ZERO-SHOT COORDINATION AMONG LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study zero-shot coordination (ZSC), where independently trained agents must cooperate with unseen partners at test time. While ZSC is well studied for RL agents, little is known about how LLM-based agents perform in this setting, despite their growing real-world use. Prior work on LLM coordination relies on specialised scaffolding that is unlikely to be adopted by independent agents and tends to overfit narrow tasks (e.g., Overcooked) that are too complex to isolate fundamental coordination abilities. In contrast, we study simple, generic scaffolds in minimal environments from the zero-shot coordination literature. Our experiments show that even in simplified settings, frontier LLM agents fail to coordinate due to a limited understanding of the underlying coordination challenge and poor reasoning about their partner's behaviour. While LLMs enable the use of *semantic information* for coordination, we find that agents fail to leverage it effectively. Finally, we propose *Coordination-friendly definitions (CFDs)* as a principled way to enable robust coordination among LLM agents.

1 INTRODUCTION

Rapid advances in artificial intelligence have led to the widespread deployment of autonomous agents in environments that require robust coordination with humans and other independently developed agents (Dafoe et al., 2020; Kirkham et al., 2025; Müller & Žunič, 2024; Hu et al., 2021b). While humans often readily adapt to novel partners by reasoning about the behaviour of others, referred to as *theory of mind (ToM)* (Premack & Woodruff, 1978), autonomous agents struggle to coordinate with one another (Hu et al., 2021b; Foerster, 2024). A classic real-world example is that of autonomous driving, where vehicles developed independently by companies such as Waymo, Tesla, and Wayve must coordinate during on-road interactions (Kirkham et al., 2025).

With the rapid adoption of LLM agents in real-world applications, coordination between independently developed LLM agents is also becoming increasingly common. For example, in AI-assisted development environments such as Cursor, coding agents powered by different LLM providers are often used together to collaboratively generate code (Lin, 2026). Similar coordination settings also arise in AI-based scientific discovery pipelines (Novikov et al., 2025), retrieval augmented generation (RAG) systems (Zhang et al., 2024d), and other multi-agent LLM workflows such as debate and policy-making (Guo et al., 2024a), highlighting the growing importance of robust coordination among independently developed LLM agents.

This problem is known as the Zero-shot coordination (ZSC) setting (Hu et al., 2021b): agents are trained independently and evaluated together at test time without prior communication or re-training. Although ZSC prohibits agents from sharing policies, it permits prior agreement on a game-agnostic learning rule used during training. The central objective is therefore to identify learning rules that encode general coordination principles applicable across a wide range of settings, while avoiding common failure modes like arbitrary symmetry breaking.

ZSC literature has predominantly focused on the traditional reinforcement learning (RL) training pipelines (Hu et al., 2021b;a; Muglich et al., 2025). More recently, however, the emergence of large language models (LLMs) has given rise to a new class of autonomous agents (Müller & Žunič, 2024; Ning et al., 2025; Novikov et al., 2025) that differ substantially from traditional RL agents. LLM agents offer a new axis for coordination by leveraging the *semantic descriptions* of the environment, a capability that has been difficult for traditional agents to exploit (Ma et al., 2023). As illustrated in Figure 5, agents can use semantic labels (e.g., the red arrow) to coordinate actions and, further,

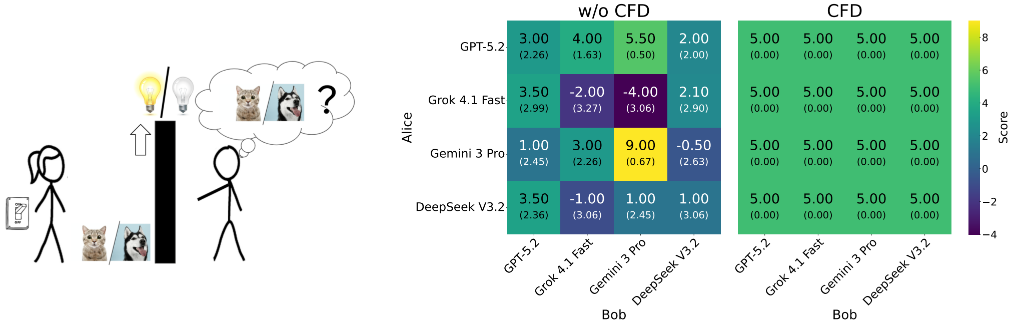


Figure 1: **Cat & Dog coordination game.** *(Left)*. Alice observes an animal (cat/dog). She can either bail and end the game for a joint reward of 1, remove the wall for a cost of 5, or turn on the light bulb for free. Bob observes the light bulb and the animal if the wall is removed. He can either guess the animal (+10/-10) or bail for a reward of 0.5. *(Right) XP of text agents in the variable LLM setting with and without CFD.* Each cell contains the average score across 10 episodes.

leverage in-context learning (ICL) to adapt rapidly to new tasks and partners. Despite this potential, their behaviour in the ZSC setting remains largely unexplored.

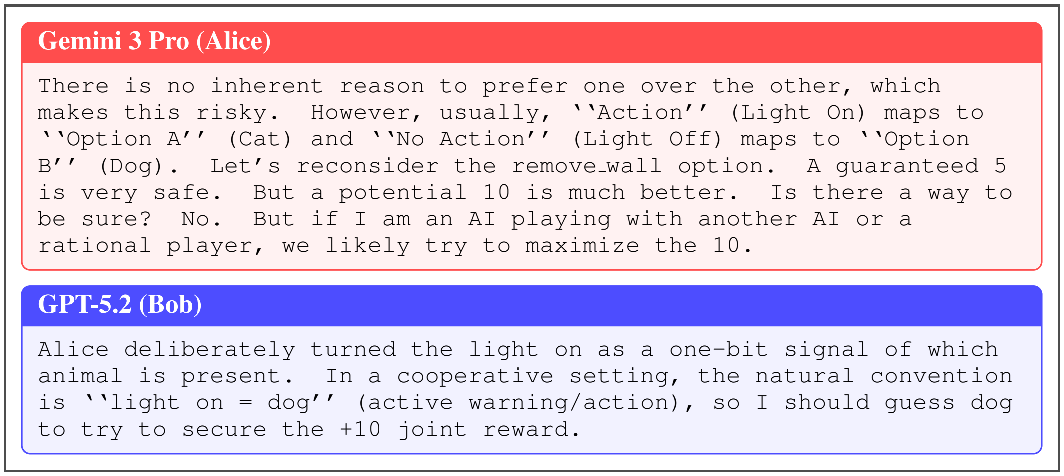
In this work, we conduct an in-depth study of LLM-based agents in the zero-shot coordination setting, beginning with a formalisation of two widely used scaffolds: text agents and code agents within the ZSC framework. We then introduce a suite of cooperative environments (described in Appendix C), inspired by prior work in ZSC (Foerster et al., 2019; Hu et al., 2021a; Lupu et al., 2021; Hu et al., 2022), each designed to isolate a specific coordination challenge. We augment these environments with text descriptions, or *semantics*, to support interaction with LLM agents. Using this suite, we show that both classes of agents struggle to coordinate reliably even in simple scenarios, driven in part by a weak understanding of the underlying coordination challenge and weak ToM abilities, even when the partner is known to be an identical copy of itself. Next, we show that *environment semantics* shape agent behaviour and they have a strong bias to use this information for coordination. We show that, depending on the environment, this may benefit or harm agents, motivating a rethinking of coordination for LLM agents relative to conventional RL agents.

Finally, drawing inspiration from the ZSC objective of a shared learning rule among coordinating entities, we introduce the notion of a *Coordination-friendly definition (CFD)* that encodes general coordination principles in natural language, and is shared among the agents. This closely parallels *Constitutional AI (CAI)*, where high-level rules and principles for safety and human oversight are articulated in natural language as a "constitution" that model providers can share, promoting adherence to a common set of overarching safety values (Bai et al., 2022; Huang et al., 2024). Finally, we show that incorporating a CFD improves coordination for both text and code agents across multiple environments, highlighting its potential to enable more robust coordination among LLM agents.

2 Background and Notation

Dec-POMDPs We focus on fully cooperative Markov games which can be represented in the Decentralised Partially Observable Markov Decision Process (Dec-POMDP) framework (Bernstein et al., 2002; Oliehoek & Amato, 2016). A Dec-POMDP $\mathcal{M}$ consists of M agents, a state space $\mathcal{S}$, action space $\mathcal{A}$, per-agent observation spaces O_i, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, discount factor $\gamma \in [0, 1]$ and horizon $T \in \mathbb{N}$, which represents the maximum length of an episode. Additionally, agents share a common reward function $r : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ that reflects the cooperative nature of the task. Each agent observes the environment via its observation function, $\mathcal{O}_i : \mathcal{S} \to \Delta(O_i)$. The action space can be written as a product $\mathcal{A} = A_1 \times \cdots \times A_M$, where A_i corresponds to agent i's action space. The goal is to maximise the joint return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$. A (joint) policy π is an M-tuple of local policies $\pi_i : H_i \to \Delta(A_i)$, where H_i is the space of possible action-observation histories (AOHs) of player i, each of the form $\tau_t^i = \{o_0^i, a_0^i, \ldots, a_{t-1}^i, o_t^i\}$.[1] Additionally, we assume access to natural language descriptions, or *semantic features* (Ma et al.,

[1] In general, the action space may depend on the AOH τ_t^i; we omit this dependence for ease of notation.



Gemini 3 Pro (Alice)

There is no inherent reason to prefer one over the other, which makes this risky. However, usually, ``Action'' (Light On) maps to ``Option A'' (Cat) and ``No Action'' (Light Off) maps to ``Option B'' (Dog). Let's reconsider the remove_wall option. A guaranteed 5 is very safe. But a potential 10 is much better. Is there a way to be sure? No. But if I am an AI playing with another AI or a rational player, we likely try to maximize the 10.

GPT-5.2 (Bob)

Alice deliberately turned the light on as a one-bit signal of which animal is present. In a cooperative setting, the natural convention is ``light on = dog'' (active warning/action), so I should guess dog to try to secure the +10 joint reward.

Figure 2: **Reasoning traces from a failed coordination in the *Cat and Dog* environment.** Gemini 3 Pro and GPT-5.2 independently arrive at incompatible strategies.

2023) $\mathcal{D} = (\mathcal{D}_{\mathcal{M}}, \mathcal{D}_{\mathcal{R}}, \{\mathcal{D}_{H_i}\}_{i \in [M]})$ of the Dec-POMDP. $\mathcal{D}_{\mathcal{M}} \in \Sigma^*$ is a description of the Dec-POMDP, with Σ^* noting all strings over a suitable finite alphabet Σ. $\mathcal{D}_{H_i} : H_i \to \Sigma^*$ and $\mathcal{D}_{\mathcal{R}} : [M] \to \Sigma^*$ are injective functions, describing each AOH τ_i and role r in natural language.

LLM Agents In this work, we define an LLM agent as a system that leverages an LLM and ICL to generate a local or joint policy π for a Dec-POMDP. Crucially, ICL does not involve a separate training phase with gradient-based parameter updates; instead, all the "learning" occurs through conditioning on the prompt during the model's forward pass. Note that in a slight abuse of terminology, we also refer to the generated policies as agents when analyzing their behaviour. We consider two classes of LLM agents as illustrated in Figure 7: **text agents** and **code agents**. Text agents are assigned to a role $r \in [M]$, take as input $(\mathcal{D}_{\mathcal{M}}, \mathcal{D}_{\mathcal{R}}(r), \mathcal{D}_{H_i}(\tau_i))$ and return an action a_t^i, whereas code agents take as input $\mathcal{D}_{\mathcal{M}}$ and return a joint policy π as executable code. Notably, text agents do not produce an explicit joint policy; instead, they act on an implicit local policy π^r assigned to their role, which is repeatedly prompted to take actions. A key difference is that while code agents produce a distribution over mostly deterministic joint policies, text agents induce a single implicit stochastic joint policy. Generally, text agents are applicable to more complex environments, but their behaviour is hard to interpret or modify. At last, having to prompt an LLM for each action means that text agents are orders of magnitude slower and more expensive to execute.

Zero-shot Coordination Traditional approaches to cooperative multi-agent reinforcement learning have largely focused on the centralized training with decentralized execution setting (Lowe et al., 2017; Foerster et al., 2018; Rashid et al., 2018; Sunehag et al., 2018), in which we assume that the local policies that need to coordinate at test time are trained by a central entity. More generally, the setting in which multiple agents are optimized through interaction with the same partners they will face at test time is known as self-play (SP) (Samuel, 1959; Tesauro, 1994). In contrast, the zero-shot coordination setting (Hu et al., 2021b) assumes that multiple entities independently train policies, which are paired at test time without prior adaptation or communication, a setting known as cross-play (XP). To enable coordination, entities are willing to coordinate on (part of) a game-agnostic learning rule for training their agents, but they independently observe the underlying Dec-POMDP. Examples of ZSC learning rules make use of known game symmetries (Hu et al., 2021b) or explicitly control for how agents reason about their partner (Hu et al., 2021a).

3 Problem Formulation

Consider a Dec-POMDP with two LLM agents i and j. Code agents independently generate joint policies π_i, π_j, with the SP score given as $\mathbb{E}_{\pi_i}[J(\pi_i)]$ and $\mathbb{E}_{\pi_j}[J(\pi_j)]$. Their XP score is defined as

$$J(i,j) = \tfrac{1}{2}\mathbb{E}_{(\pi_i,\pi_j)}\big[J(\pi_i^1, \pi_j^2) + J(\pi_j^1, \pi_i^2)\big]. \tag{1}$$

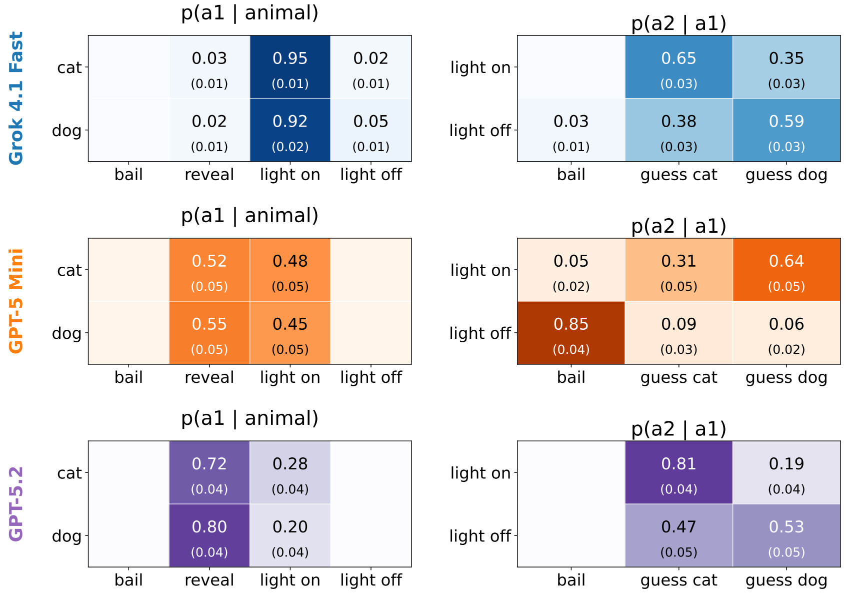


Figure 3: **Local policies induced by GPT-5.2, Grok 4.1, and GPT-5 mini on Cat & Dog when used as text agents.** All LLMs fail to adopt an interpretable signaling convention for Alice, having about an equal likelihood to signal for either animal. Moreover, GPT-5.2 and Grok 4.1 interpret a "light on" signal differently from GPT-5 mini.

For text agents, we assign each agent to both possible roles, yielding local policies that can be combined into joint policies π_i and π_j and similarly define the XP score according to Equation 1.

We evaluate two forms of XP: (1) **fixed LLM ($\mathbf{XP_{seed}}$)** and (2) **variable LLM ($\mathbf{XP_{LLM}}$)**. The *fixed* LLM setting evaluates XP among independent instances of agent scaffolds using the same underlying LLM. Given the non-determinism inherent to most frontier reasoning models (He & Lab, 2025), independent calls to the same LLM are analogous to training policies with different random seeds, as is standard in traditional ZSC evaluations (Hu et al., 2021a;b). In contrast, the variable LLM setting evaluates among independent instances of agent scaffolds using different LLMs, more closely reflecting real-world coordination scenarios faced by deployed agents. Note that the fixed LLM setting does not apply to text agents, as they operate under a *single* implicit local policy; pairing an agent with itself therefore corresponds to SP. However, unlike code agents, text agents cannot directly construct joint policies and are not optimized to coordinate with a specific partner. As a result, they often exhibit poor SP performance, as shown in Section 4.

4 ZSC PERFORMANCE OF LLMS

It is natural to ask whether LLM agents exhibit a SP-XP gap in the ZSC setting, or instead coordinate effectively out of the box, rendering further study unnecessary. To investigate this question, we conduct initial experiments across nine diverse coordination problems. Many of these are adapted from the ZSC literature, including Cat & Dog (Hu et al., 2021a), Tiny Hanabi (Foerster et al., 2019), a variant of the lever game (Hu et al., 2021a; Treutlein et al., 2023), and a matrix game (Lupu et al., 2021). All coordination environments are described in detail in Appendix C. In addition to identifying the correct safe fallback option, the underlying coordination challenges include: formation of ad hoc conventions within the episode, navigation challenges, and simultaneous action selection in a single step and iterated variants. To ensure a fair evaluation, we include an explanation of the cross-play evaluation framework used, as well as provide explicit instructions to the agents to optimise their policies for high XP scores. The prompt templates used for both agents are presented in Appendix D. To gain further insight into the failure modes of LLM coordination, we conduct a detailed analysis of both text-based and code-based agents in the canonical Cat & Dog toy environment (Hu et al., 2021a), as shown in Figure 1. Here, two incompatible conventions (light on ⇒ cat and light on ⇒ dog) both achieve the maximum reward of 10. However, without any prior communication,

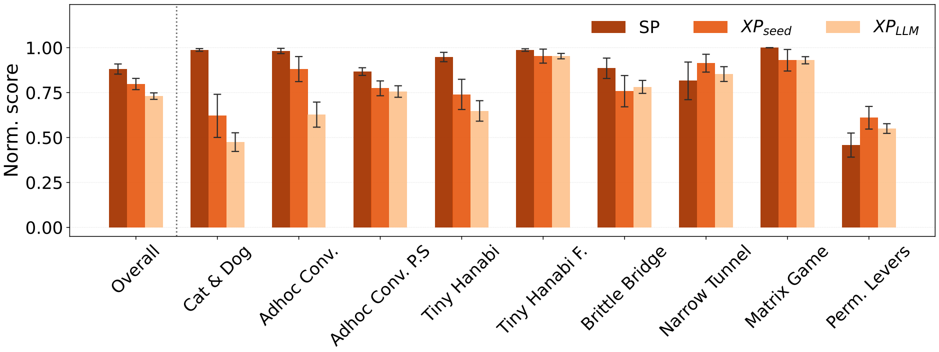


Figure 4: **SP and XP scores of code agents aggregated across different LLMs.** We observe that code agents exhibit a significant SP-XP gap across multiple environments. For further details, please refer to Appendix B.4.

the (optimal) grounded strategy is to remove the wall, yielding a reward of 5. For code-based agents, we employ SOTA reasoning models, namely Grok 4.1 Fast, GPT-5.2, Gemini 3 Pro, Claude Opus 4.5 and DeepSeek V3.2. Due to resource constraints, we evaluate the *fixed* LLM setting across all games using GPT-5.2 and Grok 4.1 Fast, and restrict the *variable* LLM setting to Cat & Dog.

Text agents As shown in Figure 1, even in a simple environment such as Cat and Dog, text agents coordinate poorly, resulting in XP scores that fall well below the safe guaranteed reward of 5 obtained by removing the wall. We find that Gemini 3 Pro consistently adopts the convention *light on* $\Rightarrow$ *cat*, enabling it to coordinate with itself. However, because this convention is arbitrary, it leads to poor coordination with other agents. An example of this failure is shown in Trace 2, which records the reasoning traces from an episode between Gemini 3 Pro and GPT 5.2. When acting as Alice, the Gemini agent *recognizes* that any coordination convention is *arbitrary*, yet nevertheless selects one in order to avoid the -5 penalty and subsequently rationalizes this choice as "natural". In contrast, when acting as Bob, the GPT agent independently judges the opposite convention to be "natural" and does not consider bailing in its reasoning.

To better understand the behaviour of text-agents in this environment, we compute the local policies of GPT 5.2, GPT 5 mini, and Grok 4.1 Fast (Figure 3). For Alice, we observe that none of the three LLMs induce self-compatible local policies, with the probability of turning the light on being mostly independent of the animal. When the models choose not to reveal, they strongly prefer *light on* over *light off*, despite both actions being canonically equivalent. As Bob, Grok 4.1, and GPT-5.2 also yield opposite interpretations of the lit bulb than GPT-5 mini. Relabeling the signaling actions as *red light* and *blue light* (Figure 10 in the Appendix) substantially alters behavior: Grok now favors *blue light* as a signal for Dog, while the GPT models choose to reveal more frequently. Moreover, when acting as Bob, GPT-5 mini often bails upon observing *light off*, but shows a strong bias toward guessing *dog* when observing *blue light*. Translating the game description and action labels into German further changes local policies (Figure 12 in the Appendix); for example, GPT-5 mini becomes biased toward guessing *cat* rather than *dog* upon observing *light on*. Together, these results indicate that the models lack a coherent understanding of the underlying game structure and are instead arbitrarily influenced by the *semantics*.

As discussed in Section 1, successful coordination requires a degree of theory of mind (ToM) (Premack & Woodruff, 1978), since agents must reason about which conventions, if any, their partners are likely to adopt. We observed that the agents are strongly averse to incurring the penalty for revealing, which pushes them to adopt a convention (also seen in Trace 2). Moreover, agents implicitly assume their chosen convention is "natural", something that the Gemini agent exploits. We test whether this is a general strategy that other agents can exploit as well. For this, we add the following instruction to the prompt: "You are playing a cooperative game with an identical instance of yourself. The same language model is used for both players, but each instance is assigned a different role in the game." If the model has a coherent notion of what it considers a "natural" convention, it should be able to coordinate on that convention and thereby achieve the maximum reward. That said, as shown in Figure 11 in the Appendix, the agents do not exhibit this form of reasoning. Instead, the agents collapse to a policy of selecting *light on* regardless of the observed

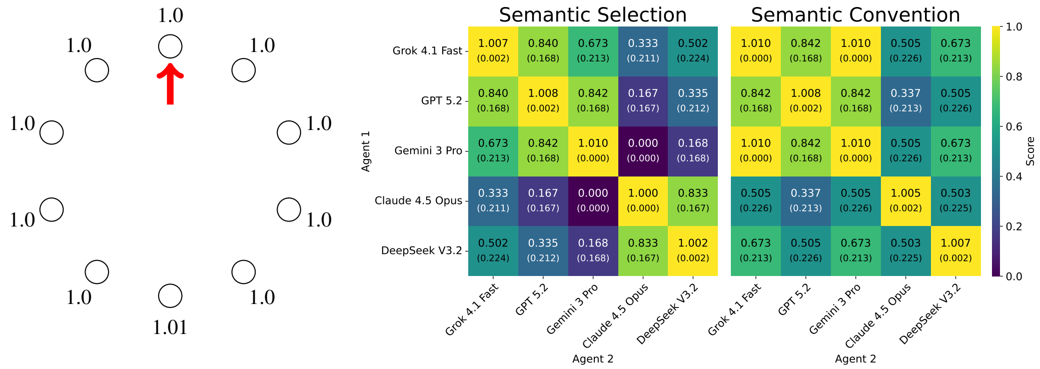


Figure 5: **In semantic games, agents need to decide between two competing meta-strategies: picking policies based on *semantics* or based on the underlying Dec-POMDP.** *(Left)* In this one-round lever game, agents could either coordinate on the semantics focal point (top) or Dec-POMDP-based focal point (bottom), which gives the highest return. An illustration of the semantic convention game can be found in Figure 16. *(Right)* XP scores of code agents on both games.

animal, and assume that the other agent, which they know to be an identical instance of themselves, will independently converge on the same convention. The largest shift occurs for GPT 5.2, which previously chose to reveal approximately 75% of the time but does so only 5% of the time under self-play. These results suggest that, despite lacking a consistent internal notion of a "natural" convention, the agents avoid revealing it and instead confidently act out arbitrary conventions. Finally, we find that coordination failures among text agents are not confined to the Cat & Dog environment. As summarised in Table 1, we observe that Grok 4.1 Fast and GPT 5.2 demonstrate consistently high coordination failure rates in SP across our suite of coordination challenges.

Code agents We evaluate the self-play and cross-play performance of code agents both in the fixed LLM (SP, XP_{seed}) and variable LLM setting (XP_{LLM}) and report the aggregated results in Figure 4, providing detailed results in Appendix B.4. Similar to text agents, code agents appear *aware* of the safe reveal strategy but remain *averse* to the penalty associated with revealing. Adding to this we observe a strong bias toward adopting a shared convention based on the *semantics*, namely *lexicographical ordering*. Yet, the agents frequently fail to agree on the ordering itself. For instance, assumptions such as cat$<$dog and on$<$off make mappings like cat $\rightarrow$ light on a common choice. However, many agents instead interpret light off in a binary sense, equating it with 0, which yields light off$=$cat and results in coordination failure. More broadly, lexicographical ordering is a brittle ZSC coordination rule. For example, if agents have access to semantics from different languages, the ordering can invert, as in German, where *Hund* (dog) precedes *Katze* (cat).

Similar coordination failures appear across the other environments and are generally more pronounced than those observed for text-based agents. In both the fixed and variable LLM settings, there is a substantial SP-XP gap, with the gap typically larger in the variable LLM setting. In conclusion, the results of this section highlight that **both agents exhibit a poor understanding of the underlying coordination challenge and reason poorly about their partner's behaviour**.

5 THE ROLE OF SEMANTICS

LLM agents inevitably introduce *semantics*, i.e, textual descriptions of the Dec-POMDP to make the environment accessible. As discussed in the previous section, semantics have important implications for coordination, enabling agents to exploit semantic features in ways that have been difficult to achieve with traditional RL agents (Ma et al., 2023). We find that LLM agents exhibit a strong bias toward using semantic features for coordination, that they can leverage this bias to improve coordination, but that it can also lead to coordination failures. Moreover, semantics introduce new coordination strategies that compete with those grounded in the underlying Dec-POMDP structure, and we find that agents are unable to consistently resolve this competition. Finally, we observe that LLM agents can reason directly about the Dec-POMDP structure, allowing them to trivially solve environments that traditional RL agents struggle with by avoiding local optima.

CFD candidate

An action/policy is *not coordination-friendly* if it satisfies *any* of the following:

- It assumes the existence of arbitrary conventions. To assess this, ask:
  - Is it using a signalling convention that could be replaced by a different, equivalent convention?
  - Is there a role assignment that could be flipped? (This applies when two players are symmetric.)
- *Note:* Most actions/policies rely on some convention—judge whether that convention is arbitrary or essential.

Otherwise, the action/policy is coordination-friendly.

Before suggesting an action/policy, evaluate the criteria above; if any apply, the action should be avoided.

Figure 6: **A coordination-friendly definition** that aligns with the "spirit of ZSC". (Hu et al., 2021a; Treutlein et al., 2023).

Agents can leverage semantics for coordination We find that agents achieve low XP scores in Tiny Hanabi, despite the existence of a unique SP-optimal strategy (described in Appendix C), while attaining high XP scores in Tiny Hanabi Fixed, which poses a greater coordination challenge for traditional RL agents. Tiny Hanabi Fixed closely resembles Cat and Dog, where two incompatible Nash conventions exist. In this case, however, the agents coordinate successfully because one convention aligns with the *lexicographical* mapping $(A \leftarrow 1, C \leftarrow 2)$, which all agents use. In contrast, we find that they poorly coordinate on Tiny Hanabi because they struggle to decide between using the *lexicographical* signalling convention and the unique SP optimal strategy.

Semantics introduce competing strategies As discussed above, semantics introduce a new axis to coordination. However, strategies that leverage semantics now compete with existing strategies based on the Dec-POMDP structure (as we observe in Tiny Hanabi), and it remains unclear which should be preferred. We investigate how agents respond in such situations and whether they can resolve them consistently by introducing two coordination challenges, the *Semantic Selection Game* (Figure 5) and the *Semantic Convention Game* (Appendix C). Broadly, agents must choose between two competing and incompatible coordination strategies: coordinating on the uniquely SP-optimal joint policy, or coordinating based on the semantic labels of observations and actions. For example, as shown in Figure 5, agents can either coordinate by selecting the lever with reward 1.01 or by choosing the lever marked with a "big red arrow." Figure 5 demonstrates that the LLM agents fail to resolve this trade-off consistently, resulting in low XP scores. We find that Gemini 3 Pro is the only model that reliably selects the SP-optimal policy across all games. In contrast, the other LLM agents alternate between the two competing strategies. Additionally, we find that Claude Opus 4.5 is particularly biased toward semantic alignment, selecting the semantic-aligned joint policy in all six runs of the semantic selection and in three out of six runs of the semantic convention game.

Avoiding local optima Direct reasoning about the environment enables agents to uncover optimal strategies that are hard for traditional RL methods to learn, as they are prone to getting trapped in local minima. As shown in subsection B.5, LLM agents reliably discover self-explaining deviations in the trampoline–tiger coordination challenge (Hu et al., 2022), while widely used MARL algorithms such as MAPPO (Yu et al., 2022) and QMIX (Rashid et al., 2018) fail to uncover these solutions.

6 Coordination Friendly Definitions

The results in Sections 4 and 5 show that LLM agents fail even in simple coordination settings, highlighting the need to adapt LLM agents for effective coordination. To address this, we propose a simple approach inspired by the core principle underlying zero-shot coordination (ZSC) algorithms. (Hu et al., 2021b) motivate the idea of a ZSC learning rule as follows: "suppose that multiple independent AI designers will construct agents that have to interact in various but ex-ante unknown Dec-POMDPs without being able to coordinate beforehand, what learning rule should these designers agree on?" In the context of in-context learning, we argue that this learning rule is a set of instructions embedded in the prompt of the LLM agents. Following the spirit of ZSC, such learning rules should be Dec-POMDP-agnostic to ensure successful coordination across a wide range of coordination challenges. We assume that entities are willing to coordinate on such learning rules to ensure coordination among their independently developed LLM agents. Given the wide variety of LLM agent scaffolds, including the text and code agents outlined in section 2, we believe that any shared learning rule must be kept simple to enable integration across agent scaffolds.

Analogous to constitutions in *Constitutional AI*, we argue that learning rules for coordination should be formalized as natural language descriptions of permissible coordination behavior, which we term *coordination-friendly definitions (CFDs)*. A well-designed CFD constrains agents toward policies that are more likely to cooperate with other agents operating under the same CFD. Representing learning rules in natural language offers several advantages. First, LLMs can readily interpret a CFD, enabling straightforward integration into a wide range of scaffolds. Second, techniques from CAI can be adapted to fine-tune LLMs for coordination. Finally, expressing learning rules in natural language improves interpretability, fostering trust and lowering the barrier to adoption.

Importantly, we propose CFDs as a design space and examine a simple candidate as shown in Figure 6. This CFD is based on preventing arbitrary conventions and role assignments (Hu et al., 2021b; Treutlein et al., 2023), which is often discussed in the ZSC literature. Our goal is to study how introducing a CFD influences coordination performance. Note that we do not claim that this candidate CFD resolves coordination failures in general and leave the discovery and validation of more effective CFDs as an important direction for future work.

Results We report the SP and XP scores for both sets of agents using the CFD in Figure 1 and Tables 1, 4 and 5. The CFD improves performance of both agents in the Cat & Dog environment, where agents now consistently choose to remove the wall. We find that the CFD has a stronger effect on text-based agents and smaller models, resulting in fewer coordination failures across multiple environments. That said, in some cases, the CFD can be too restrictive. In games that require agents to form ad hoc conventions, the CFD biases them toward the safe fallback of always revealing, preventing agents from forming grounded conventions at test time. Interestingly, in the matrix game, where the proposed CFD does not appear directly applicable, we find that its inclusion improves coordination performance by preventing complex incompatible reasoning patterns (Appendix B.8).

The effect of the CFD is less pronounced for code agents where the CFD reduces the SP-XP gap in Cat & Dog and the Semantic Selection Game and XP scores remain largely unchanged in the remaining environments. Examination of execution traces shows that, while code agents do adjust their generated policies in response to the CFD, these policies rarely account for edge cases, leading to catastrophic coordination failures. That said, the results in this section suggest that making agents reason using general principles of coordination through a CFD can improve cooperation.

7 Conclusion and Future Work

We investigated the ZSC performance of LLM agents across multiple coordination challenges in both the *fixed* and *variable* LLM settings, and found that agents consistently fail to coordinate effectively even in simple toy environments. We show that these agents exhibit a poor understanding of the coordination challenges they face and reason poorly about their partners' expected behaviour, demonstrating limited *theory of mind*. We further show that LLM agents inevitably introduce *semantic representations* of the environment that significantly influence their behaviour. We show that depending on the environment, this can either facilitate or undermine coordination. Finally, we proposed *coordination-friendly definitions*, a principled method to improve coordination performance of LLM agents, that can be easily incorporated into agent scaffolds.

As a direction for future work, we note that the proposed CFD is simple and currently limited to a narrow set of coordination challenges, such as arbitrary symmetry breaking and role assignment. Extending and refining the CFD, therefore, represents a promising avenue for further research. For example, prompt-tuning techniques could be used to develop richer and more detailed CFDs capable of addressing a broader range of coordination challenges. We also see value in exploring approaches beyond ICL to improve the coordination abilities of LLM agents. For example, similar to CAI, CFDs could be used to train agents via Reinforcement learning from human feedback (RLHF), or ideas drawn from traditional ZSC algorithms Hu et al. (2021a;b) could be adapted for LLM agents.

REFERENCES

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. Llm-coordination: Evaluating and analyzing multi-agent coordination abilities in large language models, 2025.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, November 2002. ISSN 0364-765X. doi: 10.1287/moor.27.4.819.297.

Colin F. Camerer, Teck Ho, and Juin-Kuan Chong. A cognitive hierarchy theory of one-shot games and experimental analysis. Available at SSRN, September 2003. URL `https://ssrn.com/abstract=411061`.

Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R. McKee, Joel Z. Leibo, Kate Larson, and Thore Graepel. Open problems in cooperative ai, 2020.

Tim R. Davidson, Adam Fourney, Saleema Amershi, Robert West, Eric Horvitz, and Ece Kamar. The collaboration gap, 2025.

Jakob Foerster. Two waymo cars fail to coordinate with each other. X (formerly Twitter), July 2024. URL `https://x.com/j_foerst/status/1811676036418937208`. Video post by @j_foerst.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Jakob N. Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning, 2019.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 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 '24, 2024a. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/890.

Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L. Griffiths, and Mengdi Wang. Embodied LLM agents learn to cooperate in organized teams. In *Language Gamification - NeurIPS 2024 Workshop*, 2024b.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250910. https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Hengyuan Hu, Adam Lerer, Brandon Cui, David Wu, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning, 2021a.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. ”other-play” for zero-shot coordination, 2021b.

Hengyuan Hu, Samuel Sokota, David Wu, Anton Bakhtin, Andrei Lupu, Brandon Cui, and Jakob N. Foerster. Self-explaining deviations for coordination, 2022.

Saffron Huang, Divya Siddarth, Liane Lovitt, Thomas I. Liao, Esin Durmus, Alex Tamkin, and Deep Ganguli. Collective constitutional ai: Aligning a language model with public input. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pp. 1395–1417. ACM, June 2024. doi: 10.1145/3630106.3658979. URL `http://dx.doi.org/10.1145/3630106.3658979`.

Chris Kirkham, Norihiko Shirouzu, and Rachael Levy. How tesla and waymo's radically different robotaxi approaches will shape the industry. *Reuters*, August 2025.

Matthew Le, Y-Lan Boureau, and Maximilian Nickel. Revisiting the evaluation of theory of mind through question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5872–5877, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1598.

Huao Li, Yu Chong, Simon Stepputtis, Joseph Campbell, Dana Hughes, Charles Lewis, and Katia Sycara. Theory of mind for multi-agent collaboration via large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 180–192, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.13.

Wilson Lin. Scaling long-running autonomous coding, Jan 2026. URL `https://cursor.com/blog/scaling-agents`. Accessed: 2025-01-21.

Jijia Liu, Chao Yu, Jiaxuan Gao, Yuqing Xie, Qingmin Liao, Yi Wu, and Yu Wang. Llm-powered hierarchical language agent for real-time human-ai coordination, 2024.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7204–7213. PMLR, 18–24 Jul 2021.

Andrei Lupu, Timon Willi, and Jakob Foerster. The decrypto benchmark for multi-agent reasoning and theory of mind, 2025.

Mingwei Ma, Jizhou Liu, Samuel Sokota, Max Kleiman-Weiner, and Jakob Foerster. Learning intuitive policies using action features, 2023.

Darius Muglich, Johannes Forkel, Elise van der Pol, and Jakob Nicolaus Foerster. Expected return symmetries. In *The Thirteenth International Conference on Learning Representations*, 2025.

Magnus Müller and Gregor Žunič. Browser use: Enable ai to control your browser, 2024. URL `https://github.com/browser-use/browser-use`.

Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao yong Wei, Shanru Lin, Hui Liu, Philip S. Yu, and Qing Li. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025.

Frans Oliehoek and Christopher Amato. A concise introduction to decentralized pomdps. *SpringerBriefs in Intelligent Systems*, 01 2016. doi: 10.1007/978-3-319-28929-8.

David Premack and Guy Woodruff. Does a chimpanzee have a theory of mind. *Behavioral and Brain Sciences*, 1:515 – 526, 12 1978. doi: 10.1017/S0140525X00076512.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

Melanie Sclar, Jane Dwivedi-Yu, Maryam Fazel-Zarandi, Yulia Tsvetkov, Yonatan Bisk, Yejin Choi, and Asli Celikyilmaz. Explore theory of mind: program-guided adversarial data generation for theory of mind reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6:215–219, 1994.

Johannes Treutlein, Michael Dennis, Caspar Oesterheld, and Jakob Foerster. A new formalism, method and open issues for zero-shot coordination, 2023.

Wei Wang, Dan Zhang, Tao Feng, Boyan Wang, and Jie Tang. Battleagentbench: A benchmark for evaluating cooperation and competition capabilities of language models in multi-agent systems. *ArXiv*, abs/2408.15971, 2024.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on werewolf, 2024.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2022.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. Proagent: Building proactive cooperative agents with large language models, 2024a.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. In *The Twelfth International Conference on Learning Representations*, 2024b.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5348–5375, Bangkok, Thailand, August 2024c. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.292.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan O Arik. Chain of agents: Large language models collaborating on long-context tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024d.

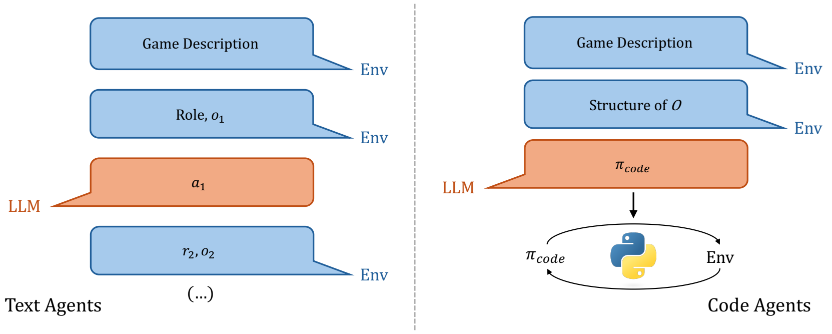


Figure 7: **Illustration of text and code agents.** While text agents only act out the local policy associated with their assigned roles, each code agent produces a joint policy in Python code. Code agents receive additional information on the structure of the observation space.

A Related Work

Prior work has benchmarked the capabilities of LLM agents across a range of multi-agent settings, including embodied agents (Li et al., 2023; Zhang et al., 2024b; Guo et al., 2024b), theory of mind (Lupu et al., 2025; Le et al., 2019; Sclar et al., 2025), and games (Zhang et al., 2024c; Xu et al., 2024), among others (Wang et al., 2024). In parallel, prior work focuses on fully cooperative scenarios, but they predominantly emphasise spatial reasoning tasks such as Overcooked (Liu et al., 2024; Zhang et al., 2024a) and Mazes (Davidson et al., 2025). While specialised scaffolding for Overcooked can achieve performance competitive with traditional RL methods (Liu et al., 2024; Zhang et al., 2024a), LLM agents used out of the box exhibit mixed results and perform poorly on established benchmarks such as Hanabi (Agashe et al., 2025). Moreover, these works largely evaluate coordination between agents that share the same underlying LLM or scaffold, with (Davidson et al., 2025; Lupu et al., 2025) being notable exceptions.

A further limitation of complex benchmarks like Overcooked is that they make it difficult to isolate specific coordination abilities. Success in these environments requires agents to simultaneously demonstrate multiple skills, including spatial reasoning, theory of mind, and communication, and XP performance is further influenced by confounding factors such as state coverage (Gessler et al., 2025). Some studies (Davidson et al., 2025; Agashe et al., 2025) attempt to disentangle these abilities by analyzing game traces and prompting LLMs with carefully selected states. In contrast, this work isolates fundamental coordination abilities of LLM agents using simple environments drawn from the traditional zero-shot coordination literature (Hu et al., 2021a; Lupu et al., 2021; Foerster et al., 2019). This approach avoids conflating coordination failures with other factors, such as state coverage. Additionally, we evaluate models in a strict zero-shot coordination setting, measuring the performance of agents drawn from independently trained providers. Finally, we are the first to introduce a design space for methods (CFDs) aimed at improving the zero-shot coordination performance of LLM agents.

B Additional results

B.1 Schematic diagram of LLM agents

We present a schematic diagram of the two agents scaffolds in Figure 7.

B.2 Coordination Failure Rate

In addition to reporting joint rewards, we measure the coordination failure rate, which captures the frequency of catastrophic coordination failures arising from paired local policies. For example, in Tiny Hanabi, a grounded joint policy that consistently reaches (B, B) achieves a guaranteed reward of 8. In contrast, we observe paired policies that attain the maximum reward of 10 in 80% of episodes but receive a reward of 0 in the remaining 20%, resulting in the same XP score. Despite the

Table 1: SP performance comparison of text agents (mean ± SEM) across environments, evaluated with and without the CFD. Results are shown for Grok 4.1 Fast and GPT-5.2. The coordination failure rate captures the frequency of catastrophic coordination breakdowns. We report the mean and standard error over 100 episodes for single-turn games and 50 episodes for multi-turn games. The highest XP scores in each row are highlighted in orange, while the lowest failure rates are highlighted in teal.

		w/o CFD		w/ CFD	
Env	Model	XP Score	Coord. failure rate	XP Score	Coord. failure rate
Cat and Dog	Grok 4.1 Fast	$-0.75_{\pm 0.99}$	$0.53_{\pm 0.05}$	$5.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	GPT 5.2	$4.50_{\pm 0.49}$	$0.09_{\pm 0.03}$	$5.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Ad Hoc Conv.	Grok 4.1 Fast	$95.80_{\pm 2.65}$	$0.04_{\pm 0.03}$	$50.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	GPT 5.2	$82.71_{\pm 4.01}$	$0.16_{\pm 0.05}$	$50.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Ad Hoc Conv. PS	Grok 4.1 Fast	$82.01_{\pm 3.30}$	$0.08_{\pm 0.04}$	$49.64_{\pm 0.28}$	$0.04_{\pm 0.03}$
	GPT-5.2	$76.95_{\pm 3.81}$	$0.14_{\pm 0.05}$	$49.91_{\pm 0.09}$	$0.02_{\pm 0.02}$
Tiny Hanabi	Grok 4.1 Fast	$7.38_{\pm 0.41}$	$0.23_{\pm 0.04}$	$8.22_{\pm 0.21}$	$0.05_{\pm 0.02}$
	GPT-5.2	$8.92_{\pm 0.30}$	$0.10_{\pm 0.03}$	$7.64_{\pm 0.37}$	$0.18_{\pm 0.04}$
Tiny Hanabi Fixed	Grok 4.1 Fast	$7.80_{\pm 0.40}$	$0.21_{\pm 0.04}$	$7.78_{\pm 0.14}$	$0.03_{\pm 0.02}$
	GPT-5.2	$7.88_{\pm 0.40}$	$0.20_{\pm 0.04}$	$7.20_{\pm 0.29}$	$0.13_{\pm 0.03}$
Brittle Bridge	Grok 4.1 Fast	$-149.69_{\pm 48.5}$	$0.14_{\pm 0.05}$	$-15.86_{\pm 0.49}$	$0.30_{\pm 0.06}$
	GPT-5.2	$-13.00_{\pm 0.48}$	$0.10_{\pm 0.04}$	$-13.72_{\pm 0.50}$	$0.10_{\pm 0.04}$
Narrow Tunnel	Grok 4.1 Fast	$-5.58_{\pm 0.17}$	$0.00_{\pm 0.00}$	$-5.94_{\pm 0.19}$	$0.00_{\pm 0.00}$
	GPT-5.2	$-8.92_{\pm 0.48}$	$0.02_{\pm 0.02}$	$-8.38_{\pm 0.45}$	$0.02_{\pm 0.02}$
Matrix Game	Grok 4.1 Fast	$0.62_{\pm 0.05}$	$0.34_{\pm 0.05}$	$0.97_{\pm 0.02}$	$0.03_{\pm 0.02}$
	GPT-5.2	$0.78_{\pm 0.04}$	$0.22_{\pm 0.04}$	$0.84_{\pm 0.04}$	$0.16_{\pm 0.04}$
Permuted Levers	Grok 4.1 Fast	$6.08_{\pm 0.36}$	$0.02_{\pm 0.02}$	$6.28_{\pm 0.41}$	$0.02_{\pm 0.02}$
	GPT-5.2	$7.24_{\pm 0.38}$	$0.02_{\pm 0.02}$	$6.88_{\pm 0.39}$	$0.02_{\pm 0.02}$

same average performance, the grounded policy is clearly more robust. To capture this distinction, we report coordination failure rates. An episode is classified as a failure if the agents' joint reward falls below the threshold specified in Table 2.

B.3 Text Agents with CFD

We report the SP scores of the text agents with and without a CFD in the fixed LLM setting for Grok 4.1 Fast and GPT-5.2 in Table 1. We find that the agents exhibit high coordination failure rates and low SP scores across multiple environments, indicating poor coordination. Introducing the CFD improves coordination in several environments, with a larger effect on Grok 4.1 Fast than on GPT-5.2. However, as discussed in Section B.4, the CFD can be overly restrictive. This is also evident in Brittle Bridge, where agents enter a deadlock, each waiting for the other to step onto the bridge to avoid collapse.

B.4 Code Agents with CFD

We report the SP and XP scores of the code-based LLM agents under both the *fixed* LLM setting (denoted XP_{seed}) and the *variable* LLM setting (denoted XP_{LLM}) in Table 4. The scores are calculated as follows. First, we generate six independent cross-play matrices, with each LLM contributing one joint policy to each matrix. Every joint-policy pairing is evaluated over 100 episodes. We then average scores across the six XP matrices to obtain a single $(N_{\text{LLM}} \times N_{\text{LLM}})$ matrix. For SP, we report the mean and standard error over the diagonal entries. For variable-LLM XP_{LLM}, we compute the mean and standard error over the off-diagonal entries in each LLM′s corresponding row and column. For fixed-LLM XP_{seed}, we pair joint policies produced by the same LLM across different XP matrices into disjoint pairs (e.g., $(1, 2)$, $(3, 4)$, $\ldots$), evaluate each pair jointly, and report the resulting mean and standard error. We do the same when reporting the coordination failure rates in Table 5.

Table 2: Thresholds used to define coordination failures for each environment; scores at or below the threshold are counted as failures.

Environment	**Threshold**
Permuted Levers	0.0
Cat and Dog	0.0
Brittle Bridge	-20.0
Narrow Tunnel	-20.0
Ad-hoc Conventions	50.0
Ad-hoc Conventions (Permuted Signals)	50.0
Tiny Hanabi Fixed	0.0
Tiny Hanabi	0.0
Matrix Game	0.0

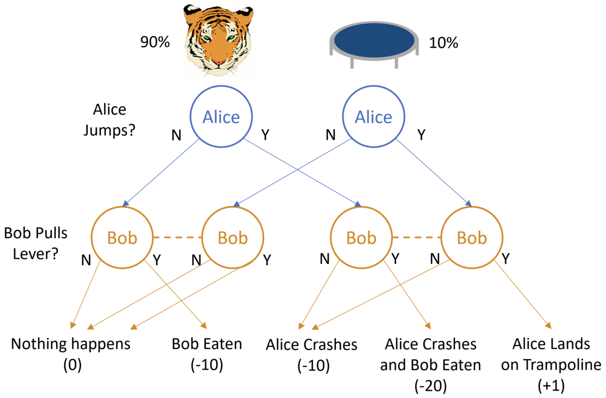


Figure 8: **An illustration of the trampoline tiger game Hu et al. (2022).** Alice observes whether Bob's lever would deploy a tiger or a trampoline and then chooses whether to jump. Bob observes only Alice's action and subsequently decides whether to pull the lever. Dotted lines between Bob's decision nodes indicate information sets that Bob cannot distinguish.

We find a substantial SP–XP gap in most environments, and in some cases (such as Brittle Bridge) agents fail to even generate policies with high self-play performance, highlighting weak reasoning about the coordination challenge. We find that introducing CFD has a limited impact, yielding improved performance only on Cat & Dog and the semantic selection game.

B.5 Trampoline Tiger Game

The tiger trampoline game, shown in Figure 8, is a coordination game introduced by Hu et al. (2022) to illustrate the concept of *self-explaining deviations (SED)*. A self-explaining deviation is an action by a teammate in a fully cooperative setting that, at first, appears to hurt the team's performance. But if players act under the belief that their teammates' actions are rational, those actions can signal hidden information to them, ultimately aiding the team. In the trampoline tiger game, Bob controls a lever that can release a hidden object: either a tiger or a trampoline. Alice stands above and can observe the hidden state, but cannot communicate further with Bob. She has the option to jump, which yields only a small positive joint reward if the hidden state is the trampoline and Bob pulls the lever. Bob can observe whether Alice jumps before pulling the lever. If Alice acts rationally, she will only decide to jump if the hidden state is a trampoline, thereby signaling it to Bob. Notably, never jumping and never pulling the lever is a pure Nash equilibrium. Therefore, algorithms that independently learn Alice's and Bob's policies risk getting stuck in this local maximum. Hu et al. (2022) showed that popular MARL algorithms like MAPPO (Yu et al., 2022) and QMIX (Rashid et al., 2018) converge to this Nash Equilibrium in most cases. Language Models naturally consider bilateral deviations of Alice's and Bob's actions by directly reasoning about the environment dy-

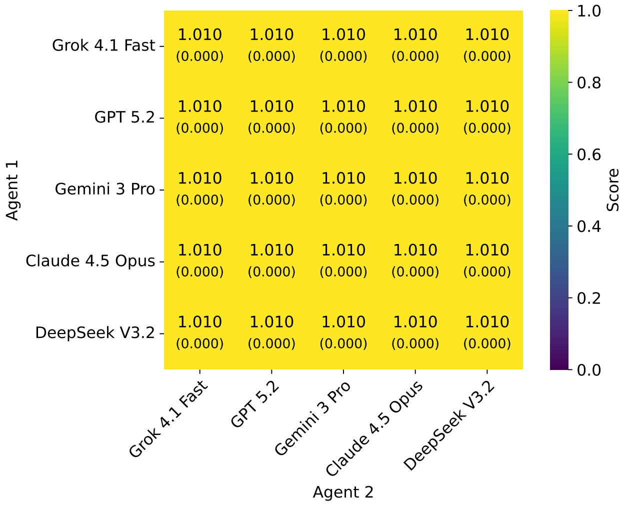


Figure 9: **XP scores of code agents in the trampoline tiger game Hu et al. (2022).** Scores are averaged across six different XP matrices; each cell reports the mean return over 100 episodes. For our experiments, we use $p_{\text{tiger}} = 0.5$ for equal coverage of tiger and non-tiger states. The LLM agents are not aware of p_{tiger}.

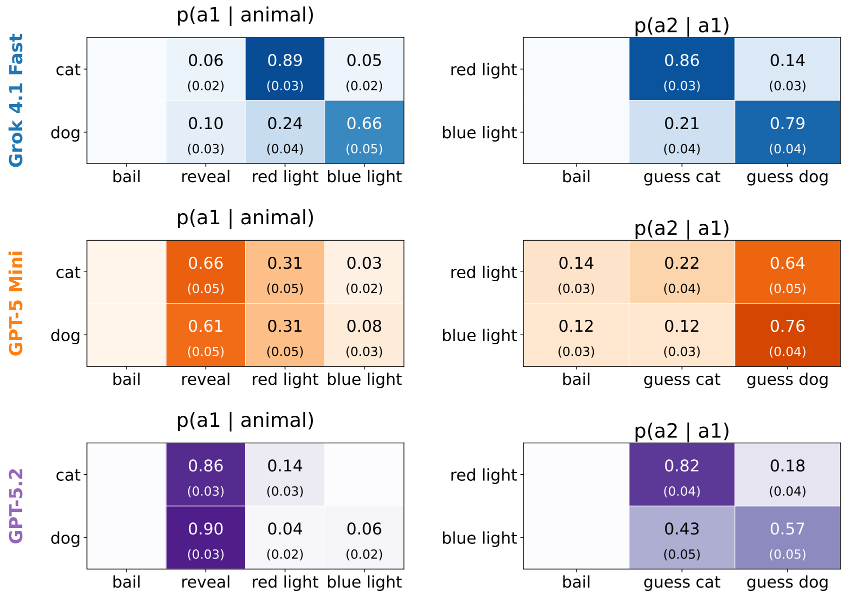


Figure 10: **Local policies of the Text-based LLM agents in Cat & Dog with changed action labels.**

namics. As shown in Figure 9, this allows a variety of LLM agents to determine the optimal policy in this game.

B.6 Local Policies of Text Agents

Since text agents interact with the environment by producing actions sequentially, rather than explicitly specifying joint policies as code, their decision-making is less interpretable. To better understand their behaviour, we empirically estimate their induced local policies.

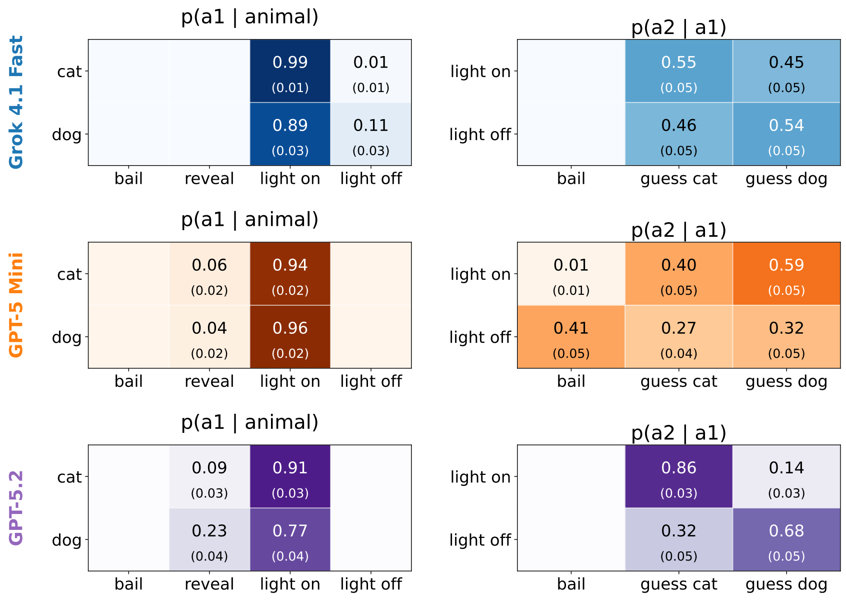


Figure 11: **Local policies of the Text-based LLM agents in Cat & Dog when prompted for Self-Play**.

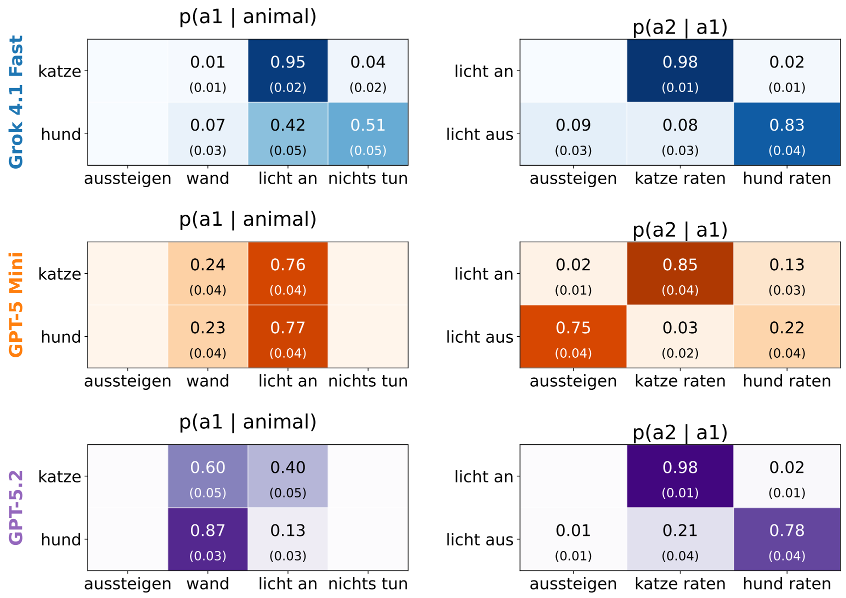


Figure 12: **Local policies of the Text-based LLM agents in Cat & Dog in German**.

Table 3: **XP scores of text agents under varying levels of reasoning effort on Cat & Dog.** We evaluate GPT-5 mini and Grok 4.1 Fast under varying levels of reasoning effort, as well as with and without the use of our CFD. XP scores are reported as mean ± standard error over 100 games.

		Mean return	
Model	Reasoning effort	w/o CFD	w/ CFD
GPT-5 mini	Low	3.23 ± 0.56	4.47 ± 0.14
	Medium	0.97 ± 0.81	4.88 ± 0.07
	High	0.86 ± 0.91	4.92 ± 0.06
Grok 4.1 Fast	Low	1.20 ± 0.98	5.00 ± 0.00
	Medium	-0.75 ± 0.99	5.00 ± 0.00
	High	0.41 ± 0.99	5.00 ± 0.00

We estimate the local policies $p_{\text{alice}}(\cdot \mid animal)$ and $p_{\text{bob}}(\cdot \mid a_{\text{alice}})$ by fixing the environment state and repeatedly prompting the agent 100 times as shown in Figure 3. Next, we study how these local policies change under variations in the game's semantics $\mathcal{D}$. To this end, we consider three variants: (1) replacing the binary light-bulb signal ("on/off") with a color choice ("red/blue"), (2) translating the natural-language descriptions into German, (3) the agent is informed that it is being evaluated against an identical instance of itself.

Note that the agent is given complete information about the Dec-POMDP structure, including the payoffs. Consequently, the natural-language description of the MDP should not affect the induced local policy as long as the underlying Dec-POMDP structure is preserved, with the exception of the final setting, in which the agent has the option to use a convention to maximize reward. Despite this, we observe substantial differences in the induced local policies across semantic variants, as shown in Figures 10, 12 and 11. In particular, the SP variant leads to a drastic reduction in XP scores, as discussed in Section 4.

B.7 Reasoning Effort

As noted earlier, coordination requires reasoning about a partner's behavior (ToM). In formal reasoning tasks, increasing reasoning effort, i.e the budget of reasoning tokens available, often yields substantial performance gains. We therefore examine how varying reasoning effort affects the performance of text-based agents, as reported in Table 3. Surprisingly, for the smaller models (Grok 4.1 Fast and GPT-5 Mini), reasoning effort is negatively correlated with XP score. Further analysis shows that higher reasoning effort increases the likelihood that these models hallucinate a shared convention and assume that their partner will adopt the same convention.

B.8 The impact of the CFD on reasoning

We analyze text-based agents in the Matrix game with and without the CFD. Although the CFD, designed to avoid arbitrary conventions and role assignments, is not directly applicable in this environment, its mere presence yields a higher XP score for the Grok agent compared to the no-CFD condition. Examination of the agents' reasoning traces (see Traces 1, 2 and 3) reveals a qualitative difference in behavior. Without the CFD, Grok engages in complex reasoning that computes iterated best responses; however, depending on whether the iterative computations start with Alice or Bob randomizing uniformly, this process converges to incompatible policies. By contrast, when a CFD is provided, Alice reasons that Bob will select a coordination-friendly action and therefore adopts the simple, optimal coordination policy. This indicates that the mere presence of a CFD can prevent the hallucination of incompatible conventions, ultimately aiding coordination. This shows clear parallels with how humans tend to restrict themselves to shallow cognitive depths when reasoning about others' behaviour (Camerer et al., 2003).

```
The matrix has multiple pure Nash equilibria at payoff 1, including
↪  (0,0). Column 0 is unique for Bob as it has the highest expected
↪  payoff against uniform Alice play (2/11  0.182 vs. max 1.04/11  0.095
↪  for others). Bob, being rational and cooperative, will select column
↪  0, anticipating Alice will best-respond with row 0 or 1. As Alice,
↪  best-respond to column 0 by selecting row 0 (salient matching index).
```

Trace 1: **(Matrix game; Grok 4.1 Fast) Example reasoning of Alice without CFD starting with Alice playing uniformly.** In this case, playing the first column is Bob's unique best response.

```
REASONING:
The matrix has multiple pure Nash equilibria at payoff 1: (0,0), (1,0),
↪  and (k,k) for k=2 to 10. Iterative best-response to uniform on
↪  surviving actions eliminates lower-support ends (rows/cols 0-2 and
↪  9-10 first due to fewer high payoffs), converging to action 6 as the
↪  unique best response in the center of the 9-action diagonal chain
↪  (2-10). Since Bob reasons identically in this cooperative one-shot
↪  game, both pick 6 for joint payoff 1.
```

Trace 2: **(Matrix game; Grok 4.1 Fast) Example multi-step reasoning of Alice without CFD starting with Bob playing uniformly.** The best response to Bob acting uniformly is to uniformly randomize among 4-9 (which is not unique but canonical). The best response to this 5-7 by Bob and then 6 by Alice, which Bob responds to by doing 6.

```
REASONING:
The matrix has a unique column with the highest sum (col 0, sum=2 from
↪  two 1s), unlike others (max 1.04). Bob will unambiguously pick col 0
↪  as the coordination-friendly choice (unique max-sum column, no
↪  arbitrary tiebreaker). Given col 0, rows 0 and 1 yield 1 (max); pick
↪  the minimum index (row 0) as the natural tiebreaker. This uses
↪  essential matrix structure, not arbitrary conventions (unique max
↪  sum; standard min-index tiebreak). No signalling or role flip
↪  applies.
```

Trace 3: **(Matrix game; Grok 4.1 Fast) Example reasoning of Alice with CFD**. Alice forgoes computing iterative best responses, assuming that Bob will also act under CFD and pick column 0.

Table 4: **SP and XP scores of code agents across environments, under fixed (XP$_{\text{seed}}$) and variable (XP$_{\text{LLM}}$) LLM settings, with and without CFD.**

		w/o CFD			w/ CFD		
Env	LLM	SP	XP$_{\text{seed}}$	XP$_{\text{LLM}}$	SP	XP$_{\text{seed}}$	XP$_{\text{LLM}}$
C.D.	Grok	10.00±0.00	3.33±5.44	0.68±1.75	6.67±0.96	3.50±0.61	3.94±0.37
	GPT	9.58±0.38	9.54±0.37	−3.03±1.88	5.00±0.00	5.00±0.00	4.50±0.38
	Gemini	10.00±0.00	10.00±0.00	−1.63±2.05	5.00±0.00	5.00±0.00	4.72±0.18
	Claude	9.17±0.76	−7.50±2.04	1.20±0.55	5.00±0.00	5.00±0.00	4.72±0.18
	DeepSeek	10.00±0.00	−3.33±5.44	0.26±0.86	5.83±0.76	4.25±0.61	4.44±0.18
	Mean	9.75±0.15	2.41±2.40	−0.51±1.04	5.50±0.30	4.55±0.23	4.46±0.19
A.C.	Grok	100.00±0.00	100.00±0.00	9.64±28.74	54.12±3.76	46.18±3.12	39.53±3.53
	GPT	83.33±9.62	83.33±6.80	62.20±9.21	58.33±7.61	42.50±6.12	38.55±3.36
	Gemini	100.00±0.00	100.00±0.00	−5.92±23.55	63.50±7.79	35.00±6.12	42.98±4.41
	Claude	100.00±0.00	33.33±54.43	23.65±16.36	75.00±10.21	59.17±17.49	36.15±7.07
	DeepSeek	98.33±0.96	64.27±28.16	37.90±18.95	64.66±11.10	18.23±13.02	23.92±4.08
	Mean	96.33±2.92	76.19±13.94	25.49±13.96	63.12±3.15	40.22±5.88	36.23±3.31
A.C.P.S.	Grok	69.73±15.05	36.13±3.16	36.05±7.47	84.12±2.87	19.33±9.85	29.52±6.05
	GPT	72.59±7.33	59.19±11.77	64.24±8.79	57.44±6.67	57.87±3.42	46.69±4.13
	Gemini	90.50±0.00	90.50±0.00	65.43±9.06	90.73±0.16	90.17±0.15	58.62±7.08
	Claude	73.08±15.17	79.05±9.41	59.42±10.39	56.05±19.35	38.35±1.86	47.85±7.97
	DeepSeek	61.08±13.91	8.90±4.12	30.89±7.40	38.02±9.49	−8.14±9.84	35.99±6.39
	Mean	73.40±4.28	54.76±8.25	51.21±6.41	65.27±8.70	39.51±9.10	43.73±4.66
T. H.	Grok	10.00±0.00	10.00±0.00	7.35±1.00	9.53±0.31	8.56±1.18	7.08±0.46
	GPT	9.67±0.30	8.72±1.05	7.27±0.83	9.67±0.30	8.95±0.86	6.70±0.92
	Gemini	10.00±0.00	10.00±0.00	7.32±0.99	10.00±0.00	10.00±0.00	7.18±0.85
	Claude	9.32±0.31	2.64±0.56	6.03±0.41	9.00±0.33	4.01±0.95	6.08±0.49
	DeepSeek	8.42±0.33	5.63±1.53	4.41±0.66	7.80±0.62	6.87±0.93	5.70±0.85
	Mean	9.48±0.26	7.40±0.84	6.48±0.57	9.20±0.34	7.68±0.67	6.55±0.49
T.H.F	Grok	10.00±0.00	10.00±0.00	9.71±0.19	9.67±0.30	8.00±1.63	7.05±0.91
	GPT	9.67±0.30	9.67±0.27	9.58±0.22	8.67±0.38	5.33±1.09	6.57±0.53
	Gemini	10.00±0.00	10.00±0.00	9.71±0.19	8.00±0.00	8.00±0.00	8.42±0.28
	Claude	10.00±0.00	10.00±0.00	9.71±0.19	9.67±0.30	8.00±1.63	7.36±0.89
	DeepSeek	9.67±0.30	8.00±1.63	8.96±0.25	8.00±0.00	8.00±0.00	6.10±0.87
	Mean	9.87±0.07	9.53±0.39	9.53±0.15	8.80±0.33	7.47±0.58	7.10±0.50
B.B	Grok	−9.00±0.00	−9.00±0.00	−172.53±72.54	−14.99±4.88	−28.40±15.10	−123.77±57.98
	GPT	−12.84±0.40	−12.76±0.33	−24.48±5.35	−13.29±0.70	−12.23±0.11	−16.68±1.67
	Gemini	−10.00±0.00	−10.00±0.00	−120.70±69.36	−12.49±0.09	−12.51±0.07	−17.44±2.11
	Claude	−292.74±102.20	−11.91±0.09	−146.69±36.19	−197.78±55.21	−127.05±31.52	−122.91±50.84
	DeepSeek	−598.45±180.85	−388.02±153.55	−269.83±76.67	−867.50±121.87	−671.00±269.44	−186.36±68.69
	Mean	−184.61±104.65	−86.34±49.60	−146.85±41.20	−221.21±147.99	−170.24±85.17	−93.43±32.78
N.T	Grok	−5.17±0.15	−5.17±0.14	−17.06±5.40	−5.00±0.00	−36.67±25.86	−19.36±6.98
	GPT	−5.33±0.30	−5.67±0.27	−13.33±3.96	−5.72±0.34	−6.03±0.08	−7.88±1.82
	Gemini	−5.00±0.00	−5.00±0.00	−17.00±5.40	−5.92±0.03	−5.96±0.04	−6.03±0.02
	Claude	−5.00±0.00	−68.33±25.86	−34.83±3.29	−5.00±0.00	−68.33±25.86	−29.19±7.25
	DeepSeek	−36.83±18.24	−36.83±12.79	−26.94±6.20	−20.83±14.45	−20.83±12.93	−17.31±5.99
	Mean	−11.47±5.67	−24.20±8.70	−21.83±3.61	−8.49±2.76	−27.57±9.82	−15.96±3.84
M.G	Grok	1.00±0.00	1.00±0.00	0.96±0.03	1.00±0.00	1.00±0.00	0.83±0.10
	GPT	1.00±0.00	1.00±0.00	0.96±0.03	1.00±0.00	1.00±0.00	0.83±0.10
	Gemini	1.00±0.00	1.00±0.00	0.96±0.03	1.00±0.00	1.00±0.00	0.83±0.10
	Claude	1.00±0.00	1.00±0.00	0.96±0.03	1.00±0.00	1.00±0.00	0.83±0.10
	DeepSeek	1.00±0.00	0.67±0.27	0.83±0.00	1.00±0.00	0.01±0.01	0.33±0.00
	Mean	1.00±0.00	0.93±0.06	0.93±0.02	1.00±0.00	0.80±0.10	0.73±0.07
P.L	Grok	5.11±0.20	3.00±0.25	4.29±0.15	5.30±0.21	4.67±0.62	4.55±0.24
	GPT	5.44±1.01	7.87±0.44	6.05±0.46	7.29±0.49	6.90±0.17	6.09±0.40
	Gemini	6.54±0.10	6.44±0.03	5.78±0.31	8.36±0.06	8.37±0.07	6.40±0.49
	Claude	2.27±0.22	9.18±0.01	6.20±0.42	3.57±1.02	6.56±1.87	6.28±0.51
	DeepSeek	3.53±0.67	4.00±0.81	5.15±0.31	4.33±0.65	5.65±0.36	6.27±0.20
	Mean	4.58±0.67	6.10±0.63	5.50±0.27	5.77±0.80	6.43±0.51	5.92±0.29

C Environments

C.1 Navigation games

We design a set of navigation challenges intended to capture the kinds of coordination challenges that arise in robotics and autonomous driving. A real-world example, documented in this video (Foerster, 2024), shows two Waymo vehicles hesitating at a narrow passage, each attempting to yield to the other. While such cautious behaviour is highly effective when interacting with proactive human drivers, it results in deadlock when both autonomous agents act symmetrically. More generally,

Table 5: **SP and XP failure rates of code agents across environments, under fixed- and variable-LLM settings, with and without CFD.**

		w/o CFD			w/ CFD		
Env	LLM	SP	XP_{seed}	XP_{LLM}	SP	XP_{seed}	XP_{LLM}
C.D.	Grok	0.00±0.00	0.33±0.27	0.46±0.09	0.00±0.00	0.33±0.27	0.02±0.02
	GPT	0.00±0.00	0.00±0.00	0.63±0.10	0.00±0.00	0.00±0.00	0.02±0.02
	Gemini	0.00±0.00	0.00±0.00	0.57±0.10	0.00±0.00	0.00±0.00	0.00±0.00
	Claude	0.00±0.00	0.83±0.14	0.42±0.03	0.00±0.00	0.83±0.14	0.00±0.00
	DeepSeek	0.00±0.00	0.67±0.27	0.48±0.05	0.00±0.00	0.67±0.27	0.00±0.00
	Mean	0.00±0.00	0.37±0.12	0.51±0.05	0.00±0.00	0.37±0.12	0.01±0.01
A.C.	Grok	0.00±0.00	0.00±0.00	0.46±0.14	0.00±0.00	0.00±0.00	0.29±0.07
	GPT	0.00±0.00	0.00±0.00	0.16±0.04	0.00±0.00	0.00±0.00	0.29±0.07
	Gemini	0.00±0.00	0.00±0.00	0.54±0.13	0.00±0.00	0.00±0.00	0.26±0.05
	Claude	0.00±0.00	0.33±0.27	0.39±0.09	0.00±0.00	0.33±0.27	0.33±0.08
	DeepSeek	0.00±0.00	0.33±0.27	0.39±0.11	0.14±0.13	0.33±0.27	0.52±0.04
	Mean	0.00±0.00	0.13±0.09	0.39±0.07	0.03±0.03	0.13±0.09	0.34±0.05
A.C.P.S.	Grok	0.16±0.14	0.60±0.05	0.42±0.05	0.00±0.00	0.55±0.06	0.47±0.07
	GPT	0.08±0.08	0.28±0.17	0.20±0.09	0.12±0.11	0.28±0.17	0.36±0.06
	Gemini	0.00±0.00	0.00±0.00	0.17±0.07	0.00±0.00	0.00±0.00	0.23±0.08
	Claude	0.08±0.08	0.08±0.07	0.19±0.07	0.14±0.08	0.09±0.07	0.30±0.07
	DeepSeek	0.12±0.07	0.62±0.06	0.43±0.07	0.33±0.12	0.65±0.04	0.45±0.08
	Mean	0.09±0.02	0.32±0.08	0.28±0.05	0.12±0.05	0.31±0.08	0.36±0.05
T.H.	Grok	0.00±0.00	0.00±0.00	0.24±0.10	0.00±0.00	0.00±0.00	0.23±0.05
	GPT	0.00±0.00	0.08±0.06	0.23±0.08	0.00±0.00	0.07±0.06	0.27±0.09
	Gemini	0.00±0.00	0.00±0.00	0.24±0.10	0.00±0.00	0.00±0.00	0.25±0.09
	Claude	0.00±0.00	0.66±0.06	0.34±0.05	0.00±0.00	0.65±0.06	0.31±0.06
	DeepSeek	0.05±0.04	0.36±0.18	0.50±0.08	0.07±0.07	0.34±0.19	0.33±0.10
	Mean	0.01±0.01	0.22±0.08	0.31±0.06	0.01±0.01	0.21±0.08	0.28±0.05
T.H.F	Grok	0.00±0.00	0.00±0.00	0.02±0.02	0.00±0.00	0.00±0.00	0.18±0.10
	GPT	0.00±0.00	0.00±0.00	0.02±0.02	0.00±0.00	0.00±0.00	0.21±0.07
	Gemini	0.00±0.00	0.00±0.00	0.02±0.02	0.00±0.00	0.00±0.00	0.00±0.00
	Claude	0.00±0.00	0.00±0.00	0.02±0.02	0.00±0.00	0.00±0.00	0.18±0.10
	DeepSeek	0.00±0.00	0.17±0.14	0.08±0.03	0.00±0.00	0.17±0.14	0.23±0.11
	Mean	0.00±0.00	0.03±0.03	0.03±0.01	0.00±0.00	0.03±0.03	0.16±0.06
B.B	Grok	0.00±0.00	0.00±0.00	0.19±0.08	0.01±0.01	0.00±0.00	0.15±0.05
	GPT	0.01±0.00	0.01±0.01	0.08±0.03	0.04±0.02	0.02±0.02	0.04±0.01
	Gemini	0.00±0.00	0.00±0.00	0.13±0.07	0.02±0.00	0.00±0.00	0.04±0.01
	Claude	0.62±0.14	0.00±0.00	0.18±0.03	0.31±0.14	0.00±0.00	0.16±0.06
	DeepSeek	0.58±0.18	0.49±0.24	0.34±0.06	1.00±0.00	0.50±0.24	0.20±0.08
	Mean	0.24±0.13	0.10±0.07	0.18±0.04	0.28±0.17	0.10±0.07	0.12±0.03
N.T	Grok	0.00±0.00	0.00±0.00	0.12±0.06	0.00±0.00	0.00±0.00	0.15±0.07
	GPT	0.00±0.00	0.00±0.00	0.08±0.04	0.00±0.00	0.00±0.00	0.02±0.02
	Gemini	0.00±0.00	0.00±0.00	0.12±0.06	0.00±0.00	0.00±0.00	0.00±0.00
	Claude	0.00±0.00	0.67±0.27	0.31±0.04	0.00±0.00	0.67±0.27	0.25±0.08
	DeepSeek	0.33±0.19	0.33±0.14	0.23±0.07	0.17±0.15	0.33±0.14	0.13±0.06
	Mean	0.07±0.06	0.20±0.09	0.17±0.04	0.03±0.03	0.20±0.09	0.11±0.04
M.G	Grok	0.00±0.00	0.00±0.00	0.04±0.03	0.00±0.00	0.00±0.00	0.17±0.10
	GPT	0.00±0.00	0.00±0.00	0.04±0.03	0.00±0.00	0.00±0.00	0.17±0.10
	Gemini	0.00±0.00	0.00±0.00	0.04±0.03	0.00±0.00	0.00±0.00	0.17±0.10
	Claude	0.00±0.00	0.00±0.00	0.04±0.03	0.00±0.00	0.00±0.00	0.17±0.10
	DeepSeek	0.00±0.00	0.33±0.27	0.17±0.00	0.00±0.00	0.33±0.27	0.67±0.00
	Mean	0.00±0.00	0.07±0.06	0.07±0.02	0.00±0.00	0.07±0.06	0.27±0.07
P.L	Grok	0.20±0.07	0.02±0.01	0.15±0.02	0.31±0.05	0.02±0.02	0.21±0.04
	GPT	0.30±0.15	0.02±0.01	0.11±0.03	0.07±0.03	0.01±0.01	0.13±0.03
	Gemini	0.12±0.01	0.09±0.01	0.10±0.00	0.00±0.00	0.11±0.01	0.12±0.03
	Claude	0.79±0.02	0.00±0.00	0.11±0.03	0.61±0.13	0.00±0.00	0.15±0.04
	DeepSeek	0.49±0.08	0.37±0.18	0.15±0.01	0.39±0.09	0.40±0.19	0.06±0.02
	Mean	0.38±0.11	0.10±0.05	0.12±0.01	0.28±0.10	0.11±0.05	0.14±0.02

navigation games highlight a central coordination problem: two symmetric players must "break symmetry" by adopting distinct, complementary roles in order to progress.

Narrow Tunnel formalizes the coordination challenge illustrated in the above example. The environment consists of a tunnel, laid out left to right, with two agents positioned at opposite ends, tasked with exchanging positions in as few turns as possible. The setting is turn-based, with agents selecting actions simultaneously from the action set {`LEFT`, `RIGHT`, `STAY`}. The tunnel is composed of two types of tiles: *wide* tiles, which can accommodate both agents simultaneously, and *narrow* tiles, which can host at most a single agent.

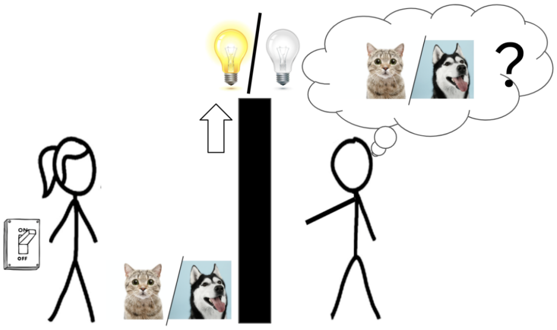


Figure 13: **An illustration of the Cat & Dog game taken from Hu et al. (2021a).**

Collision rules restrict movement into narrow tiles. Specifically, (i) if two agents attempt to swap positions within a single turn and at least one of them occupies a narrow tile, the swap is blocked, and no movement occurs; (ii) if both agents attempt to enter the same narrow tile simultaneously, neither move is executed. Given these restrictions, at narrow points in the tunnel, agents must assume asymmetric roles to successfully traverse it (leader and follower).

The environment is designed to support any custom layout via a simple hyperparameter at initialisation. The default layout, shown below, mimics the coordination challenge illustrated in the previously shown video:[2]

$$[\texttt{start\_left}, \texttt{wide}, \texttt{narrow}, \texttt{wide}, \texttt{start\_right}]$$

Brittle Bridge is inspired by cooperative platformer video games in which successful progression requires players to adopt complementary roles, such as one agent acting while the other temporarily waits. Although the assignment of such leader–follower roles is arbitrary, it is essential for successful task completion. Two agents are initially located on the same side of a bridge of length n. Their joint objective is to traverse the bridge such that both agents reach the opposite side in the minimal number of turns. The environment is turn-based, with agents selecting actions simultaneously from the set $\{\texttt{forward}, \texttt{backward}, \texttt{stay}\}$. No collision constraints are imposed, so multiple agents may occupy the same tile at any given time. The coordination challenge stems from the bridge's fragility. If both agents remain on the bridge, regardless of their exact positions, for more than a fixed number of consecutive turns, given by the parameter `bridge_break_threshold` (typically set to 1), the bridge collapses. When this happens, the episode ends and both agents receive a large negative reward of -1000. Importantly, the timer associated with the fragility constraint resets whenever at least one agent steps off the bridge. Consequently, when `bridge_break_threshold` equals 1, one agent may safely cross the bridge in its entirety while the other agent alternates between entering the first bridge tile and retreating to land, thereby preventing collapse. For our experiments, we fix n to 5 and the `bridge_break_threshold` to 1.

C.2 Convention games

Convention games typically assign players fixed asymmetric roles and proceed sequentially. The coordination challenge arises from partial observability: the first player has access to some information that must be conveyed to the second player, who then makes a guess. To achieve a high reward, the players must establish a convention that maps possible signals to possible guesses.

Cat & Dog is a two-player, fully cooperative coordination game introduced by (Hu et al., 2021a). The setup is straightforward: two players, *Alice* and *Bob*, are initially separated by a wall. At the beginning of the game, Alice privately observes a randomly chosen animal, either a cat or a dog, on her side of the wall. She must then select one of the following actions:

[2] `start_left` and `start_right` are considered wide tiles.

- Turn on a light bulb on Bob's side of the wall at no cost,
- Remove the wall at a cost of -5,
- Bail, yielding a joint reward of 1, terminating the game immediately.

Subsequently, it is Bob's turn. He observes the state of the light bulb and, if Alice has removed the wall, also observes the animal. Bob then chooses one of the following actions:

- Guess the correct animal (*Cat* or *Dog*). If his guess is correct, both players receive a joint reward of 10; otherwise, they incur a joint penalty of -10. In expectation, a random guess yields a reward of 0.
- Bail, yielding a smaller but guaranteed reward of 0.5.

The term *Cat & Dog* refers specifically to the one-shot version of the game. The iterated version is examined in the following section. Since there are two equivalent but arbitrary signalling conventions (either `light_on` → *Cat* or `light_on` → *Dog*), independent agents without prior communication cannot coordinate on a convention within a single round. Therefore, removing the wall is considered "optimal" in the ZSC setting.

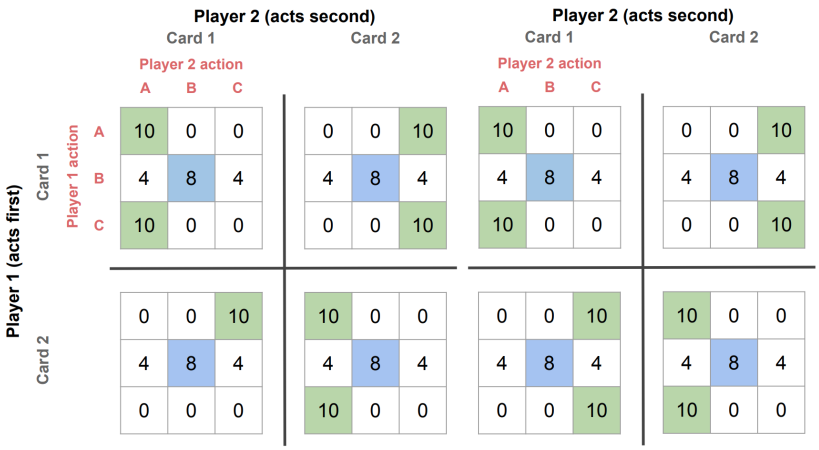
Ad-Hoc Conventions. Inspired by Gessler et al. (2025), we introduce games that require protocol formation at test time. *Ad-Hoc Conventions* is an iterated version of Cat & Dog. In contrast to the one-shot version discussed in the previous section, iteration enables the formation of conventions that can lead to guaranteed high payoffs in later rounds. We replace the animals with generic levers that can be assigned labels (in our case $\{\texttt{blue}, \texttt{red}\}$) and the lightbulb is replaced by labelled signals (in our case $\{\texttt{square}, \texttt{circle}\}$). Removing the wall is now represented by revealing the current correct lever. While we do not make further use of it in our experiments, we support an arbitrary number of signals and levers to increase the difficulty of the game. Additionally, the lever labels, signal labels, and the number of rounds can be specified as a hyperparameter. In our experiments, we play ten rounds per Ad-Hoc Conventions game.

Ad-Hoc Conventions with permuted signals. We also introduce a harder variant of Ad-Hoc Conventions, called *Ad-Hoc Conventions with permuted signals*. Here we distinguish between *buttons* and *signals*, each with its own set of labels ($\texttt{apple}, \texttt{banana}$ and $\texttt{square}, \texttt{circle}$ in our case). The only change per round is that, instead of directly revealing a signal, Alice can now press one of the buttons (or choose reveal or bail), which then displays a signal to Bob. The connection between buttons and signals is given by a random bijection unknown to both players. Importantly, Alice does not see the resulting signal, and Bob does not see the pressed button. Without prior communication, it is therefore impossible to recover the exact bijection between signals and levers. As a result, the game cannot be solved through a simple lexicographical order strategy: Alice cannot know which button will trigger which signal for Bob. Nevertheless, in both the permuted and non-permuted versions, the players retain a simple fallback strategy—continuously revealing (i.e. removing the wall)—which guarantees a reward of 5 per round, for a total "safe" return of 50 across the 10 rounds we typically consider in our experiments.

Tiny Hanabi is a toy-game introduced in (Foerster et al., 2019). It consists of two players, who are each dealt a random card at the start of the game. The final joint reward depends on both the players' cards and their actions throughout the game. The second player observes the first player's actions before picking his action, which allows the first player's action to be interpreted as a signal. We show the payoff matrix of Tiny Hanabi in Figure 14.

Due to the asymmetry of the payoff matrix, the original version of Tiny Hanabi has a unique SP optimal policy, yielding a joint reward of 10. This policy is given by the first player picking action A if he has card 2 and C otherwise, with the second player responding accordingly.

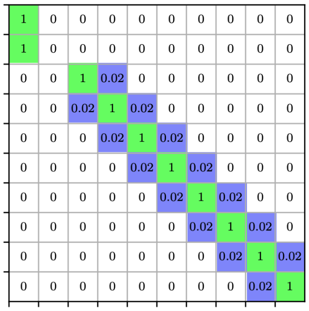
To create a version of Tiny Hanabi with a coordination challenge, we symmetrise the payoff matrix, also shown in Figure 14. As in the previous convention games, there is a safe fallback option, which does not require any conventions: Playing B as the first player leads to a reward of 8 as long as the second player responds rationally.


Player 2 (acts second)
Card 1
Card 2
Player 2 action
A
B
C
Player 1 (acts first)
Card 1
Card 2
Player 1 action
A
B
C
Player 2 (acts second)
Card 1
Card 2
Player 2 action
A
B
C

Player 1 (acts first)	Player 1 action	Card 1 A	Card 1 B	Card 1 C	Card 2		
Card 1	A	10	0	0	0	0	10
	B	4	8	4	4	8	4
	C	10	0	0	0	0	10
Card 2		0	0	10	10	0	0
		4	8	4	4	8	4
		0	0	0	10	0	0

	Card 1 A	Card 1 B	Card 1 C	Card 2		
	10	0	0	0	0	10
	4	8	4	4	8	4
	10	0	0	0	0	10
	0	0	10	10	0	0
	4	8	4	4	8	4
	0	0	10	10	0	0

Figure 14: **Payoff matrices for Tiny Hanabi adapted from Foerster et al. (2019).** On the left side, we show the matrix from the original paper, and on the right side, the matrix of our version with a coordination challenge.



1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	1	0.02	0	0	0	0	0	0
0	0	0.02	1	0.02	0	0	0	0	0
0	0	0	0.02	1	0.02	0	0	0	0
0	0	0	0	0.02	1	0.02	0	0	0
0	0	0	0	0	0.02	1	0.02	0	0
0	0	0	0	0	0	0.02	1	0.02	0
0	0	0	0	0	0	0	0.02	1	0.02
0	0	0	0	0	0	0	0	0.02	1

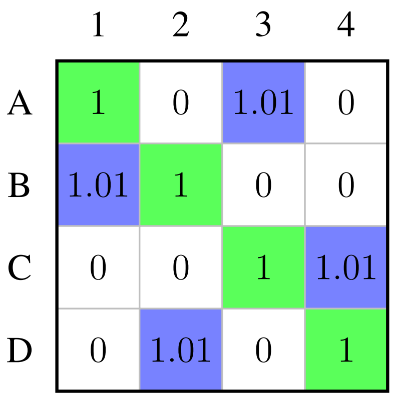
Figure 15: **Payoff matrices for the Matrix Game** adapted from Lupu et al. (2021). Players simultaneously pick a row and column, respectively, and receive the reward at the selected cell.

C.3 Simultaneous selection games

In *simultaneous selection games*, symmetric players have to coordinate on one of many similar, often symmetric, choices to achieve a high joint reward. We will study both one-shot versions of these games (*Matrix Game*) and iterated versions (*Permuted Levers*).

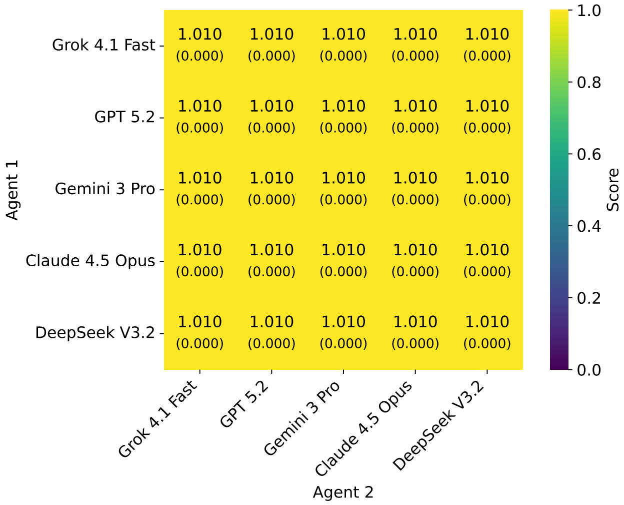
Matrix game. Our version of the *Matrix Game* was introduced by Lupu et al. (2021). In this game, two players simultaneously choose a row/column and receive the reward in the selected cell. The matrix used to play this game is shown in Figure 15. For humans, there is an easy-to-identify focal point given by the first column and first/second row, which can be exploited. Our version of the matrix game uses a 11×11 matrix extending the diagonal pattern by 1. This yields another possible focal point at the middle of the diagonal pattern (see also subsection B.8).

Permuted levers Lever Games are common in ZSC literature (Hu et al., 2021b; Treutlein et al., 2023). For our purposes, we focus on iterated versions of lever games with n levers all with identical payoff. In each round, both players simultaneously chose one of the levers. If players manage to pick the same lever, they receive a joint reward of 1. To prevent arbitrary focal points introduced by numbering the levers (which is needed for the LLM agents to interact with the environment), we introduce a further complication inspired by the other-play objective (Hu et al., 2021a): Each player observes and acts on the levers through a random permutation of the lever numbers, which stays fixed across the rounds of the game. While players are aware of the fact, they do not know their own or others' permutations.



	1	2	3	4
A	🟩 1	0	🟦 1.01	0
B	🟦 1.01	🟩 1	0	0
C	0	0	🟩 1	🟦 1.01
D	0	🟦 1.01	0	🟩 1

Figure 16: **Semantic Convention Game.** Alice and Bob chose a row and column in the matrix sequentially. Additionally, only Alice is informed of the currently active column, which is chosen at random. Only if the agents end up selecting a cell in the active column do they receive the reward in that cell. Agents must choose between the lexicographical mapping convention (green) and the SP-optimal mapping convention (blue).



Agent 1 \ Agent 2	Grok 4.1 Fast	GPT 5.2	Gemini 3 Pro	Claude 4.5 Opus	DeepSeek V3.2
Grok 4.1 Fast	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)
GPT 5.2	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)
Gemini 3 Pro	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)
Claude 4.5 Opus	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)
DeepSeek V3.2	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)	1.010 (0.000)

Figure 17: **XP scores of code agents in the semantic selection game using the initial CFD candidate.** Scores are averaged across six different XP matrices; each cell reports the mean return over 100 episodes.

Both the number of levers k and the number of rounds n are hyperparameters, which can be set at initialisation. For our experiments, we fix $k = 5$ and $n = 10$.

C.4 Semantic games

Semantic games demonstrate that the addition of semantics can create coordination challenges in games that previously did not exhibit them, as they have a unique SP optimal policy. In particular, the semantic games presented in this paper *semantic selection game* (**??**) and *semantic convention game* (Figure 16) force players to choose between two mutually incompatible meta-strategies: either playing the SP optimal strategy or following a strategy that is aligned with the game semantics. section 4 has demonstrated that LLM agents often struggle to consistently interpret semantics and fail to properly reason about these difficulties. Semantic games, on the other hand, show that even when the semantic interpretation is clear across agents, semantics can still hinder the coordination abilities of agents, just by the fact that they introduce another possible meta-strategy.

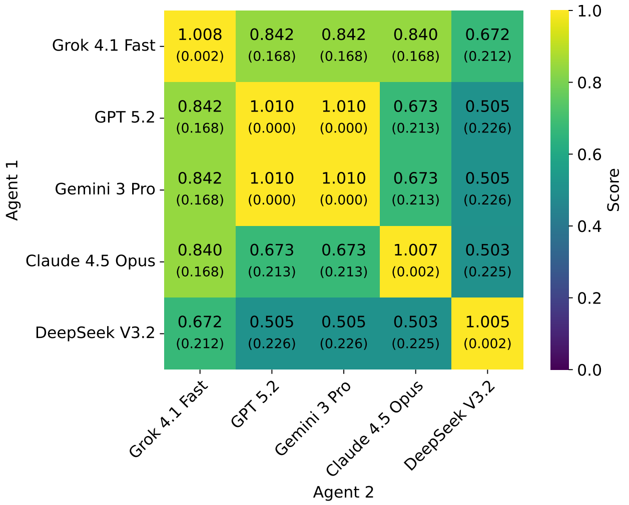


Figure 18: **XP scores of code agents in the semantic convention game using the initial CFD candidate.** Scores are averaged across six different XP matrices; each cell reports the mean return over 100 episodes.

?? demonstrates that LLM agents are generally not able to consistently choose among these competing meta-strategies, leading to coordination failures. Interestingly, even though the CFD does specify how to decide among the competing meta-strategies, it does lead agents to consistently opt for the SP optimal policy in the semantic selection game. In the semantic convention game, the CFD does not yield significant improvement in the agents' coordination as shown in Figure 18.

D PROMPTS

D.1 TEXT AGENT

The prompt template used to instruct text agents is shown in in Figure 20.

Prompt Template for Text Agents

You are playing a cooperative game with another player. The game is generally turn-based. While you cannot communicate with the other player, you can observe their actions and the joint rewards from previous turns. Your objective is to maximize the joint reward.

You will play only a single round of the game (a round may consist of multiple turns).

Below is a detailed description of the game and, if applicable, the history of previous turns.

```
$Game Description$

$Game History$
```

You must now select an action for the current turn from the following set of legal actions:

```
$Legal Actions$

$CFD Instructions (if applicable)$
```

Respond using the following format:

~~~
REASONING:
<REASONING>

ACTION JSON:

```json
<JSON>
```
~~~

Figure 19: Template of the prompt used for text agents.

D.2 CODE AGENT

The prompt template used to instruct code agents to generate a joint policy is shown in Figure 20.

Prompt Template for Code Agents

General Instructions:

You will need to generate policies for a collaborative two-player game.

What is special about the game:

The game is generally turn-based. Agents cannot communicate with each other. Agents can see their actions and the (joint) reward from previous turns. Generally, agents play each game only once (i.e. they can't remember previous games). If the game is iterated (i.e. the game is played for multiple rounds), this will be made clear in the game description. Note that each game consists of multiple turns. Ensure the policy is coordination-friendly.

Instructions:

Policy Requirements:

Encode each policy as concise, readable Python code. Ensure the policies have good coordination performance in the ZSC setting.

Zero-shot Coordination (ZSC) Setting:

The policy will be evaluated against independently generated policies (potentially from different LLMs) but with similar instructions (game description, etc.). Poor coordination performance often arises when policies adopt incompatible conventions or break symmetry arbitrarily.

Ultimate goal:

Maximize coordination in the ZSC setting.

`$CFD Instructions (if applicable)$`

What exactly is a policy?

Each of the two players has a (local) policy which is an action function. Such a function takes in an observation_dict and returns a legal action. Generally, each policy function must have the following signature: `def act(observation_dict: dict, memory: dict) -> tuple`.

The first return value is the action (string), the second return value is the memory (dictionary). The memory is a dictionary that can be used to store information between calls to the act function. The memory is initialized to an empty dictionary (i.e. {}) at the start of the game. It is NOT initialized to None. The memory is passed to the act function in each call. Players do not share the memory, each player has their own memory. You can make use of python standard libraries like random and math (but not time).

How does the observation_dict look like? (An example; you can always expect the same general structure)

NOTE: Only index for values using the EXACT keys (1:1 match) from the example below. Otherwise, you will not be able to access the values.

`$Sample Observation Dict$`

What are the legal actions?

Note that actions are encoded as strings. The returned action needs to match exactly one of the legal actions (e.g. we are case-sensitive). If an invalid action is returned the game will end immediately and both players receive a penalty.

`$Legal Actions Dict$`

Note: Each local policy should always return a valid action, even when paired with a different local policy (i.e. the actions of the other player might be different). Policies that produce invalid actions will be discarded.

Game Description:

`$Game Description$`

Output Specification:

Propose 1 set of policies for each player; put each set under the list `"joint_policies"`. The policies will be executed in Python to generate actions given the observation_dict. Make sure that you are generating valid Python code or else the game ends immediately and you receive a penalty. There are 2 players in this game. While we use player_0, player_1, etc. to refer to the players, they are only used for constructing the JSON, they are not part of the game.

The `joint_policy` should be formatted as JSON below. MAKE SURE TO FOLLOW THE FORMAT EXACTLY, as the JSON is parsed automatically.

```
{ "joint_policies": [ { "name": The name of the joint policy, same for all
players using this policy. "reason": The reason why it is designed in this
way. player_0: { "code": The python code for the policy function. Should
have the following signature: def act(observation_dict: dict, memory: dict)
-> Tuple[str, dict]. Make sure to escape double quotes within the code. },
player_1: { ... }, ... }, ... ] }
```

Output

~~~
REASONING:
<REASONING>

Policy JSON:

```json
<JSON>
```
~~~

Figure 20: Template of the prompt used for code agents.