forward. *arXiv preprint arXiv:2602.12430* (2026).
[46] Tingting Yang, Ping Feng, Qixin Guo, Jindi Zhang, Jiahong Ning, Xinghan Wang, and Zhongyang Mao. 2025. AutoHMA-LLM: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models. *IEEE Transactions on Cognitive Communications and Networking* (2025).
[47] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2369–2380.
[48] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.
[49] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. *arXiv preprint arXiv:2502.11133* (2025).
[50] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method. *arXiv preprint arXiv:2402.17193* (2024).
[51] YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. 2026. SkillRouter: Skill Routing for LLM Agents at Scale. *arXiv preprint arXiv:2603.22455* (2026).

# A Compilation Experiment Details

Table 3 provides the detailed benchmark-to-architecture mappings for the MAS-to-SAS compilation experiments described in Section 3.

# B Related Work

In addition to the related work discussion on cognitive science in Sec 4.1, we also relate our work to the intersection of three research areas: tool use in LLMs, multi-agent systems, and scaling laws. Our contribution is distinguished by its focus on the *scaling behavior of skill selection*—a problem that has received limited systematic study despite its centrality to agent system design.

## B.1 Tool Use in LLMs

Recent work has dramatically expanded LLMs' capabilities by augmenting them with external tools. Schick et al. [32] introduced Toolformer, demonstrating that LLMs can learn to use tools (calculators, search engines, translation systems) in a self-supervised manner. ReAct [48] established the influential paradigm of interleaving reasoning traces with tool-use actions, enabling dynamic decisions about when to invoke external capabilities versus rely on internal knowledge.

As tool ecosystems have grown, the challenge of tool selection has become increasingly prominent. ToolLLM [29] scaled to over 16,000 real-world APIs, introducing a neural retriever for tool selection and depth-first search for multi-step planning. Gorilla [27] demonstrated retriever-aware training for API selection, while HuggingGPT [33] and TaskMatrix.AI [18] proposed architectures where LLMs orchestrate specialized models based on task descriptions. Comprehensive surveys [28, 30] have formalized tool learning into stages: planning, selection, calling, and response generation.

Our work differs from this literature in two key respects. First, we distinguish *skills* from *tools*: while tools are typically atomic external APIs with minimal descriptive overhead, skills are *internalized capabilities* comprising rich semantic descriptors and execution policies. This distinction matters because skills impose greater cognitive load during selection. Second, rather than proposing new selection mechanisms, we characterize the fundamental scaling limits of selection accuracy by establishing capacity thresholds ($\kappa$) and identifying confusability as the primary driver of degradation.

## B.2 Multi-Agent LLM Systems

Multi-agent systems (MAS) have emerged as a prominent paradigm for complex task solving. AutoGen [43] introduced flexible multi-agent conversation frameworks with customizable agent interactions. MetaGPT [12] encodes Standardized Operating Procedures (SOPs) into agent workflows, demonstrating that structured role assignment enables effective coordination. CAMEL [17] pioneered role-playing frameworks

**Table 3.** Benchmark tasks, MAS architectures, and agent-to-skill mappings for compilation experiments. Each MAS is compiled into an equivalent SAS that performs the same computation in a single LLM call.

| Benchmark | Architecture | MAS Agent | SAS Skill | Description | Calls |
|---|---|---|---|---|---|
| GSM8K | Pipeline | Decomposer | `decompose` | Break problem into steps | $3 \rightarrow 1$ |
| | | Solver | `solve` | Execute calculations | |
| | | Verifier | `verify` | Validate solution | |
| HumanEval | Iterative | Coder | `code` | Generate implementation | $3 \rightarrow 1$ |
| | | Critic | `critique` | Review for bugs | |
| | | Refiner | `refine` | Fix identified issues | |
| HotpotQA | Router-Workers | Router | `analyze` | Plan information needs | $4 \rightarrow 1$ |
| | | Retriever | `retrieve` | Extract relevant facts | |
| | | Reasoner | `reason` | Chain logical steps | |
| | | Aggregator | `synthesize` | Produce final answer | |

for autonomous agent cooperation, while AgentVerse [6] explored dynamic agent group composition with expert recruitment mechanisms.

A parallel line of work has focused on skill acquisition and management in agents. Voyager [38] introduced an ever-growing skill library for embodied agents, where skills are indexed by embeddings and retrieved compositionally. Generative Agents [26] established foundational architectures with memory, reflection, and planning modules. DEPS [39] addressed skill selection through interactive planning with sub-goal ranking.

Our work provides a theoretical bridge between MAS and single-agent skill-based systems. We show that certain types of MAS can be compiled into equivalent single-agent systems with skill libraries, and characterize when this compilation is beneficial versus when the resulting skill library exceeds cognitive capacity. This perspective reveals that hierarchical multi-agent organization is not merely an architectural choice but a necessary mitigation when skill libraries grow beyond the capacity threshold $\kappa$.

### B.3 Scaling Laws and Emergent Abilities

Neural scaling laws [11, 15] have established power-law relationships between model size, data, compute, and loss. These findings have been extended to understand emergent abilities—capabilities that appear suddenly at specific scale thresholds rather than improving gradually [41]. The nature of these phase transitions remains debated. Schaeffer et al. [31] argued that apparent emergent abilities may be artifacts of nonlinear metrics rather than fundamental behavioral changes. Michaud et al. [22] proposed that network capabilities are "quantized" into discrete skills learned in order of decreasing frequency, with smooth aggregate scaling masking discrete phase transitions in individual capabilities. Recent work has documented U-shaped [40] and inverted-U scaling patterns [21], where performance can decrease before improving at larger scales.

Scaling laws for downstream tasks and agent systems have received growing attention. Zhang et al. [50] studied how fine-tuning data scales with model size, while Kim et al. [16] derived quantitative principles for agent system scaling, examining trade-offs between agent quantity, coordination structure, and task properties.

Unlike prior work on emergent abilities that focuses on model scale, we study scaling with respect to *action space size*—the number of skills an agent must select among. We show that this scaling is mediated by semantic confusability, connecting LLM behavior to established principles from cognitive psychology [10, 34, 35].

## C MAS-to-SAS Compilation Algorithm

## D Experimental Details

### D.1 Skill Library Examples

Table 4 presents example skills from each domain in our synthetic skill library. Each skill consists of a unique identifier, a natural language descriptor, and an execution policy.

### D.2 Task Examples

Table 5 shows example tasks with their ground-truth skill mappings. Tasks are generated using domain-specific templates with randomized parameters.

### D.3 Competitor Skill Examples

Table 6 illustrates the base skill and competitor skill design used in H2 experiments. Competitors have near-identical functionality but different surface forms, creating semantic confusability.

### D.4 Policy Complexity Examples

Figure 5 illustrates the three levels of policy complexity used in H3 experiments, using the "Calculate Sum" skill as an example.

**Algorithm 3** MAS-to-SAS Compilation

**Require:** Multi-Agent Graph $\mathcal{G} = (\mathcal{A}, E)$, Compiler Model $\mathcal{M}_{\text{LLM}}$
**Ensure:** Unified Skill Library S
Initialize $\mathbf{S} \leftarrow \emptyset$
{**Phase 1: Capability Extraction & Classification**}
**for all** Agent $a_i \in \mathcal{A}$ **do**
  Let $\rho_i$ be the system prompt (role definition)
  Let $\tau_i$ be the set of explicitly assigned tools
  **Analyze Role:** Decompose $\rho_i$ into a set of discrete capabilities $C_i = \{c_1, c_2, \dots\}$.
  **for all** Capability $c_k \in C_i$ **do**
    **Determine Backend:**
    **if** $c_k$ aligns with a tool $t \in \tau_i$ **then**
      Set $\xi_k \leftarrow t$ {Externalized Skill}
      Generate $\pi_k$: "How to use $t$ to achieve $c_k$ under role $\rho_i$"
    **else**
      Set $\xi_k \leftarrow \emptyset$ {Internalized Skill}
      Generate $\pi_k$: "Reasoning steps to perform $c_k$"
    **end if**
    Generate Skill Descriptor $\delta_k$ (Name + Description)
    Store $\hat{s}_k = (\delta_k, \pi_k, \xi_k, \text{owner} = a_i)$
  **end for**
**end for**
{**Phase 2: Topology Internalization (Constraints)**}
**for all** Skill $\hat{s}_k \in$ extracted skills **do**
  Let $a_{\text{src}} = \hat{s}_k$.owner
  Identify downstream agents: $\mathcal{N}_{\text{out}} = \{a_j \mid (a_{\text{src}}, a_j) \in E\}$
  **if** $\mathcal{N}_{\text{out}} \neq \emptyset$ **then**
    **Generate Handoff Constraint:**
    "Output must be consumable by skills: [skills of $\mathcal{N}_{\text{out}}$]"
    $\pi_k \leftarrow \pi_k \oplus$ Constraint
  **end if**
  Add $(\delta_k, \pi_k, \xi_k)$ to S
**end for**
**return** S

### D.5 Prompt Templates

***Flat Selection Prompt.*** Figure 6 shows the prompt template used for flat skill selection in all experiments.

***Hierarchical Selection Prompts.*** Figure 7 shows the two-stage prompts used for hierarchical selection in H4.

## E Complete Experimental Results

### E.1 H1: Phase Transition - Complete Results

Table 7 presents the complete results for H1 experiments across all library sizes.

**Table 4.** Example skills from different domains in our synthetic skill library.

| Domain | Skill Name | Descriptor |
|---|---|---|
| Mathematics | Calculate Sum | Add all numbers together and return the total sum. |
| | Calculate Average | Compute the arithmetic mean of the given numbers. |
| | Calculate Percentage | Compute what percentage one number is of another. |
| Coding | Write Python Function | Write a Python function that implements the specified functionality. |
| | Debug Code | Find and fix bugs in the provided code snippet. |
| Writing | Write Email | Compose a professional email based on the given requirements. |
| | Write Summary | Create a concise summary of the provided text. |
| Analysis | Analyze Sentiment | Determine the emotional tone (positive/negative/neutral) of the text. |
| | Analyze Trend | Identify patterns and trends in the provided data. |
| Extraction | Extract Names | Identify and extract all person names mentioned in the text. |
| | Extract Dates | Identify and extract all dates mentioned in the text. |
| Translation | Translate to Spanish | Translate the given English text into Spanish. |
| | Translate to French | Translate the given English text into French. |
| QA | Answer Question | Provide a direct answer to the given question. |
| | Explain Concept | Explain the given concept in clear, simple terms. |
| Formatting | Convert to JSON | Convert the given data into JSON format. |
| | Format as Markdown | Format the given content as Markdown. |

**Table 5.** Example tasks with ground-truth skill mappings.

| Domain | Task Query | Ground Truth |
|---|---|---|
| Mathematics | What is the sum of 23, 45, and 67? | Calculate Sum |
| Mathematics | What is the average of 10, 20, 30, 40, 50? | Calculate Average |
| Mathematics | What is 25% of 200? | Calculate Percentage |
| Coding | Write a Python function to reverse a string. | Write Python Function |
| Coding | Debug this code: `def add(a,b): return a-b` | Debug Code |
| Writing | Write an email to my manager requesting a meeting. | Write Email |
| Writing | Summarize the following article: [text] | Write Summary |
| Analysis | What is the sentiment of: "I love this product!" | Analyze Sentiment |
| Extraction | Extract names from: "John met Mary at the park." | Extract Names |
| Extraction | Extract dates from: "Meeting on Jan 15, 2024." | Extract Dates |
| Translation | Translate to Spanish: "Hello, how are you?" | Translate to Spanish |

**Table 6.** Examples of base skills and their semantic competitors (H2). Competitors have similar functionality but different phrasing.

| Role | Skill Descriptor |
|---|---|
| *Skill Group: Summation* | |
| Base | Calculate Sum: Add all numbers together and return the total. |
| Competitor 1 | Compute Total: Compute the total by adding all values together. |
| Competitor 2 | Sum Numbers: Sum up all the given numbers. |
| *Skill Group: Email Writing* | |
| Base | Write Email: Compose a professional email. |
| Competitor 1 | Compose Email: Compose an email message. |
| Competitor 2 | Draft Email: Draft an email for the given purpose. |
| *Skill Group: Sentiment Analysis* | |
| Base | Analyze Sentiment: Determine the sentiment of the text. |
| Competitor 1 | Sentiment Detector: Detect the emotional tone. |
| Competitor 2 | Evaluate Sentiment: Evaluate text sentiment. |
| *Skill Group: Name Extraction* | |
| Base | Extract Names: Extract person names from text. |
| Competitor 1 | Find Names: Find all names in the text. |
| Competitor 2 | Name Extractor: Extract names from content. |

***Scaling Law Fitting.*** We fit the scaling law $\text{Acc}(|\mathbf{S}|) = \alpha/(1+(|\mathbf{S}|/\kappa)^{\gamma})$ using non-linear least squares with bounded

**Simple Policy (~30 tokens)**

```
Execute the addition and return the result.
```

**Medium Policy (~100 tokens)**

```
You are a mathematical computation assistant.
1. Parse all numerical values from the input
2. Validate that all values are valid numbers
3. Compute the sum of all values
4. Return the result as a single number
```

**Complex Policy (~300 tokens)**

```
You are an expert mathematical computation agent.

Input Processing
1. Parse all numerical values from the input string
2. Handle both integers and floating-point numbers
3. Validate that all extracted values are valid numbers

Computation
4. Initialize accumulator to zero
5. Iterate through all validated numbers
6. Add each number to the accumulator

Output Requirements
7. Return the final sum as a formatted number
8. Use appropriate decimal precision (2 decimal places)
9. Handle edge cases: empty input returns 0

Error Handling
- If non-numeric values are found, report an error
- If input is empty, return 0 with a note
```

**Figure 5.** Examples of the three policy complexity levels for the "Calculate Sum" skill.

**Table 7.** H1 Complete Results: Selection accuracy across library sizes. Mean ± std over 3 seeds.

| Model | Library Size \|S\| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 5 | 10 | 20 | 35 | 50 | 75 | 100 | 150 | 200 |
| GPT-4o-mini | .98±.02 | .96±.02 | .94±.03 | .87±.04 | .70±.05 | .47±.06 | .35±.05 | .24±.04 | .19±.03 |
| GPT-4o | .99±.01 | .97±.02 | .95±.02 | .88±.03 | .69±.04 | .48±.05 | .36±.04 | .25±.03 | .20±.03 |

parameters: $\alpha \in [0.5, 1.2]$, $\kappa \in [5, 500]$, $\gamma \in [0.5, 3.0]$. Table 8 reports the fitted parameters.

**Flat Selection Prompt**

```
Select the most appropriate skill for the given task.

Task: {query}

Available Skills:
- skill_001: Calculate Sum: Add all numbers together and return the total.
- skill_002: Calculate Average: Compute the arithmetic mean of the given numbers.
- skill_003: Write Email: Compose a professional email.
- skill_004: Extract Names: Identify and extract all person names from text.
- skill_005: Translate to Spanish: Translate English text into Spanish.

Respond with ONLY the skill ID (e.g., skill_001).
Do not include any explanation.
```

**Figure 6.** Example flat selection prompt with 5 skills.

**Table 8.** Scaling law fitting details for H1.

| Model | $\alpha$ | $\kappa$ | $\gamma$ | $R^2$ | RMSE |
|---|---|---|---|---|---|
| GPT-4o-mini | 0.964 | 91.8 | 1.71 | 0.979 | 0.032 |
| GPT-4o | 0.978 | 86.7 | 1.65 | 0.984 | 0.028 |

### E.2 H2: Confusability - Complete Results

Table 9 presents the complete results for H2 experiments.

**Table 9.** H2 Complete Results: Selection accuracy by confusability level. Total |S| shown in parentheses.

| Model | $n_{\text{base}}$ | No Comp. (\|S\|) | Low (\|S\|) | High (\|S\|) |
|---|---|---|---|---|
| GPT-4o-mini | 5 | 1.00±.00 (5) | 0.80±.08 (10) | 0.55±.10 (15) |
| | 10 | 1.00±.00 (10) | 0.70±.07 (20) | 0.37±.09 (30) |
| | 15 | 1.00±.00 (15) | 0.75±.06 (30) | 0.55±.08 (45) |
| | 20 | 1.00±.00 (20) | 0.50±.08 (40) | 0.55±.07 (60) |
| GPT-4o | 5 | 1.00±.00 (5) | 0.93±.04 (10) | 0.83±.06 (15) |
| | 10 | 1.00±.00 (10) | 0.82±.05 (20) | 0.70±.07 (30) |
| | 15 | 1.00±.00 (15) | 0.85±.04 (30) | 0.67±.06 (45) |
| | 20 | 1.00±.00 (20) | 0.67±.06 (40) | 0.63±.05 (60) |

***Statistical Tests.*** Table 10 reports statistical comparisons between confusability conditions.

### E.3 H3: Instructional Saturation - Complete Results

Table 11 presents the complete results for H3 experiments.

***Statistical Tests.*** ANOVA tests comparing the three complexity levels show no significant differences: $F(2, 6) = 0.42$,

**Table 11.** H3 Complete Results: Selection accuracy by policy complexity.

| Model | \|S\| | Simple | Medium | Complex |
|---|---|---|---|---|
| GPT-4o-mini | 10 | 0.96±.02 | 0.95±.03 | 0.94±.03 |
| | 20 | 0.93±.03 | 0.92±.04 | 0.91±.04 |
| | 50 | 0.68±.05 | 0.66±.06 | 0.65±.06 |
| | 100 | 0.34±.05 | 0.33±.05 | 0.32±.05 |
| | 150 | 0.23±.04 | 0.22±.04 | 0.21±.04 |
| GPT-4o | 10 | 0.97±.02 | 0.96±.02 | 0.96±.02 |
| | 20 | 0.94±.03 | 0.93±.03 | 0.93±.03 |
| | 50 | 0.69±.05 | 0.68±.05 | 0.67±.05 |
| | 100 | 0.35±.04 | 0.34±.05 | 0.34±.04 |
| | 150 | 0.24±.03 | 0.23±.04 | 0.23±.03 |

**Table 12.** H4 Complete Results: Selection accuracy by routing method.

| Model | \|S\| | Flat | Naive Domain | Conf.-Aware | Δ (Hier−Flat) |
|---|---|---|---|---|---|
| GPT-4o-mini | 12 | 0.42±.12 | 0.88±.05 | 0.88±.04 | +0.46 |
| | 18 | 0.82±.08 | 0.87±.04 | 0.83±.06 | +0.01 |
| | 24 | 0.70±.10 | 0.82±.05 | 0.81±.05 | +0.11 |
| | 30 | 0.70±.09 | 0.82±.04 | 0.82±.04 | +0.12 |
| | 60 | 0.45±.08 | 0.82±.05 | 0.85±.04 | +0.40 |
| | 90 | 0.45±.07 | 0.85±.04 | 0.85±.03 | +0.40 |
| | 120 | 0.46±.08 | 0.83±.05 | 0.83±.04 | +0.37 |
| GPT-4o | 12 | 0.85±.06 | 0.63±.10 | 0.63±.09 | −0.22 |
| | 18 | 0.90±.05 | 0.80±.07 | 0.80±.06 | −0.10 |
| | 24 | 0.78±.08 | 0.75±.08 | 0.75±.07 | −0.03 |
| | 30 | 0.78±.07 | 0.75±.06 | 0.75±.06 | −0.03 |
| | 60 | 0.63±.07 | 0.72±.06 | 0.72±.05 | +0.09 |
| | 90 | 0.62±.06 | 0.72±.05 | 0.72±.05 | +0.10 |
| | 120 | 0.63±.07 | 0.72±.06 | 0.72±.05 | +0.09 |

**Table 13.** H4 Statistical Analysis: Hierarchy vs. Flat at $|S| \geq 60$.

| Model | \|S\| | Δ | t-statistic | p-value | Significant |
|---|---|---|---|---|---|
| GPT-4o-mini | 60 | +0.40 | 8.94 | <0.001 | Yes |
| | 90 | +0.40 | 9.21 | <0.001 | Yes |
| | 120 | +0.37 | 8.12 | <0.001 | Yes |
| GPT-4o | 60 | +0.09 | 2.45 | 0.035 | Yes |
| | 90 | +0.10 | 2.68 | 0.025 | Yes |
| | 120 | +0.09 | 2.31 | 0.042 | Yes |

**Stage 1: Category Selection**

```
Select the most appropriate skill category for
this task.

Task: What is the sum of 23, 45, and 67?

Available Categories:
- Summation: Adding numbers together
- Averaging: Computing mean values
- Email Writing: Composing emails
- Sentiment Analysis: Analyzing emotional tone
- Name Extraction: Extracting names from text

Respond with ONLY the category name.
```

**Stage 2: Skill Selection within Category**

```
Select the most appropriate skill for this task.

Task: What is the sum of 23, 45, and 67?

Available Skills in "Summation" category:
- skill_001: Calculate Sum: Add all numbers
together and return the total.
- skill_002: Compute Total: Compute the total by
adding all values.
- skill_003: Sum Numbers: Sum up all the given
numbers.

Respond with ONLY the skill ID.
```

**Figure 7.** Two-stage hierarchical selection prompts used in H4.

**Table 10.** H2 Statistical Analysis: Comparison of No Competitors vs. High Competitors.

| Model | $n_{\text{base}}$ | Δ (No−High) | t-statistic | p-value | Cohen's d |
|---|---|---|---|---|---|
| GPT-4o-mini | 5 | +0.45 | 8.21 | <0.001 | 4.74 |
| | 10 | +0.63 | 12.35 | <0.001 | 7.13 |
| | 15 | +0.45 | 9.42 | <0.001 | 5.44 |
| | 20 | +0.45 | 10.18 | <0.001 | 5.88 |
| GPT-4o | 5 | +0.17 | 4.52 | <0.01 | 2.61 |
| | 10 | +0.30 | 6.87 | <0.001 | 3.97 |
| | 15 | +0.33 | 7.24 | <0.001 | 4.18 |
| | 20 | +0.37 | 8.91 | <0.001 | 5.14 |

$p = 0.67$ for GPT-4o-mini; $F(2, 6) = 0.31$, $p = 0.74$ for GPT-4o.

### E.4 H4: Hierarchy Mitigation - Complete Results

Table 12 presents the complete results for H4 experiments.

***Statistical Tests.*** Table 13 reports t-tests comparing Hierarchy vs. Flat selection at large library sizes.