The Influence of Scaffolds on Coordination Scaling Laws in LLM Agents

Mariana Meireles[*]
UC Berkeley
marianameireles@protonmail.com

Rupali Bhati[*]
Northeastern University
bhati.r@northeastern.edu

Niklas Lauffer[†]
UC Berkeley
nlauffer@berkeley.edu

Cameron Allen[†]
UC Berkeley
camallen@berkeley.edu

Abstract

As large language models improve in capability, they are increasingly taking on agentic and interactive roles in multi-agent settings that demand effective communication and coordination. In order to measure a model's capabilities in these settings, new benchmarks are quickly emerging to study language-based, multi-agent interaction, often by adding language scaffolds on top of existing multi-agent environments. However, when evaluating agents on such benchmarks, an agent's performance can be significantly influenced by implicit factors related to the design of the scaffolds, rather than the inherent properties of the model. Moreover, it is unclear if coordination among agents in these settings follows scaling laws. We consider one such environment—the popular collaborative cooking environment, Collab-Overcooked—and characterize how scaffolding plays a role in successful collaborations between models of varying sizes. We perform empirical evaluations on the collaborative capabilities of agents and find that, as long as models are given clear instructions on *how* to collaborate, their capabilities follow positive scaling laws in both self-play and cross-play. However, without a scaffold that explicitly defines how the collaboration should be done, we find models struggle to develop effective methods for collaboration, and scaling laws break down. Our experiments highlight how subtle changes in agent scaffolds can drastically impact their collaborative capabilities and raises questions on how to design evaluations for agents that may have to collaborate with open-ended partners.

1 Introduction

Large language models (LLMs) are moving from single-agent deployments to open, multi-agent ecologies in which AI assistants interact with humans and other AI agents to carry out real-world tasks. Improvements in AI agent capabilities are allowing them to be deployed for increasingly complex tasks Kwa et al. [2025]. Such tasks range from scheduling a table at a restaurant or grocery shopping Rogers [2025], to genuinely consequential ones, such as handling bureaucratic obligations, advising about medical care and other high-stakes decisions Palantir Technologies [2025], Preiksaitis et al. [2024], Li et al. [2024], Newman et al. [2022]. As assistants assume such open-ended roles, they will necessarily need to interact with other agents. Thus, we focus on the ability of models to *coordinate*: how well they form shared plans and distribute roles between varying partners. Moreover,

[*]Equal Contribution
[†]Equal Advising

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: First Workshop on Multi-Turn Interactions in Large Language Models.

we examine when coordination follows *scaling laws*, and how design choices beyond the underlying model (scaffolds, prompts, and environmental factors) shape their ability to coordinate.

We study coordination in Collab-Overcooked [Sun et al., 2025], a two-player, fully cooperative benchmark featuring explicit inter-agent communication, interactive planning, and sparse rewards. Collab-Overcooked instantiates the core primitives required for successful coordination between LLM agents—language-mediated negotiation, interdependent sub-tasks, shared-resource management, and rapid role allocation—within a controlled setting that enables precise, repeatable measurement.

Since LLM agents operating in the real-world need to be able to coordinate with open-ended partners (whether that be humans or other agents), our analysis focuses on studying models in *cross-play*: when their partner is different than themselves. Our primary analysis examines cross-play within a single model lineage (Qwen3.0; 1.7B–32B) [Yang et al., 2025]. Our first result shows that, under carefully designed scaffolds, models obey clean scaling laws when it comes to coordination: agents tend to do better when their underlying model size increases, or their partner's.

However, we find that as interactions become less clearly scripted within the agents' scaffolds, scaling laws break down. To isolate drivers of coordination, we vary three additional factors: (i) the *scaffolds* that define an agent's role in the interaction, (ii) *game structure*, varying between asymmetric and symmetric layouts ; (iii) *turn order*, swapping which assistant moves first. Our results indicate that LLM agent coordination capabilities become more limited as multi-agent interactions become less well-defined and more open-ended. Our main empirical contributions are as follows:

- **Scaling trends in cross-play.** With clearly defined scaffolds, performance generally increases with the model size of either partner with a compounding effect.
- **Open-ended interaction degrades scaling laws.** When the restrictions on how agents can interact are relaxed, scaling laws break down.
- **Emergence of hierarchy predicts success.** Task success is correlated with how strong of a leader-follower hierarchy emerges in the collaboration between agents in the game.

2 Related Work

LLM agent coordination. We use *Collab-Overcooked* [Sun et al., 2025] as the basis for our experiments, a benchmark originally used to study self-play cooperation. We build on this work by studying model's in cross-play as well as introducing modifications to the environment and scaffolds to study their effects on the coordination capabilities of agents. There are an increasing number of other benchmarks arising for studying the coordination capabilities of agents, ranging from various types of simple matrix games including "Guess 2/3 of the Average", "El Farol Bar", "Divide the Dollar" tse Huang et al. [2025] to a cooperative symmetric game of traveling salesman [Jeknic et al., 2025], to graph-based coordination environments [de Carvalho Silva and Macharet, 2025], and simulating societies of agents [Piatti et al., 2024]. However, while these settings do require communication and coordination between LLMs to succeed, they do not evaluate them in multi-step agentic coordination settings.

There is some, but limited work, on investigating LLMs in our setting of multi-agent cross-play scenarios. In [Curvo et al., 2025], the authors evaluate models of different sizes and providers in a mixed-motive game, finding that smaller, lower-performing agents can be influenced by larger ones. Another work Chen et al. [2025] studies the ability of models that are specialized for different tasks to coordinate on time series anomaly detection tasks.

Scaling laws. Several works have studied various scaling laws in language models Kaplan et al. [2020] and agentic capabilities Kwa et al. [2025]. Other prior work has also documented *inverse scaling* in language models: some behaviors worsen with scale and, with improved prompting or data, can become U-shaped [McKenzie et al., 2023, Wei et al., 2022]. We connect scaling laws to multi-agent coordination: demonstrating when and how language models follow scaling laws in cross-play.

Prompting and Scaffolds. Several works have focused on building scaffolds that coordinate multiple LLMs in a way to outperform a single model [Wu et al., 2023, Suzgun and Kalai, 2024]. For example, Wang et al. [2024] introduces a scaffold that consists of two LLM agents with distinct

roles: a planning agent and a reasoning agent. Cross et al. [2024] scaffold agents to explicitly engage in theory of mind to improve cooperation capabilities in mixed-motive settings. Similarly, our work studies the impact on agent's coordination abilities of various design choices in the scaffolds within the Collab-Overcooked environment.

3 The Environment

Benchmark and tasks. We evaluate agents in *Collab-Overcooked* [Sun et al., 2025], a two–agent kitchen simulation that enforces collaboration via resource isolation and asymmetric task knowledge. Figure 1 shows the asymmetric layout: the *Chef* controls the pot, oven, and delivery pass; the *Assistant* has access to the ingredient dispenser, chopping board, dish stack, and blender; both agents share a central counter for hand-offs.

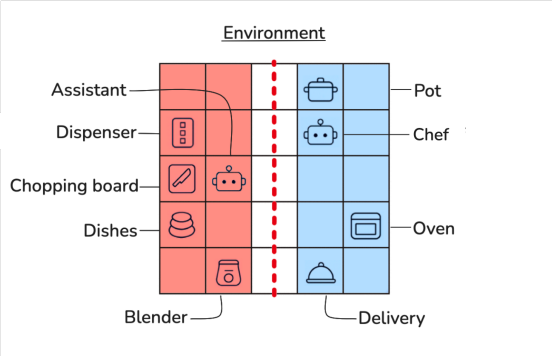


Figure 1: Collab-Overcooked layout. The column of three central tiles in which the agents are standing are walkable floor; red/blue tiles denote counters. The central line demarcates the asymmetric partition for the asymmetric environment. In the symmetric variant, we remove this divider and convert the previous shared counter area to walkable floor; all other items remain in place. The Chef becomes the First Player in the symmetric environment while the Assistant becomes the Second Player.

We study two variants of this environment:

- **Asymmetric:** Roles and access are fixed as above. This setting resolves all hierarchical ambiguity: one agent naturally plans and delivers (Chef) while the other supplies and prepares ingredients (Assistant).
- **Symmetric:** The physical partition is removed; both agents see identical task information (including the recipe) and can access all stations. No roles are prescribed—agents must negotiate a plan and divide work via communication. Prompt details are provided in the Appendix's Figure 15. As a convention we say that the agent that used to be the Chef is now First Agent and the agent that used to be the Assistant the second agent.

Agents and pairings. We study self-play (both agents are controlled by the same underlying model), and cross-play (heterogeneous models) in both the asymmetric and symmetric environments. In the asymmetric environment, we also swap which agent acts first to test turn-order effects. We use **Qwen 3.0** models at five scales: `{1.7B, 4B, 8B, 14B, 32B}`. LLMs use temperature 0.7, and a max communication budget of 4 SAY turns per communication window. We disable external tool usage.

Prompts and memory. In the *asymmetrical* environment, both players receive high-level context regarding the structure of the task, who they are, the current environment state s_t, and role-specific capabilities. Moreover, the chef has exclusive access to the recipes. In the *symmetrical* environment, agents receive the same information as above, but now they share the same prompt and context, which establishes no role but still informs them about the task to be completed. In both asymmetrical and symmetrical cases, agents retain a short-horizon scratch memory (last K transitions, K=10) and the transcript of the most recent communication window.

3.1 Experimental protocol

We evaluate *ordered* model pairs to capture role/turn-order effects. All metrics in §3.2 are computed per episode, averaged over the E episodes within a recipe, then over the S recipes within a level, and finally (when reported "overall") averaged across levels. Each episode is capped at $T_{\max} = 120$ environment timesteps.

- **Asymmetric.** For each ordered pair of models (i, j) we run S=5 recipes, each with E=10 episodes, across 5 levels (levels 1–5). This yields $5 \times 5 \times 10 = 250$ episodes per ordered pair.
- **Symmetric.** For comparability and tractability, we restrict evaluation to the two levels that admit faithful symmetric/asymmetric mappings and span different coordination regimes (levels 2 and 4; details in the Appendix D). For each ordered pair (i, j) we run the same S=5 recipes with E=10 episodes per recipe, giving $2 \times 5 \times 10 = 100$ episodes per ordered pair.

Reproducibility. All code, prompts, and scripts to regenerate figures are available at https://github.com/marimeireles/Collab-Overcooked/

3.2 Metrics

We evaluate agent coordination using two complementary metrics, success rate and similarity to the Referential Action Trajectory (RAT). The first captures whether agents successfully complete recipes, while the second measures how closely they follow optimal behavior—allowing us to assess execution quality and partial progress even in failed episodes.

1. Success Rate:

$$\texttt{success rate} = \frac{\texttt{number of times the experiment is successful}}{\texttt{total number of experiments}}$$

2. Similarity to the RAT: We measure how closely each agent follows optimal behavior using the Referential Action Trajectory (RAT). For each recipe, we precompute the complete set of optimal trajectories—all possible action sequences that complete the recipe in the minimum number of steps. Each trajectory is role-specific: in the asymmetric environment, we generate separate RAT sets for the Chef and Assistant based on their respective action spaces and responsibilities; in the symmetric environment, we generate new RAT sets that reflect the unified action space available to both players. Figure 14 in the Appendix shows an example RAT.

To score an agent's trajectory, we find the RAT in the appropriate set that has the maximum overlap with the agent's actual actions. Let n_k be the length of the RAT for agent k, and let D_j^{max} be the length of the longest subsequence match between agent j's trajectory and the closest RAT. This allows us to measure partial task completion even in episodes where agents fail to deliver the recipe. We define similarity to the RAT as:

$$\texttt{similarity to the RAT} = \frac{D_j^{max}}{n_k}$$

We compute both metrics per episode and aggregate them across recipes and levels as described in §3.1. When analyzing cross-play, we additionally report RAT similarity separately by role to identify asymmetries between Chef and Assistant performance.

4 Results

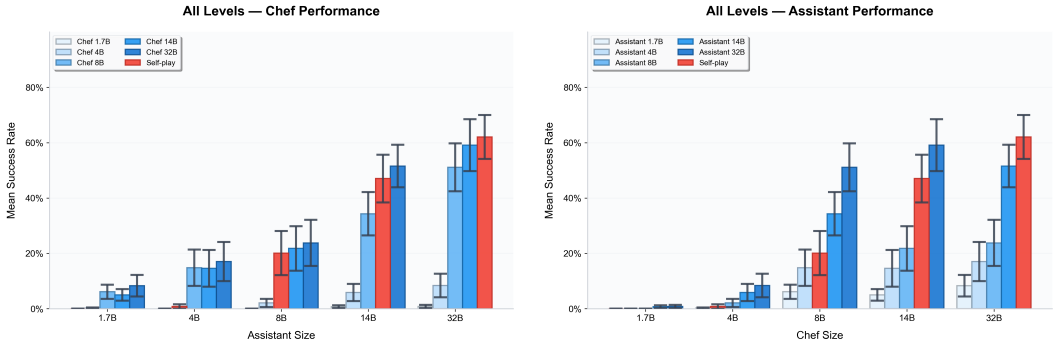
We present our empirical findings on how scaffolds influence coordination scaling in LLM agents. Through our experiments, we investigate the following hypotheses:

- H_1 (*Self-play scaling*). Within a fixed model family, self-play performance increases monotonically with model size.

- $H_{2'}$ (*Scaffold dependence of scaling*). Under less prescriptive scaffolds (no role definitions), scaling trends become less evident, indicating that scaling can be scaffold-dependent.
- $H_{3'}$ (*Hierarchy predicts success*). The emergence of a clear leader–follower hierarchy predicts higher task performance.
- $H_{4'}$ *(Parallelization amplifies coordination)*. For tasks with greater opportunity for work in parallel (higher decomposability), cooperative performance increases, and scaling trends become more evident.

4.1 Asymmetric Env

Does cooperation among LLM agents follow scaling laws? Figure 2 shows a clear scaling trend: as model size increases, mean success rises in both self-play and cross-play. The improvements are most pronounced up to mid-scale (notably from 4B to 8B and again to 14B) and then taper off, with smaller, overlapping gains beyond 14B. This pattern suggests that once a basic planning/communication competence is achieved, raw capacity ceases to be the primary limiter to increase the success rate; interaction dynamics become the bottleneck. We explain why this happens using the symmetric setting in §4.2, where turn order and emergent role assignment explain both the departures from a purely size-ordered ranking.



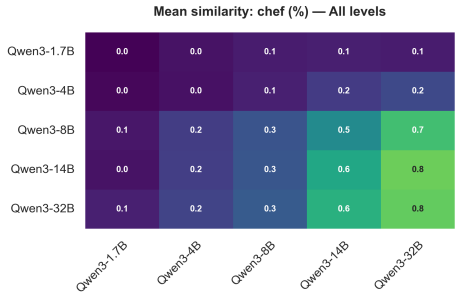
(a) With Assistant size fixed, larger Chefs raise success, but gains beyond 14B lie within overlapping confidence intervals. Chef scaling is less effective than Assistant scalling's in Fig. 2b.

(b) With Chef size fixed, larger Assistants improve success. A 32B Assistant especially boosts smaller Chefs—note the strong performance when paired with 8B and 14B models.

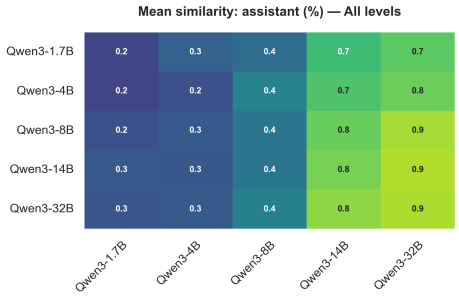
Figure 2: Cross-play and self-play outcomes in the asymmetric Overcooked setting (means with 95% CIs, averaged over all tasks)

How do LLMs behave in different roles? Figure 3 decomposes the similarity to RAT by role. Larger models not only plan well as Chef; they also track partner plans and recover from partner errors more effectively as Assistant, as we will see in Figures 4a and 4c, yielding high RAT alignment even with weaker Chefs. We have included randomly sampled examples of interactions between small models playing as Chef and larger models playing as Assistants in order to illustrate this in the Appendix F. This asymmetry explains part of the reason why some mixed pairs outperform symmetric strong–strong pairs: a capable follower resolves ambiguity early, increasing the chance for success.

Cross-vendor testing Our scaling trends are not specific to the Qwen model family. We tested three additional models in the asymmetric environment: Nemotron 14B, Gemma 4B, and Gemma 12B. These models exhibit the same coordination scaling patterns as their Qwen counterparts. Specifically, Nemotron 14B and Gemma 12B achieve success rates comparable to Qwen 14B, while Gemma 4B matches Qwen 4B performance. Figure 8 in the Appendix shows detailed success rates for these cross-vendor pairings.



(a) Mean similarity to the RAT for Chefs.

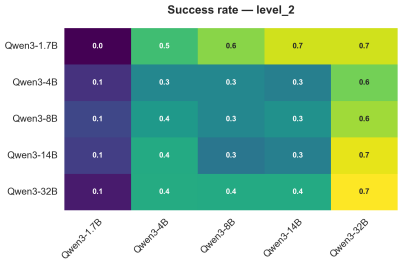


(b) Mean similarity to the RAT for Assistants.

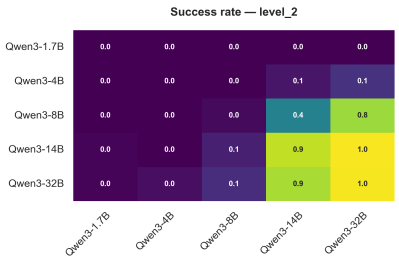
Figure 3: Mean similarity to the RAT (Referential Action Trajectory), averaged across all levels. The y-axis denotes models in the Chef role; the x-axis denotes models in the Assistant role.

4.2 Symmetric environment

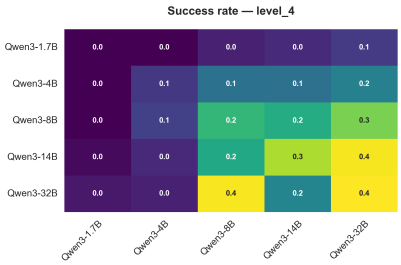
Agents perform substantially better in the symmetric environment than in the asymmetric one. Figure 4 compares success rates for levels 2 and 4 across both environments. The symmetric setting consistently yields higher completion rates across all model pairings.



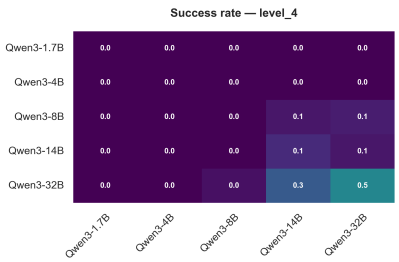
(a) Symmetric environment, level 2.



(b) Asymmetric environment, level 2.



(c) Symmetric environment, level 4.



(d) Asymmetric environment, level 4.

Figure 4: We compare the success of the models playing a symmetric version of the Overcooked game with models playing an asymmetric version. The y-axis represents the First Player while the x-axis represents the Second Player.

The symmetric environment removes one major coordination challenge: agents no longer need to track which stations their partner can access, since both agents can reach all stations. However, other coordination difficulties remain. Agents must still maintain a shared understanding of the recipe's current state (which ingredients have been prepared, what cooking steps are in progress), monitor what their partner is doing to avoid duplicate or conflicting actions, and negotiate who does what without explicit role assignments. We find that how agents resolve these remaining challenges—particularly role negotiation—strongly influences coordination success, and several factors shape these outcomes.

Turn order induces a leadership prior. The agent that acts first in the symmetric setting implicitly becomes the coordinator. This leadership role emerges not from the prompt, but simply because the

first agent reads the recipe and speaks before its partner. This turn-order effect produces three striking patterns in Figure 4a:

1. **First column is weakest (1.7B as the Second Player).** When a model is the *First Player*, it assumes the leading position and expects a cooperative follower. Instead, 1.7B tends to flood the channel with repeated recitations of the actions one should take in the environment and self-assigns actions, derailing the stronger partner's plan and stalling role resolution. This produces uniformly low success when 1.7B is cast as the follower.
2. **First row is unexpectedly strong (1.7B as the First Player).** When 1.7B goes first, its repetitive instruction dumps act like a crude "project brief." Stronger partners interpret this as a leadership signal and, with the follower prior, often salvage the plan by extracting actionable steps. The same verbosity that harms it as follower helps it as a proto-leader because the other agent adopts the follower role early.
3. **Last column is strongest (32B as the Second Player).** Having the largest model to go last, or falling in the follower role, yields robust success rates across leaders: the 32B agent rapidly understand the context and commits to the other agent's plan (Figure 12). Furthermore, in every mirrored pairing with a smaller partner $m \in \{4, 8, 14\}$, the configuration with 32B as *follower* ($m \times 32$) outranks its role-swapped mirror ($32 \times m$) in the level-2 symmetric game (see the top-7 success rate in Table 1b).

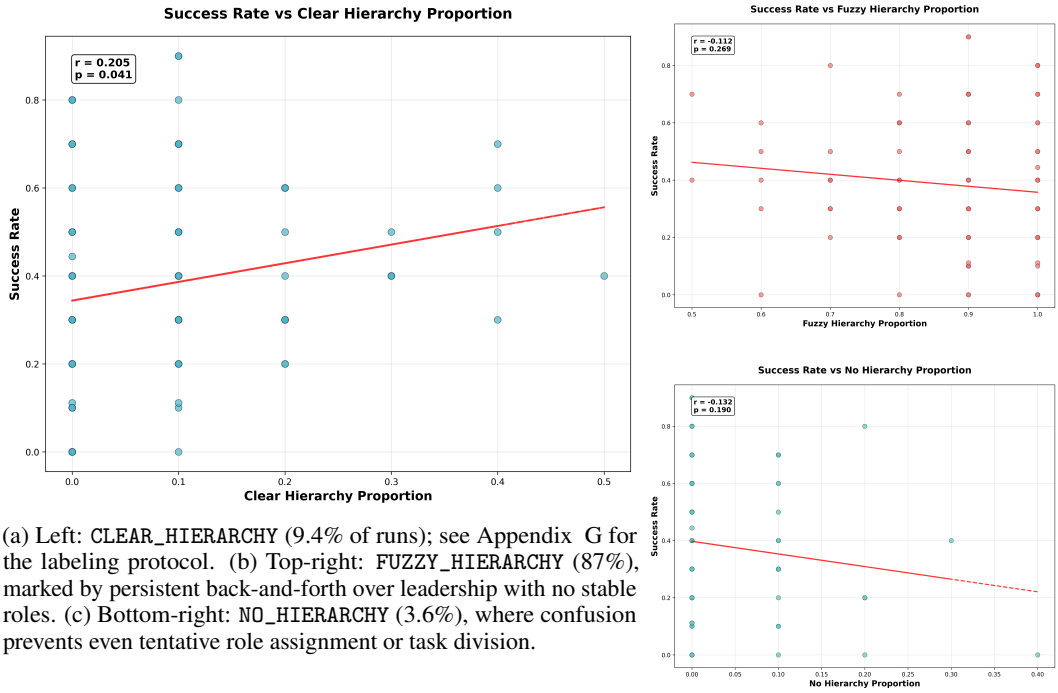
The 1.7B model's success as First Player stems from its partner compensating for its incompetence, not from genuine coordination ability. We therefore exclude 1.7B from further analysis. This model lacks basic understanding of the environment and fails to follow task constraints—it repeatedly recites action lists instead of executing them (see Appendix E for examples). While this behavior occasionally yields high success rates in level-2 symmetric tasks, this occurs only because the task is simple enough for a capable Second Player to complete alone while ignoring 1.7B's outputs. The strategy is not robust: when we increase task difficulty to level 4, the pattern disappears entirely (Figure 4c), confirming that 1.7B's apparent coordination is an artifact of partner capability and task simplicity rather than genuine competence.

Hierarchy formation correlates with performance. To measure whether agents established clear leader-follower roles, we used GPT-5 to classify each episode's interaction transcripts into three categories: `CLEAR_HIERARCHY` (one agent consistently delegates while the other executes), `FUZZY_HIERARCHY` (agents negotiate roles but never stabilize), and `NO_HIERARCHY` (no role differentiation emerges). The classifier analyzed 36 interaction snippets per episode—6 from the beginning, middle, and end for each agent—to capture role dynamics for 1000 unique games. The protocol is detailed in Appendix G.

Using this metric, we find that early hierarchy formation strongly predicts task success. In level-2 symmetric runs, pairings that quickly converged to a clear hierarchy (e.g., 14×32 and 32×32) achieved success rates of 0.6–0.7, whereas pairings with weak or unstable hierarchies (e.g., 4×14 and 8×14) hovered around 0.3. Five of the seven highest-performing combinations (Table 1b) also appear among the seven pairings with the most frequently observed clear hierarchies (Table 1a). Model capacity still matters—pairings with a large follower (e.g., 4×32 and 8×32) achieved success rates near 0.6 even without perfect hierarchy scores—but hierarchy formation is independently predictive of coordination success.

Parallelizable tasks. Level-4 tasks contain independent subtasks that can be completed simultaneously (e.g., one agent prepares ingredient A while the other prepares ingredient B; see Figure 10). When agents successfully divide this work, coordination substantially improves: agents achieve much higher success rates in level-4 symmetric tasks compared to level-4 asymmetric tasks (Figures 4c, 4d), and work division is more balanced—agents execute similar numbers of actions rather than one dominating (Figure 6b).

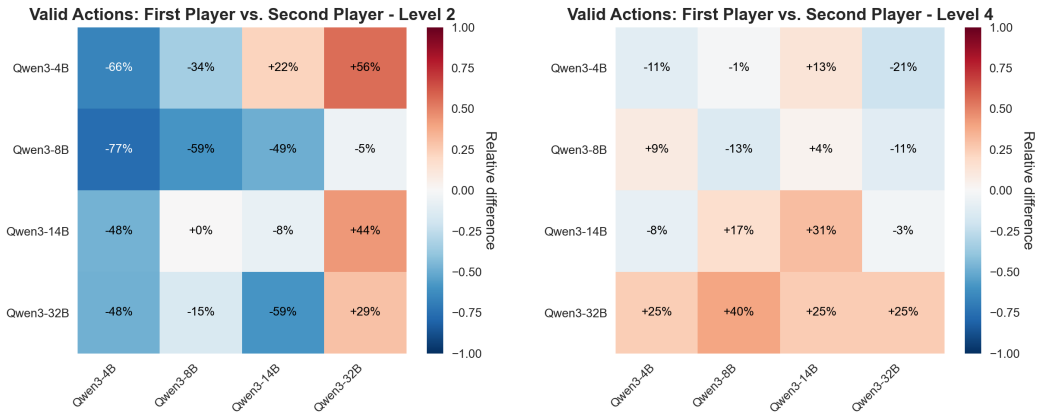
However, parallelizable structure only helps when agents successfully negotiate who does what. When coordination succeeds, two mechanisms amplify performance. First, agents complete recipes faster in the symmetric version—compare completion times between symmetric and asymmetric level-4 in Table 2. Faster completion produces shorter interaction histories, which reduces the context burden on the models. Second, once agents agree on subtask assignments, each can execute its portion independently without constantly monitoring its partner's state. This lowers coordination



(a) Left: `CLEAR_HIERARCHY` (9.4% of runs); see Appendix G for the labeling protocol. (b) Top-right: `FUZZY_HIERARCHY` (87%), marked by persistent back-and-forth over leadership with no stable roles. (c) Bottom-right: `NO_HIERARCHY` (3.6%), where confusion prevents even tentative role assignment or task division.

Figure 5: Correlation between models' success rates and hierarchy conditions.

overhead compared to tightly coupled, sequential tasks. When agents fail to establish clear roles, however, even parallelizable tasks produce poor outcomes (Figure 4). Success at level-4 therefore requires both the right task structure (parallelizable subtasks) and sufficient coordination capability (role negotiation and task division).



(a) Level 2. Number of valid actions by First Player (y-axis) vs. Second Player (x-axis).

(b) Level 4. Number of valid actions by First Player (y-axis) vs. Second Player (x-axis).

Figure 6: Task division heat maps in the symmetric environment across two difficulty levels. Values near 0 indicate an even split between players.

Task division across task complexity. Comparing level-2 and level-4, we find that explicit task division matters little in level-2, where many pairs achieve strong performance without clear role separation (Figure 6a). By contrast, at level-4, effective task division tends to emerge from successful coordination and is more tightly linked to higher success (Figure 6b). We hypothesize that level-2 problems are simple enough to solve without structured coordination, whereas the structure of level-4

inherently incentivizes parallelization; when models perceive opportunities to share work, they do so. These findings imply that coordination algorithms are complexity-dependent: strongly cooperative strategies arise only when task demands warrant them. Though as seen in Figure 4, such strategies won't emerge when model capacity is insufficient.

5 Conclusion and Future Work

Summary. We set out to understand when coordination among LLM agents obeys scaling trends and how much of the apparent scaling is induced by scaffolds rather than intrinsic model ability. Using *Collab-Overcooked* across asymmetric and symmetric variants, ordered cross-play pairings within Qwen 3.0 (1.7B–32B), and standardized prompts/memory, we find:

- $\mathbf{H}_1$ **(Self-play scaling).** With clear, prescriptive scaffolds, self-play improves monotonically with model size and exhibits a competence threshold between small and mid-scale models. Cross-play shows the same positive trend when at least one partner is sufficiently capable (Fig. 2).
- $\mathbf{H}_2$ **(Scaffold dependence).** As we remove role definitions, the neat scaling regularities break (Fig. 4). *Scaffolding* can overstate intrinsic coordination.
- $\mathbf{H}_3$ **(Hierarchy predicts success).** Stable leader–follower structure correlates with higher success (Fig. 5); turn order creates a leadership prior, and "bigger-as-follower" (e.g., 14×32, 8×32) often outperforms the mirrored pairing.
- $\mathbf{H}_4$ **(Parallelization amplifies coordination).** On tasks with decomposable subgoals, the ability to divide tasks among agents boosts division of labor, shortens trajectories, and strengthens scaling signals (Figs. 4c, 6b).

Across all settings, larger models are more valuable when serving in the Assistant role (or when they naturally assume that role in symmetric settings). These larger Assistants track global state more accurately, infer missing preconditions, and reduce coordination overhead—even when paired with weaker Chefs (Fig. 3). Cross-vendor tests with Gemma and Nemotron models show that these effects are not unique to the Qwen lineage (§4.1).

Implications for evaluation design. Our results indicate that benchmark conclusions about "coordination scaling" can be artifacts of wrappers. To make agent evaluations informative and reproducible, we recommend:

1. **Report scaffold details** (role scripts, turn order, communication budgets) and *vary* them via a *scaffold sensitivity analysis* to test whether results are robust or scaffold-induced.
2. **Include parallelizable regimes** to probe genuine cooperation rather than serial plan-following.
3. **Disaggregate by role and order** (Chef vs. Assistant; who goes first), since these systematically mediate scaling.

Taken together, H_1, $\mathrm{H}_{3'}$, and $\mathrm{H}_{4'}$ describe *when* scaling is likely to appear: clear roles, fast hierarchy formation, and opportunities for parallel work. $\mathrm{H}_{2'}$ delineates *when* scaling ceases to exist: under open-ended interaction without strong priors about who plans, who executes, and how to negotiate. The practical upshot is that *coordination scaling is conditional*—it requires scaffolds that reduce ambiguity or partners capable enough to establish structure on the fly.

6 Acknowledgment

Mariana Meireles and Rupali Bhati completed this work during an internship at the Center for Human-Compatible Artificial Intelligence. Niklas Lauffer is supported by a National Science Foundation Graduate Research Fellowship and a Cooperative AI Foundation Fellowship. Cameron Allen was supported by a gift from Open Philanthropy to the Center for Human-Compatible AI at UC Berkeley, and an AI2050 Senior Fellowship for Stuart Russell from the Schmidt Fund for Strategic Innovation.

References

F. Chen, L. Zhang, G. Pang, R. Zimmermann, and S. Deng. Synergizing large language models and task-specific models for time series anomaly detection. *arXiv preprint arXiv:2501.05675*, 2025.

L. Cross, V. Xiang, A. Bhatia, D. L. Yamins, and N. Haber. Hypothetical minds: Scaffolding theory of mind for multi-agent tasks with large language models. *arXiv preprint arXiv:2407.07086*, 2024.

P. M. P. Curvo, M. Dragomir, S. Torpes, and M. Rahimi. Reproducibility study of "cooperate or collapse: Emergence of sustainable cooperation in a society of llm agents", 2025. URL `https://arxiv.org/abs/2505.09289`.

J. V. de Carvalho Silva and D. G. Macharet. Can llm agents solve collaborative tasks? a study on urgency-aware planning and coordination, 2025. URL `https://arxiv.org/abs/2508.14635`.

I. Jeknic, A. Duchnowski, and A. Koller. Collaborative problem-solving in an optimization game, 2025. URL `https://arxiv.org/abs/2505.15490`.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

T. Kwa, B. West, J. Becker, A. Deng, K. Garcia, M. Hasin, S. Jawhar, M. Kinniment, N. Rush, S. Von Arx, et al. Measuring ai ability to complete long tasks. *arXiv preprint arXiv:2503.14499*, 2025.

S. Li, V. Balachandran, S. Feng, J. Ilgen, E. Pierson, P. W. W. Koh, and Y. Tsvetkov. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. *Advances in Neural Information Processing Systems*, 37:28858–28888, 2024.

I. R. McKenzie et al. Inverse scaling: When bigger isn't better. *arXiv preprint arXiv:2306.09479*, 2023. URL `https://arxiv.org/abs/2306.09479`.

J. Newman, M. Mintrom, and D. O'Neill. Digital technologies, artificial intelligence, and bureaucratic transformation. *Futures*, 136:102886, 2022.

Palantir Technologies. Aip for defense. `https://www.palantir.com/platforms/aip/defense/`, 2025. Accessed: 2025-09-01.

A. Piatti, R. Dantuluri, S. Narayanan, I. Schlag, G. Zheng, B. Ravindran, et al. Emergence of sustainable cooperation in a society of LLM agents. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/hash/a7e2f8c1d2ef1b0d7c99f1b36f5a9b1c-Abstract-Conference.html`. NeurIPS 2024; key kept as `llmsociety2023` for back-compat.

C. Preiksaitis, N. Ashenburg, G. Bunney, A. Chu, R. Kabeer, F. Riley, R. Ribeira, and C. Rose. The role of large language models in transforming emergency medicine: scoping review. *JMIR medical informatics*, 12:e53787, 2024.

R. Rogers. Openai adds shopping to chatgpt in a challenge to google. *WIRED*, Apr. 2025. URL `https://www.wired.com/story/openai-adds-shopping-to-chatgpt/`. WIRED Gear.

H. Sun, S. Zhang, L. Ren, H. Xu, H. Fu, C. Yuan, and X. Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. *arXiv preprint arXiv:2502.20073*, 2025. URL `https://arxiv.org/abs/2502.20073`.

M. Suzgun and A. T. Kalai. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954*, 2024.

J. tse Huang, E. J. Li, M. H. Lam, T. Liang, W. Wang, Y. Yuan, W. Jiao, X. Wang, Z. Tu, and M. R. Lyu. How far are we on the decision-making of llms? evaluating llms' gaming ability in multi-agent environments, 2025. URL `https://arxiv.org/abs/2403.11807`.

D. Wang, Z. Ye, F. Fang, and L. Li. Cooperative strategic planning enhances reasoning capabilities in large language models, 2024. URL `https://arxiv.org/abs/2410.20007`.

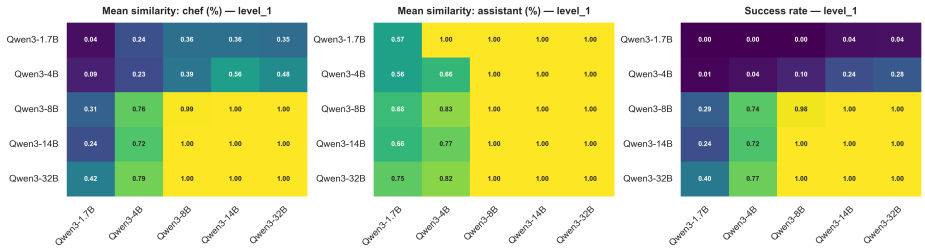
J. Wei et al. Inverse scaling can become u-shaped. *arXiv preprint arXiv:2211.02011*, 2022. URL https://arxiv.org/abs/2211.02011.

Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. doi: 10.48550/arXiv.2308.08155. URL https://arxiv.org/abs/2308.08155.

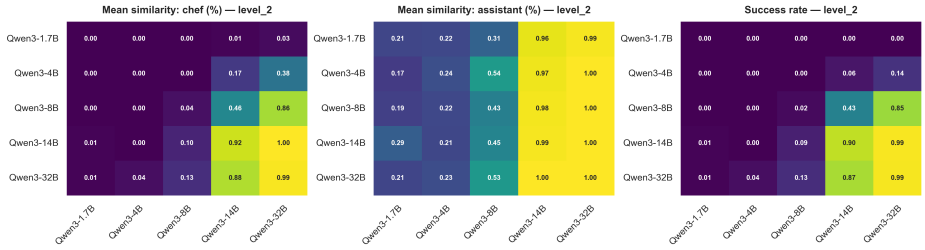
A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

A Asymmetric Environment Complete Per-level Analysis

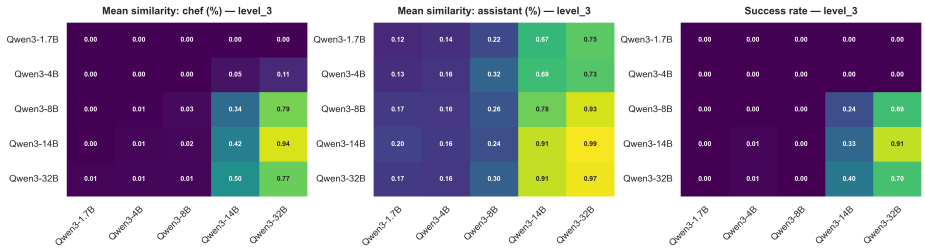
In Figure 7, we show the per-level analysis of the mean similarity to the optimal policies (RAT) and success rates per level.



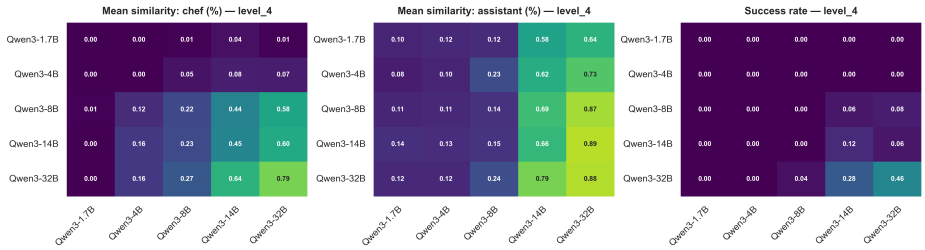
(a) Level 1 - Mean similarity of chef and assistant and success rate



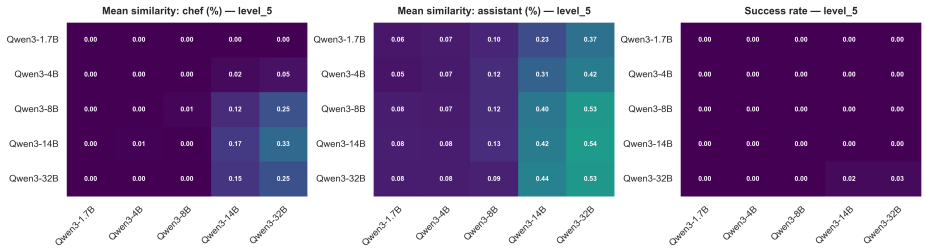
(b) Level 2 - Mean similarity of chef and assistant and success rate



(c) Level 3 - Mean similarity of chef and assistant and success rate



(d) Level 4 - Mean similarity of chef and assistant and success rate



(e) Level 5 - Mean similarity of chef and assistant and success rate

Figure 7: Mean similarity to optimal policies (RAT) and success rate in asymmetric environment

B Cross vendor pairings

To check that our coordination trends are not idiosyncratic to a single lineage, we paired similarly sized models across different families (Qwen, Gemma, Nemotron) under the *same* scaffold, prompts, decoding (temperature 0.7), and communication budget as in the main experiments. The heat map in Figure 8 summarizes success rates for these cross-vendor pairings.

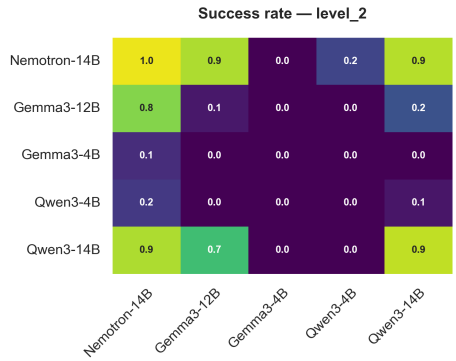


Figure 8: Success rate over similar sized models of cross vendors.

C Agent's Action Space for All Environments

These are all the actions defined by the DSL that the agents have access to and have to use in order to play the underlying implementation of Overcooked.

Action Space for Symmetric Environment

```
Action Space for all agents:
1. pickup(obj,place)
2. cut(chopping_board_name)
3. stir(blender_name)
5. put_obj_in_utensil(utensil)
6. wait(num)
7. cook(pot_name)
8. fill_dish_with_food(utensil)
9. place_obj_on_counter()
10. bake(oven_name)
11. deliver()
12. request(operation_action)
```

(a) Unified action space for all agents playing the symmetric environment.

Action Space for Asymmetric Environment

```
Action Space for Assistant:
1. pickup(obj,place)
2. cut(chopping_board_name)
3. stir(blender_name)
4. place_obj_on_counter()
5. put_obj_in_utensil(utensil)
6. wait(num

Action Space for Chef:
1. pickup(obj,place)
2. cook(pot_name)
3. place_obj_on_counter()
4. put_obj_in_utensil(utensil)
5. fill_dish_with_food(utensil)
6. bake(oven_name)
7. deliver()
8. wait(num)
```

(b) In the asymmetric environment roles come predetermined with different actions an agent can take.

Figure 9: Action space

D Rationale for Selecting Levels 2 and 4 in the Symmetric Environment

We restrict symmetric–environment evaluation to Levels 2 and 4 for empirical coverage of two distinct coordination regimes. In the asymmetric setting, Level 2 shows the greatest success-rate dispersion among smaller models (cf. Fig. 7), making it sensitive to coordination differences not swamped by raw capacity. Symmetrizing Level 2 removes access constraints while preserving difficulty, letting us probe early role negotiation of a wider range of models.

The recipes of Level 4 incentivize the development of a concurrent algorithm in order to solve them as sub-tasks (Figure 10), making it an interesting study case for *work division* and *parallel execution*. The symmetric variant lets us assess whether agents that *can* parallelize actually do, and how this interacts with emergent hierarchy.

Example of Level 4 Recipe

```
NAME:
Sliced Bell Pepper and Corn Stew

INGREDIENTS:
bell_pepper (1)
corn(1)

COOKING STEPs:
1. Cut a bell pepper into slices.
2. Place the bell pepper slices and corn in a pot.
3. Cook the ingredients in the pot for 3 timesteps.
4. Fill a dish with the stew from the pot and serve.
```

Figure 10: Example of one of the recipes used in level 4. The recipes are shared among environments.

E 1.7B Model Removal

Assistant

Analysis

"I need to complete the recipe for Boiled Corn Slices. The recipe requires cutting corn into slices, placing them in a pot, and then cooking for 3 timesteps. Since the recipe is simple, I will focus on the steps required. I need to pick up the corn, cut it, place it in the pot, and then cook it. I will also need to serve it once it's done."

Say

"request(pickup(corn, ingredient_dispenser)); request(cut(chopping_board0)); request(put_obj_in_utensil(pot0)); request(cook(pot0)); request(deliver_soup());"

Plan

"Player0 plan: \npickup(corn, ingredient_dispenser); cut(chopping_board0); put_obj_in_utensil(pot0); cook(pot0); deliver_soup();"

Figure 11: In the majority of Qwen 1.7B runs, the agent degenerates into repeatedly echoing the DSL or other role-specific instructions—often for many turns—instead of producing task-advancing actions. This is an example of a level-2 asymmetric environment run.

F Larger Models as Assistants

Larger assistants drive most of the cross-play gains. Holding the Chef fixed, success rises with the Assistant's scale (Fig. 2b); and Assistant RAT alignment remains high even when paired with weaker Chefs (Fig. 3b). Qualitatively, large Assistants act as robust followers: they maintain an accurate global state, infer missing preconditions, and issue targeted partner requests or execute the needed primitives—rather than reciting rules.

These behaviors reduce coordination overhead for the Chef and explain why "bigger-as-follower" configurations (e.g., 14×32, 8×32) often outperform their mirrors (e.g., 32×14, 32×8) in the symmetric setting (Fig. 4a).

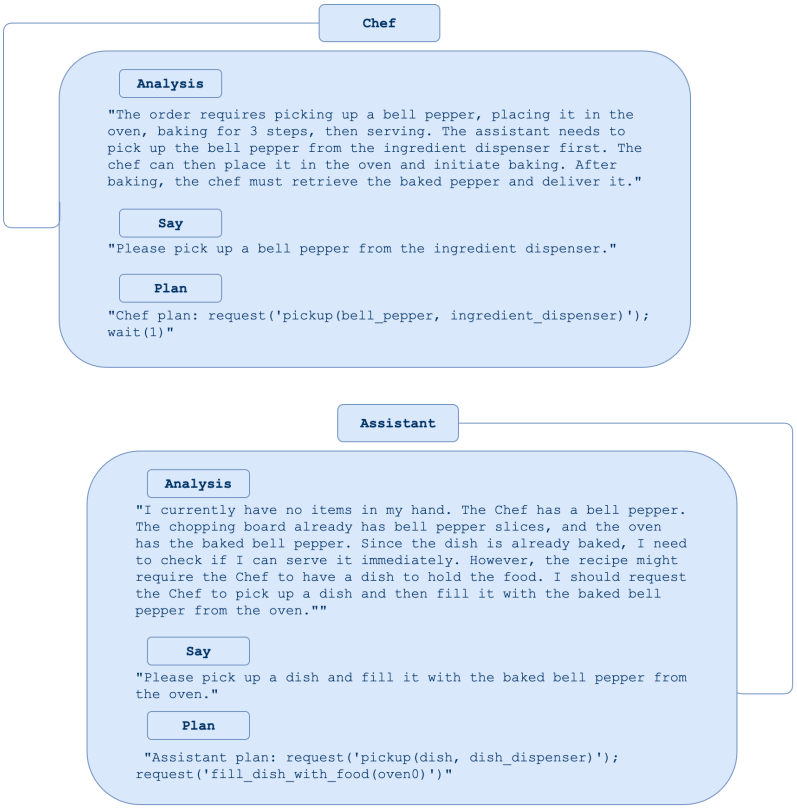


Figure 12: A 32B Assistant exhibits high-quality follower behavior: it summarizes the state (Chef holds pepper; oven contains a baked item), remembers Chef's past states (that the Chef itself seems to have forgotten), infers a latent precondition (plating requires a dish), and issues precise partner requests that close the loop—`request('pickup(dish, dish_dispenser)'); request('fill_dish_with_food(oven0)')`—instead of echoing the DSL. This example was taken from an asymmetric run.

G Top performers in the symmetric environment for level-2

This section outlines our hierarchy labeling procedure and reports the top-performing model pairings on the symmetric level-2 task, emphasizing the association between early role formation and success rates.

Hierarchy evaluation protocol. Across all runs in the symmetric setting, the distribution of hierarchy labels was `CLEAR_HIERARCHY` (9.4%), `FUZZY_HIERARCHY` (87.0%), and `NO_HIERARCHY` (3.6%). The labeling protocol is specified in Figure 13. For each episode, the evaluator received the base prompt plus *36* interaction snippets: 6 from the beginning, 6 from the middle, and 6 from the end *for each of the two agents*. Each snippet contained the complete reasoning, planning, and spoken turns at that time point. This yielded up to 36 analyzed steps per game; if an episode had fewer than 36 steps, we analyzed all available interactions.

We evaluated with `GPT-4-nano` and `GPT-4` as well, though their evaluation quality was below human-validator expectations. For `GPT-5`, a human audit of a 5% stratified sample of symmetric runs found the evaluations to be largely correct.

Table 1a lists the model pairings that most frequently exhibited a *clear* leader–follower structure in the symmetric environment. For comparison, Table 1b ranks the same set of experiments by success rate. The overlap between the two tables supports our central claim: early emergence of a stable hierarchy is positively associated with task success. Pairs that rapidly settle on complementary roles (one agent delegating, the other executing) tend to complete recipes more reliably and with fewer coordination stalls.

Hierarchy querying prompt

```
Below are agent interactions from a cooperative cooking task.
Please analyze them and provide a structured response.

{formatted_interactions}

ANALYSIS TASK:
Analyze the above interactions to determine the hierarchical
behavior between the agents.

CLASSIFICATION CATEGORIES:
1. NO_HIERARCHY: Both agents work independently or as equals
with no leadership patterns
2. FUZZY_HIERARCHY: Agents occasionally show leadership/
coordination but no clear roles
3. CLEAR_HIERARCHY: Clear hierarchical roles with consistent
leadership and delegation

2. Finally, classify the hierarchy using exactly one of:
NO_HIERARCHY, FUZZY_HIERARCHY, or CLEAR_HIERARCHY

If you classify as CLEAR_HIERARCHY, also specify the hierarchy
structure as: Agent_0 [1/0], Agent_1 [1/0] where 1=leader,
0=follower.

Your answer must contain the following words: NO_HIERARCHY,
FUZZY_HIERARCHY, or CLEAR_HIERARCHY:
```

Figure 13: Prompt used to query the OpenAI GPT-5 model in order to achieve hierarchical classification.

Table 1: Ranking of the best 7 performing models in the symmetric environment in the hierarchy and succes rate metrics.

(a) Models that most commonly formed clear hierarchies in the symmetric environment

Rank	Configuration	Success
1	14×14	0.3
2	14×32	0.7
3	14×8	0.3
4	32×14	0.4
5	32×8	0.4
6	8×32	0.6
7	32×32	0.7

(b) Highest success in the symmetric environment

Rank	Configuration
1	32×32
2	14×32
3	8×32
4	4×32
5	32×4
6	32×8
7	32×14

H Task division among players

For the task-division metric, we compute a role-specific RAT for each agent, align their trajectories, and compare the counts of RAT-consistent (valid) actions between agents to assess how evenly work was split in a setting without enforced role constraints.

RAT adaptation for the symmetric setting. In the symmetric environment we omit counter–placement and counter–pickup actions from the reference RAT. In such an unconstrained environment, handoffs via the central counter are not relevant. Besides, the task can be completed end-to-end by a single agent, so these logistics steps are not required by the optimal plan. Excluding them shortens the reference trajectory and thus reduces the number of actions an agent must match to achieve a perfect RAT alignment. We don't compare RAT across environments.

Example of Level 4 RAT

```
ANY TIME DURING GROUP 1 OR GROUP 2
"pickup(corn,ingredient_dispenser)"
"pickup(dish,dish_dispenser)"

SEQUENCE GROUP 1
"pickup(bell_pepper ingredient_dispenser)"
"put_obj_in_utensil(chopping_board0)"
"cut(chopping_board0)"
"pickup(bell_pepper_slices,chopping_board0)"

SEQUENCE GROUP 2
"put_obj_in_utensil(pot0)"
"put_obj_in_utensil(pot0)"
"cook(pot0)"

SEQUENCE GROUP 3
"fill_dish_with_food(pot0)"
"deliver_soup()"
```

Figure 14: Example of a RAT used to compare level-4 agents actions.

I Mean Timestamp for Task Completion for All Environments

Table 2: Mean timestamp by model combination and recipe level 4 running on an symmetric and asymmetric environment

Model Combination	Mean Timestamp (Symmetric)	Mean Timestamp (Asymmetric)
1.7×1.7	119.0	119.0
1.7×4	113.3	119.0
1.7×8	115.0	119.0
1.7×14	112.1	119.0
1.7×32	116.5	119.0
4×1.7	119.0	119.0
4×4	112.9	119.0
4×8	109.4	119.0
4×14	103.7	119.0
4×32	95.4	119.0
8×1.7	116.5	119.0
8×4	110.4	119.0
8×8	106.0	119.0
8×14	104.3	116.9
8×32	101.1	117.0
14×1.7	116.4	119.0
14×4	106.7	119.0
14×8	100.5	119.0
14×14	100.0	114.9
14×32	95.0	115.9
32×1.7	116.6	119.0
32×4	109.2	119.0
32×8	99.2	118.7
32×14	103.2	109.0
32×32	99.2	97.7

J Prompting

Thinking mode. Qwen 3.0 supports a “thinking" mode, but pilot runs showed episode runtimes increasing by numbers as large as an order of magnitude, making evaluation impractical. Because our focus is coordination under realistic latency constraints, we disable thinking mode and use standard decoding for all models.

Figure 15 shows the modified prompt used to establish the task context for both players in the symmetric environment.

Prompting for Agents Playing the Symmetric Environment

```
Suppose you are a player proficient in an adapted version of the
Overcooked game. Your goal is to cooperate with your teammate to complete
recipes efficiently.

You have full recipe access and can make independent decisions.
Coordinate with your teammate as an equal partner.

Due to the symmetric nature of this environment, you must strictly adhere
to the following game rules:

**Game Rules:**

- The Symmetric Overcooked_AI game requires two players with identical
capabilities to work together with the goal of completing recipes in the
shortest time.
- To finish recipes, your team needs to follow these steps:
  1. Either player can pick raw ingredients from the ingredient
dispenser.
  2. Place the ingredients in the correct utensil and initiate cooking
using the appropriate action.
  3. Either player can pick up the cooked food and decide whether to
serve it immediately.
  4. Check if you need a dish to hold the cooked food. If so, either
player can pick up a dish first and then fill it with food from the
utensil. Otherwise, either player can directly pick up the cooked food
from the utensil.
  5. Either player can deliver the food to the serving location
immediately.

*Both players have equal capabilities. Coordinate efficiently to avoid
conflicts and maximize productivity.*

- Both players have equal capabilities and recipe access
- Coordinate as peers to complete recipes efficiently
- Either player can: pickup ingredients, use any utensil, deliver food
- recipes are simple: ingredient → single utensil → finished food

- The recipe contains all the steps necessary to complete the order.
Every choice you make must be based on the recipe.
- You only need to complete one order, so focus solely on the progress of
that dish.
- Both players can pick up ingredients from the ingredient dispenser,
which has an unlimited supply.
- The utensil is a stationary unit that cannot be moved.
- After placing an ingredient into a utensil, you need to use the correct
action to start cooking.
- As long as there is something on the counter, both players can directly
pick it up.
- If you wish to place something in another position, first check if you
are holding it by verifying through "Player 0 holds XXX" or "Player 1
holds XXX." Each player can only pick up one item at a time.
- Players can only pass items by placing them on the counter. There is no
corresponding "pass" action. If you want to pass an item to your
teammate, you need to pick up the item and then place_obj_on_counter(),
and then tell your teammate to pickup the item.
```

Figure 15: Symmetric environment rules' prompt. Both agents in the symmetric environment receive the same prompt.