TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Saad, F., Murukannaiah, P. K., & Singh, M. P. (2026). Theory of mind in action: The instruction inference task in dynamic human-agent collaboration. *Artificial Intelligence*, *361*, Article 104621. https://doi.org/10.1016/j.artint.2026.104621

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

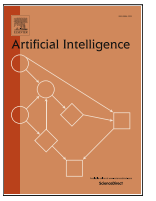
Artificial Intelligence 361 (2026) 104621



Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Theory of mind in action: The instruction inference task in dynamic human-agent collaboration

Fardin Saad [a,*], Pradeep K. Murukannaiah [b], Munindar P. Singh [a]

[a] North Carolina State University, Raleigh, NC, USA
[b] Delft University of Technology, Delft, the Netherlands

ARTICLE INFO

Keywords:
Theory of mind
Human-agent collaboration
Large language models
Instruction interpretation
Ambiguity

ABSTRACT

Successful human-agent teaming relies on an *agent* being able to understand instructions given by a (human) *principal*. In many cases, an instruction may be incomplete or ambiguous. In such cases, the agent must infer the unspoken intentions from their shared context, that is, it must exercise the principal's Theory of Mind (ToM) and infer the mental states of its principal. We consider the prospects of effective human-agent collaboration using large language models (LLMs). To assess ToM in a dynamic, goal-oriented, and collaborative environment, we introduce a novel task, *Instruction Inference*, in which an *agent* assists a *principal* in reaching a goal by interpreting incomplete or ambiguous instructions.

We present *Tomcat*, an LLM-based agent, designed to exhibit ToM reasoning in interpreting and responding to the principal's instructions. We implemented two variants of Tomcat. One, dubbed Fs-CoT (Fs for few-shot, CoT for chain-of-thought), is based on a small number of examples demonstrating the requisite structured reasoning. One, dubbed CP (commonsense prompt), relies on commonsense knowledge and information about the problem. We realized both variants of Tomcat on three leading LLMs, namely, GPT-4o, DeepSeek-R1, and Gemma-3-27B. To evaluate the effectiveness of Tomcat, we conducted a study with 52 human participants in which we provided participants with the same information as the CP variant. We computed intent accuracy, action optimality, and planning optimality to measure the ToM capabilities of Tomcat and our study participants. We found that Tomcat with Fs-CoT, particularly with GPT-4o and DeepSeek-R1, achieves performance comparable to the human participants, underscoring its ToM potential for human-agent collaboration.

Further, we measure robustness to game-specific patterns by introducing four environmental perturbations and find that performance remains unchanged. We evaluate the generalizability of Instruction Inference in a second collaborative environment, Overcooked, using four additional LLMs, namely, Claude Opus 5, GPT-5.6 Sol, Gemini 3.1 Pro, and DeepSeek V4 Pro. We analyze these results in terms of the LLMs' post-training paradigms and observe that six of the seven models behave as predicted.

1. Introduction

Natural communication is rarely limited to literal language [1,2]. Instead, collaborators frequently use indirect expressions whose intended meaning must be inferred from contextual cues and shared task understanding [3–5]. Successful collaboration therefore presupposes an ability to interpret incomplete and ambiguous language, which in turn presupposes an understanding of the

* Corresponding author.
E-mail address: fsaad@ncsu.edu (F. Saad).

https://doi.org/10.1016/j.artint.2026.104621
Received 24 June 2025; Received in revised form 30 August 2026; Accepted 2 September 2026
Available online 6 September 2026
0004-3702/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

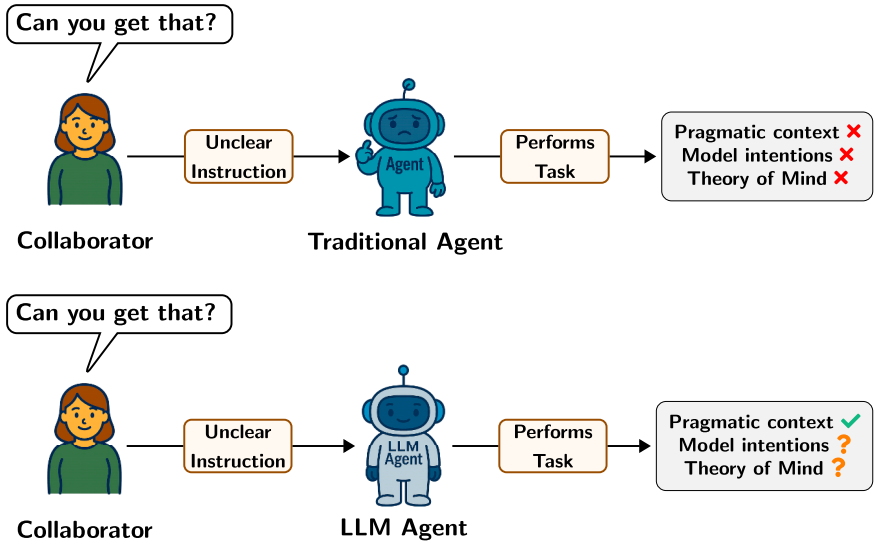


Fig. 1. Interpreting unclear instructions by traditional vs. LLM agents where the latter performs the task considering the pragmatic context.

collaborator's mental state [6,7]. The capability of modeling another entity as endowed with a *mind*, e.g., beliefs and intentions, is referred to as having a Theory of Mind (ToM) [8,9]. ToM enables people to infer unspoken intentions and respond appropriately to indirect cues [10].

For instance, a collaborator may say *Can you get that?* while pointing to an object (e.g., a key) in the shared environment. Interpreting such an instruction requires inferring which object was requested based on contextual cues and shared task understanding, thereby requiring ToM reasoning. At the first-order ToM level, this may involve reasoning such as *I believe that you want the red key*. In more complex situations, collaborators may engage in higher-order reasoning, i.e., second-order ToM, where one reasons about another party's beliefs, for instance, *I believe that you believe that I know which key is needed*. Such higher-order reasoning is often observed in strategic interactions involving persuasion, negotiation, or deception [11–13]. However, in cooperative settings, resolving unclear instructions often relies on reasoning about the collaborator's immediate goals and intentions, which can be captured through first-order ToM [4,6,14].

Equipping AI agents with ToM can foster more natural and effective human-agent interaction [15], particularly in settings where communication is incomplete or ambiguous [4]. However, traditional autonomous agents have historically struggled to interpret such language. Prior work in instruction following and human-agent collaboration shows that agents lacking pragmatic context [16–18] tend to interpret instructions literally and thus fail when instructions are incomplete or ambiguous [3,5]. As a result, these agents often cannot infer a collaborator's intended goal from contextual cues in dynamic task environments, as shown in Fig. 1.

The advent of large language models (LLMs) offers a promising avenue to bridge this gap. LLMs have demonstrated an emergent capability to account for pragmatic context and perform sophisticated linguistic reasoning [19–22]. Recent studies suggest that these models may exhibit ToM capabilities by modeling the mental states of others [20,23]. However, evaluations of the ToM capabilities of LLMs rely on static benchmarks such as false-belief, faux-pas, or conversational reasoning tasks. For example, Strachan et al. [19] measure the ToM capabilities of LLMs by providing a short conversational vignette followed by targeted questions to assess how LLMs reason about another party's mental states. These settings evaluate reasoning about mental states but do not test whether agents can use such reasoning to interpret *unclear instructions* and act appropriately in *collaborative task environments*, as shown by the contrast in Fig. 1.

To address this challenge, we introduce a new ToM task, ***Instruction Inference***, consisting of 20 distinct problem scenarios. Each scenario evaluates an agent's capability to infer a principal's (simulated human) intended goal from clear and unclear instructions and act appropriately in a collaborative environment. We use the term *instructions* to refer to commands, requests, or directives issued by the principal. To ground the design of our dataset, we draw on Grice's theory of conversational implicature [1]. Grice identifies that cooperative communication is guided by four conversational maxims: Quantity, Quality, Relation, and Manner. Violations of these maxims trigger pragmatic inference, requiring listeners to infer the speaker's intended meaning beyond the literal utterance. We distinguish two types of unclear instructions grounded in this framework. *Incomplete* instructions omit critical information required to determine the intended action; they correspond to violations of Quantity maxim. *Ambiguous* instructions yield two or more possible interpretations; they correspond to violations of Manner maxim. We focus on these two categories because violations of Quantity and Manner maxims commonly arise in cooperative settings where collaborators assume shared context and task understanding [3–5]. By contrast, violations of the Quality and Relation maxims are less characteristic of cooperative collaboration, as they typically correspond to deception or conversational breakdown.

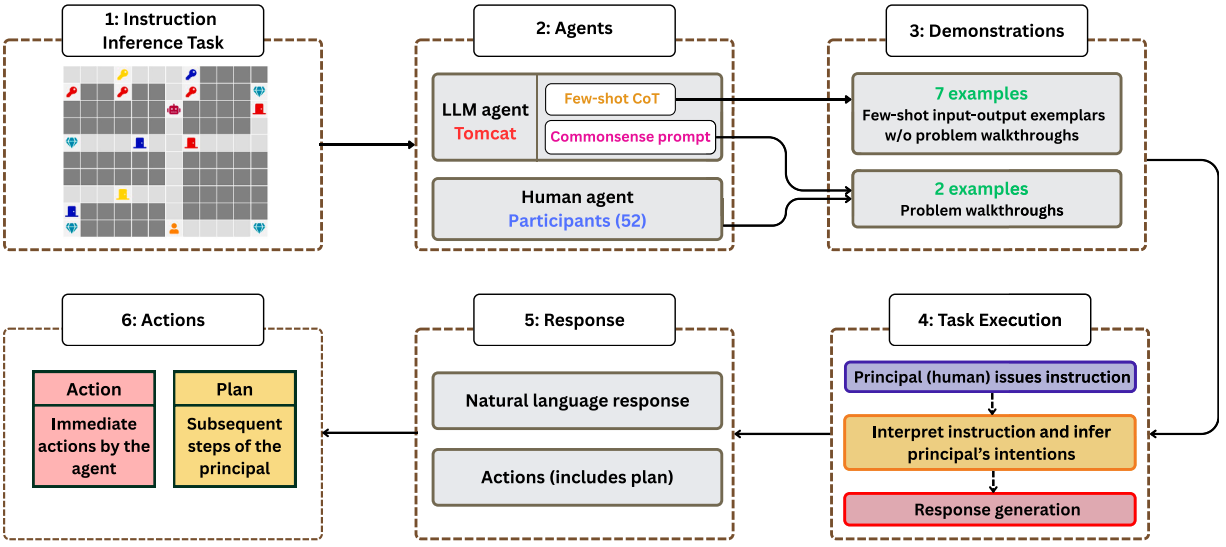


Fig. 2. Pipeline of the Instruction Inference task with Tomcat and participants.

We implement this task using *Tomcat*, an LLM-based agent deployed in a collaborative environment called Doors, Keys, and Gems [3], shown in Fig. 3. In this environment, Tomcat assists its principal in achieving their objective of collecting a gem by retrieving keys or unlocking doors. Tomcat interprets both clear and unclear instructions by reasoning about the task environment and the principal's movements. Our study employs two variants of Tomcat, *Few-shot Chain-of-thought (Fs-CoT) Prompt* and *Commonsense Prompt (CP)* (Fig. 2). Tomcat with Fs-CoT leverages in-context learning [24] and few-shot chain-of-thought (Fs-CoT) prompting [25] to incorporate task knowledge and contextual cues to infer the principal's intended goal. Tomcat with CP relies on commonsense knowledge about the Instruction Inference task with two detailed problem walkthroughs.

We realized both variants of Tomcat on three LLMs—GPT-4o, DeepSeek-R1, and Gemma-3-27B—and compare their ToM capabilities with human participants. To establish a benchmark for comparison, we conduct an empirical study with 52 human participants who perform the same Instruction Inference task under identical conditions as Tomcat with CP, i.e., using the same information and objectives. This evaluation is framed by the following research questions.

RQ$_1$: How do Tomcat's ToM capabilities on Instruction Inference compare to human performance?

RQ$_2$: Does Tomcat with Fs-CoT outperform Tomcat with CP on Instruction Inference?

Beyond this evaluation, we examine whether our findings depend on the specific environment and LLMs. We perturb the Doors, Keys, and Gems environment, reimplement Instruction Inference in Overcooked, and interpret the outcomes using the evaluated LLMs' post-training paradigms.

To summarize, our **contributions** are as follows:

1. **Instruction Inference task.** We introduce Instruction Inference, a novel task for evaluating ToM in a collaborative environment where instructions may be clear or unclear (incomplete or ambiguous). Unlike existing ToM benchmarks that rely primarily on narrative or conversational scenarios, Instruction Inference evaluates whether an agent can infer a principal's intended goal from clear and unclear instructions and act appropriately in a shared task environment.
2. **Tomcat agent for instruction inference.** We develop *Tomcat*, an LLM-based agent that performs instruction inference in dynamic environments by reasoning about pragmatic context and the principal's actions. We investigate two variants of Tomcat: a *Few-shot Chain-of-Thought (Fs-CoT)* variant using seven reasoning exemplars, and a *Commonsense Prompt (CP)* variant that relies on task knowledge with two problem walkthroughs.
3. **Empirical evaluation with human participants.** We conduct an empirical evaluation comparing both Tomcat variants' ToM capabilities, realized on three LLMs (GPT-4o, DeepSeek-R1, and Gemma-3-27B), with those of 52 human participants on our Instruction Inference task. Our results show that the Fs-CoT variants of GPT-4o and DeepSeek-R1 achieve performance comparable to human participants.
4. **Robustness, generalizability, and post-training analysis.** We perturb the Doors, Keys, and Gems environment in four ways: three preserve the optimal plan, and one alters it by design. We show that the aggregate performance remains unchanged, indicating robustness to game-specific patterns. To test generalizability, we implement the task in Overcooked with four additional LLMs, and attribute the varying benefits of Fs-CoT to post-training paradigms that predict the behavior of six of the seven models.

The subsequent sections provide a systematic analysis of each module in our pipeline, as shown in Fig. 2.

2. Related work

ToM Benchmarks. Building on the recent success in LLMs, Strachan et al. [19] evaluate models like GPT-4, GPT-3.5, and Llama2-70B on ToM tasks including false-beliefs, faux pas, irony, strange stories, and hinting tasks. The tasks adopt practical conversational social scenarios in contrast to our goal-driven collaborative setting. Strachan et al. [19] report that GPT-4 outperforms humans in irony, hinting, and strange stories, but not in faux pas. Similarly, Kosinski [20] tests foundational ToM concepts, such as belief attribution and reasoning about mental states, using structured cognitive frameworks, revealing that advanced LLMs like GPT-4 exhibit human-like reasoning in certain tasks but not in nuanced scenarios. Kim et al. [26] introduce *FANToM*, a conversational benchmark that evaluates mental state reasoning under asymmetric information. Chan et al. [27] present *NegotiationToM*, a benchmark grounded in the belief-desire-intention (BDI) framework that evaluates ToM reasoning in negotiation dialogues derived from the CaSiNo dataset [28]. Gandhi et al. [21] introduce BigToM, a large scale benchmark that extends classical narrative-based ToM tasks to evaluate belief reasoning across diverse social scenarios.

Higher-order ToM Reasoning. Street [29] posits that LLMs' ToM capabilities can help improve human-agent collaboration. Street et al. [30] develop the Multi-Order Theory of Mind Q&A (MoToMQA) test suite to evaluate LLMs on ToM tasks that require recursive reasoning about nested mental states (second- to sixth-order). They show GPT-4's capability to handle the deep recursive reasoning that is critical for understanding complex social interactions. Jamali et al. [31] analyze hidden embeddings in LLMs and identify neuron-like units responsive to true- and false-belief tasks, drawing parallels to ToM processing by humans.

ToM in Multiagent Systems. Recent work has expanded ToM into more complex domains, such as multiagent systems, automated planning, and multimodal environments. Rocha et al. [32] review the use of ToM in multiagent systems, noting its value for improving coordination, generating explanations, providing assistance, and modeling deception. In planning, Shvo et al. [33] use ToM to help agents identify conflicting beliefs about a plan's validity and correct these misunderstandings through direct communication or physical actions. Shi et al. [34] introduce MuMA-ToM, a multimodal multiagent benchmark for inferring an agent's beliefs and social goals in everyday household activities using video and textual context. Yim et al. [35] adopt the cooperative card game *Guandan* and show that ToM-based planning helps LLM agents better coordinate with allies and adapt strategies against opponents.

ToM in Multiagent Reinforcement Learning. Oguntola et al. [43] explore ToM as an intrinsic reward signal for multiagent reinforcement learning (MARL) through second-order belief prediction and find success in improving multiagent collaboration in mixed cooperative-competitive environments. Li et al. [41] compare LLMs' ToM capabilities against MARL and planning in multiagent text-based games. They demonstrate emergent collaborative behaviors and higher-order ToM reasoning in LLMs and highlight challenges due to long-horizon planning and belief tracking.

Limitations of LLMs for ToM. Zhang et al. [42] find that ToM capabilities improve understanding, but do not enhance overall team performance in a real-time shared workspace task. Amirizaniani et al. [40] using open-ended questions derived from Reddit's *ChangeMyView* subreddit, note that prompt-tuning of GPT-3.5 and GPT-4 improves performance but that LLMs do not achieve human-like reasoning in nuanced social contexts. Sap et al. [37] evaluate GPT-3's ToM capabilities using the SocialIQa and ToMi (Theory of Mind Inventory) tasks, highlighting performance significantly lower than humans and emphasizing the limitations of LLMs in social intelligence. Extending this line of inquiry, Verma et al. [39] examine GPT-3.5 and GPT-4 in Human-Robot Interaction tasks. Performance declines significantly under minor contextual perturbations, suggesting that LLMs rely on superficial reasoning rather than robust ToM capabilities. Similarly, Ullman [38] demonstrates that GPT-3 and GPT-3.5 struggle with false-belief tasks when trivial alterations, such as rephrasing or irrelevant context, are introduced, indicating their dependence on surface-level patterns. However, Strachan et al. [19] report that humans *also* fail perturbed false-belief tasks, possibly due to their tendency to maintain original beliefs despite contradictory evidence.

Instruction Inference (Our Task). Table 1 contrasts these benchmarks across key dimensions and shows that our Instruction Inference task uniquely combines human-agent, goal-driven collaboration in a changing environment with inherently incomplete and ambiguous instructions.

3. Instruction inference

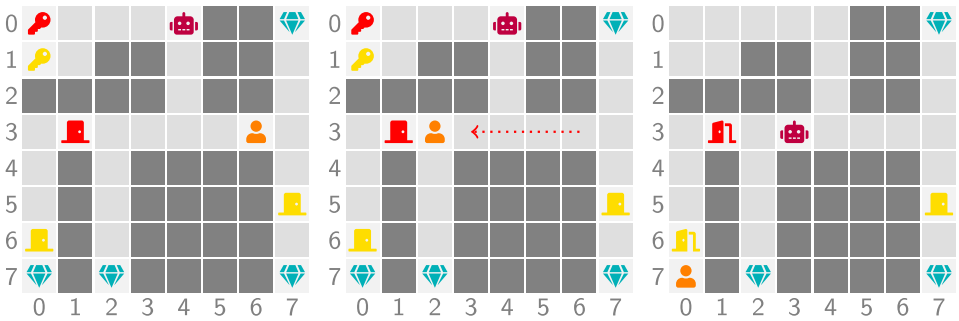
This section describes the task setting and terminology used in our study, the Doors, Keys, and Gems environment, the design rationale behind the instruction types used in our dataset, and the pipeline through which agents interpret instructions and generate actions.

3.1. Task setting and terminology

The Instruction Inference task helps assess an agent's ToM capabilities in a dynamic, goal-oriented environment involving human-agent collaboration. The agent executes actions by reasoning about the principal's mental model based on their instructions [44]. The task involves two roles: a *principal* who issues instructions and an *agent* that interprets the instructions and performs actions. In the

Table 1
Comparison of ToM benchmarks across format, modality, collaboration setting, and instruction type.

ToM Task	Format	Modality	Collaborative	Instruction	Scenario
False-Belief [19,20,31]	Narrative QA	Text	No	–	Social vignette
Faux Pas, Irony, Hinting [19]	Narrative QA	Text	No	–	Social vignette
RMET (Reading the Mind in Eye Test) [36]	Visual QA	Visual + Text	No	–	Visual vignette
SocialIQa, ToMi [37]	Multiple choice QA	Text	No	–	Social vignette
Altered ToM [38]	Narrative QA	Text	No	–	Social vignette
Robot Intent Classification [39]	Nested QA	Text	No	–	Robot behavior
ChangeMyView [40]	Open-ended QA	Text	No	–	Social vignette
MoToMQA (Multi-Order ToM Q&A) [30]	Nested QA	Text	No	–	Social vignette
FANToM [26]	Conversational QA	Text	No	–	Dialogue exchange
NegotiationToM [27]	Multiple choice QA	Text	No	–	Negotiation dialogue
BigToM [21]	Narrative QA	Text	No	–	Social vignette
Hi-ToM [22]	Nested QA	Text	No	–	Social vignette
ToM Text Games [41]	Text based game	Text	Yes (Agent–agent)	Explicit	Cooperative gameplay
MuMA-ToM [34]	Multimodal QA	Video + Text	Yes (Agent–agent)	Explicit	Household interaction
Guandan ToM [35]	Card game	Text	Yes (Agent–agent)	Explicit	Strategic gameplay
Shared Workspace Task [42]	Action execution	Text	Yes (Human–agent)	Explicit	Collaborative gameplay
Code-ToM Assistant [15]	Interactive assistant	Text	Yes (Human–agent)	Explicit	Code comprehension
Instruction Inference **(Ours)**	Plan & action execution	Text + Spatial	Yes (Human–agent)	Implicit	Uncertain collaboration



(a) Initial state of our scenario. Dark gray cells are walls and light gray cells are passages; the icons are a principal, an agent, doors, keys, and gems.

(b) The principal moves toward the red door and asks for the red key. Tomcat figures the principal wants the gem at (7, 0) and will also need the yellow key.

(c) Tomcat fetches the red (0, 0) *and* yellow (1, 0) keys to the principal, so they can unlock the red (3, 1) and yellow (6, 0) doors to reach the gem at (7, 0).

Fig. 3. Example scenario in the Doors, Keys, and Gems world, showing an agent inferring a (human) principal's mental state.

environment, the principal represents a simulated human collaborator whose movements and instructions provide contextual cues about their intended goal.

In our empirical evaluation, human participants assume the role of the *agent*. Thus, participants must interpret the principal's instructions and perform actions accordingly in the environment. Since the study involves two distinct human roles, we explicitly distinguish between them:

(i) **Human as Principal:** The simulated human principal that appears in the environment (prompts) and issues instructions.
(ii) **Human as Participant (Agent):** A human study participant who assumes the role of the agent during evaluation.

This distinction clarifies the use of the term "human" in the prompts and environment descriptions, where it refers to the principal rather than the participants. The principal is simulated so that both human participants and LLM-based agents interact with the same instructions and environment under identical task conditions.

3.2. Doors, Keys, and Gems environment

Fig. 3 illustrates a representative scenario for Instruction Inference. Fig. 3a depicts an initial grid configuration of the Doors, Keys, and Gems environment. We express coordinates as (row, column), thus the principal is at (3, 6). Fig. 3b demonstrates the observed grid configuration after the principal moves from (3, 6) to (3, 2) and says "Can you get the red key?" to the agent. The agent observes the principal moving leftward and stopping adjacent to the red door. Based on this movement, the instruction, and the grid configuration, the agent infers that the principal's desired gem is likely situated beyond the red door at (7, 0). Since access to the gem is further blocked by a yellow door, the agent collects both the red and yellow keys and delivers them to the principal. The final grid layout is illustrated in Fig. 3c, where the principal unlocks both the red and yellow doors and retrieves their desired gem at (7, 0).

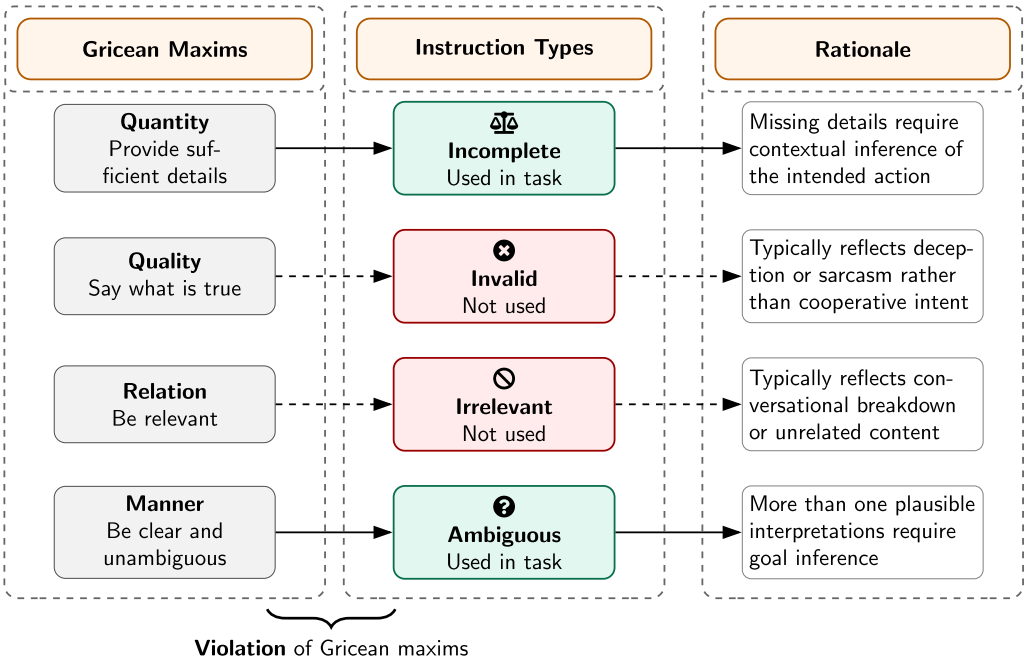


Fig. 4. Design rationale of unclear instruction types derived from violations of Gricean maxims. The Instruction Inference task categorizes *incomplete* (Quantity violation) and *ambiguous* (Manner violation) as unclear instructions.

Table 2
Instruction types in relation to violations of Gricean maxims and their distribution in the Instruction Inference task.

	Instruction Type	Gricean Maxim Violation	Used in Setup	# Instructions
Clear		None	Yes	8
Unclear	Incomplete	Quantity	Yes	6
	Ambiguous	Manner	Yes	6
	Invalid	Quality	No	–
	Irrelevant	Relation	No	–

3.3. Instruction types and design rationale

We ground the design of our instruction types in Grice's *Cooperative Principle* [1], which holds that effective communication requires participants to cooperate. Grice defined four conversational maxims under this principle: Quantity, Quality, Relation, and Manner, summarized in Fig. 4. When a maxim is violated, listeners use pragmatic inference (termed *implicature*) to infer the implied meaning from shared context [1].

Saad et al. [4] classify unclear instructions as violations of the Gricean maxims. Violations of Quantity, Quality, Relation, and Manner maxims yield incomplete, invalid, irrelevant, and ambiguous instructions, respectively. Incomplete instructions lack sufficient information needed to determine the intended action. Invalid instructions contain false information about actions unachievable within the task constraints. Irrelevant instructions have no connection to the task or to the speaker's goals. Ambiguous instructions admit more than one plausible interpretation. Clear instructions do not violate any maxim.

We use *incomplete* and *ambiguous* instructions among the four unclear categories because they are especially characteristic of cooperative collaboration, where speakers often assume shared context and therefore do not fully specify their intentions [3,4]. Conversely, invalid and irrelevant instructions are less representative of cooperative collaboration and would shift the task away from goal inference toward deception detection, sarcasm identification, or conversational breakdown.

Overall, our dataset contains three instruction categories: clear, incomplete, and ambiguous. The Instruction Inference task evaluates whether an agent can correctly interpret each category and infer the principal's intended goal. Table 2 reports the distribution of these categories in our dataset.

Response Generation

Based on your understanding of the game rules, current grid layout, human actions, instruction, and the labeled examples provided below, generate the instruction type (clear or unclear), response, and optimal actions for the following human actions and instruction, delimited by triple backticks:

```
‘‘‘
Human Action: The human moves to the right from their current position at (5,0) until they reach at (5,4) and gives the instruction.
Instruction: Can you pass me the blue key?
Type: <Identify the instruction type and provide chain-of-thought reasoning for the type based on the current grid configuration>
Response: <Generate chain-of-thought reasoning for the response based on the current grid configuration>
Actions: <Provide optimal actions from the response based on the instruction and the current grid configuration>
‘‘‘
```

Fig. 5. Excerpt of the response generation component.

3.4. Instruction inference pipeline

We show our pipeline in Fig. 2, which outlines the key components of the Instruction Inference task. In this setting, agents must interpret instructions in context, infer the principal's underlying intentions, and execute a sequence of optimal actions. The responses generated by the agents include a natural language response, an action plan, and an explanation of the reasoning behind each step. These responses collectively capture the agent's interpretation of the instruction and provide a structured basis for evaluating its ToM capabilities. By requiring agents to model latent goals and reason over incomplete or ambiguous instructions, the Instruction Inference task serves as a robust framework to assess goal-oriented ToM in a collaborative environment.

4. The Tomcat framework

Tomcat is an LLM-based agent, designed to collaborate with a principal on the Instruction Inference task. Tomcat leverages the common ground [45], including shared objectives and contextual interpretation of the environment. Further, Tomcat's decision making employs in-context learning and chain-of-thought reasoning to infer implicit instructions and execute the optimal actions. The prompt design comprises three key components:

Common Ground. It contains the background information, objectives, and rules of the collaborative game. It establishes common ground between Tomcat and the principal, including knowledge of the environment. The full prompt template is provided in Appendix A.1.

Response Generation. This incorporates the principal's actions (movements as displayed in the observed grid in Fig. 3b), the instruction, and the current grid (configuration). Using this information, Tomcat is tasked with determining the type of instruction, generating the natural language response, and the sequence of actions to achieve the shared objective. Fig. 5 shows an excerpt of the prompt (detailed prompt templates are available in Appendix A.2). The responses generated by Tomcat are as follows:

Type: Tomcat classifies the instruction as either *clear* or *unclear* by analyzing the principal's actions given the grid and the instruction, as shown in Fig. 5. Additionally, Tomcat provides a rationale for the classification, facilitating the inference of the principal's intent by identifying the desired gem.

For instance, *Type: Unclear. Based on the human's movement, instruction, and the current grid configuration, I infer that the human moved to (8,4) to collect the gem at (8,9), which is located beyond the blue door and red door at (8,6) and (8,8), respectively. Since the human asked for a blue key, it indicates they want to unlock the blue door first. However, the instruction does not mention the need for a red key to unlock the subsequent red door, making it unclear.*

(Natural Language) Response: Tomcat generates a natural language response detailing the actions it executes to assist the principal, along with the anticipated next steps of the principal. We refer to Tomcat's executed actions as *actions* and the complete response, which includes both Tomcat's action and the principal's subsequent steps, as a *plan*. This generated plan enables us to evaluate Tomcat's ToM capabilities.

For instance, *Response: The instruction asks to collect a blue key, and based on the human's movements and the grid configuration, the desired gem they want to collect is placed at (8,9). However, this gem is beyond both a blue door at (8,6) and a red door at (8,8). Therefore, I will collect one blue key and one red key located at (3,8) and (5,8), respectively. To minimize movement, I will first pick up the blue key at (3,8), followed by the red key at (5,8), as this strategy reduces the number of steps from my current position at (1,8) to the human at (8,4). Once both keys are collected, I will pass them to the human, allowing them to unlock the blue door and the red door at (8,6) and (8,8) respectively and retrieve the gem at (8,9).*

Demonstration Exemplars: CP

Use the following problems and examples, delimited by triple quotes, to understand how to generate the appropriate type, response, and actions for each instruction and human actions.
’’’
—**Problem (1)** demonstrating initial, observed, completed grids with figures and the example.

Figure 2: Initial Grid Configuration
…
–Initial Grid: This grid in Figure 2 …

Figure 3: Observed Grid Configuration
…
–Observed Grid: The human moves left to …

Figure 4: Completed Grid Configuration
…
–Completed Grid: You infer that the human …

```
Example from this grid configuration.
Human Action: The human moves left from their…
Instruction: Can you pass me the red key?
Type: Unclear. Based on the instruction and …
Response: There are two yellow doors, one …
Actions: 1) Collect red_key at (0,0)
…
```

Fig. 6. Excerpt of the demonstration exemplar component for Tomcat with CP.

Actions: Tomcat extracts the sequence of optimal actions from its response, generating them in an enumerated format. This sequence includes both Tomcat’s executed actions and the principal’s subsequent steps. Structuring responses this way facilitates efficient annotation and evaluation. For instance,

Plan
1. Collect: blue_key at (3,8). } **Action**
2. Collect: red_key at (5,8). } **Action**
3. Pass: blue_key and red_key to the human at (8,4).
4. Unlock: human unlocks the Blue_door at (8,6) and the Red_door at (8,8).
5. Retrieve: human retrieves gem at (8,9).

Demonstration Exemplars. These provide solved problem instances to illustrate correct reasoning and execution. The problems correspond to the grid scenarios depicted in Fig. 3a–c, whereas the demonstrations refer to the input-output pairs used by LLMs for in-context learning [24]. The input may include the grid, principal’s preceding actions, and the instruction, whereas the output includes the identification of the instruction type with the principal’s intent, the optimal response, and the corresponding sequence of actions. The input-output structure is defined as follows:

$$\text{Input} = \text{(Optional) Grid States} + \text{Principal's Actions} + \text{Instruction} \tag{1}$$

$$\text{Output} = \text{Type} + \text{Response} + \text{Actions} \tag{2}$$

The Tomcat framework has *two* variants: (1) the Commonsense Prompt (CP) and (2) the Few-shot Chain-of-Thought (Fs-CoT) prompt. These variants retain the three core components described earlier, but differ in the Demonstration Exemplar component with minor modifications in the common ground component. In CP (Fig. 6), the *input* within the Demonstration Exemplar includes the grid states (initial, observed, completed), whereas in Fs-CoT, it does not (Eq. 1).

4.1. Commonsense prompt (CP)

This variant of Tomcat aligns closely with the task presented to the participants: the demonstration exemplar consists of two problem scenarios shared with the participants. Each scenario features the initial, observed, and completed grid states, accompanied by explanations of the events that occur in those states in Fig. 3. Fig. 6 presents an excerpt from the demonstration component of CP (detailed prompt templates are available in Appendix A.3). The input-output format corresponds to Eqs. (1) and (2). The other components of this prompt, i.e., the common ground and the response generation, are the same as those provided to the participants, with only minor modifications, as described in Section 5.

Additional Rules for Fs-CoT

Rules of the Game:

…

…

- Always generate your response by cross-checking and verifying it against the current grid configuration to ensure accuracy.
- When traversing the grid, minimize and optimize your path, avoiding collisions with walls (W) or other obstacles.
- Infer the human's desired gem based on their movement, instruction, and grid configuration. Despite unclear instructions, generate a response that captures the human's intent and ensures successful retrieval of their desired gem.
- Apply theory of mind principles to infer the human's desired gem based on their actions and instructions.

Fig. 7. Additional rules of common ground for Tomcat with Fs-CoT.

4.2. Fs-CoT prompt

This variant employs few-shot chain-of-thought (Fs-CoT) demonstrations, which follow the input-output format of the CP exemplar shown at the bottom of Fig. 6, but *exclude* grid states to prevent exceeding the context window limit of GPT-4o. We include seven input-output demonstrations ($k = 7$) to guide Tomcat in understanding the expected response format. Through iterative testing [24], we observed that Tomcat's performance plateaued at $k = 7$ demonstrations, with no further gains from adding more examples. The remaining components follow those in the CP, with targeted enhancements to the common ground component, particularly in the *Rules* portion within the component, as shown in Fig. 7. Through prompt engineering, we found that appending relevant rules helps establish a robust common ground between Tomcat and the principal. The detailed prompt templates are available in Appendix A.4.

5. Experiments

The Instruction Inference task allows us to evaluate the ToM capabilities of an agent by assessing how well it interprets instructions, infers the principal's intent to retrieve a gem, and generates responses aligned with that intent. To this end, we conducted experiments using the Tomcat variants and an empirical study involving 52 human participants who acted as agents collaborating with the principal. The participants were undergraduate and graduate students with STEM backgrounds. This study was approved by our Institutional Review Board (IRB) with an *exempt* status.

Participants received a detailed problem description. This description includes an overview of the Doors, Keys, and Gems environment, the objectives, rules, two problem scenarios, and a task description. Participants were tasked with inferring the principal's intentions, generating an optimal response, and outlining the sequence of optimal actions. The problem description provided to participants was identical to Tomcat's CP framework, except for two key differences. First, the problem scenarios for the participants featured animated GIF representations of the grid states, whereas CP used static matrix representations accompanied by explanatory text. Second, whereas Tomcat was explicitly tasked with identifying the instruction type as part of its response, participants were not required to perform this step.

5.1. Dataset and LLM

We used 16 grid configurations, adapted from Zhi-Xuan et al. [3], as the baseline for evaluating ToM capabilities on Instruction Inference. We created 20 instructions (eight clear or direct, six incomplete, and six ambiguous), and constructed 20 grid-instruction problem scenarios. These scenarios were designed from the 16 grid configurations and 20 instructions to represent specific ToM tasks on instruction inference, rather than pairing every grid with every instruction. We used three LLMs—GPT-4o, DeepSeek-R1, and Gemma-3-27B—for Tomcat's reasoning and response generation. We selected these LLMs because of their large context window. To optimize performance, we capped the maximum token count to 512, striking a balance between comprehensive outputs and computational efficiency. The temperature parameter was set to 0.2 to reduce variability in responses while allowing some nondeterminism.

Each scenario was evaluated using three LLMs, two Tomcat variants, and 52 human participants (each completing 10 scenarios). Each evaluation produced three structured outputs (intent inference, natural-language response, and action-plan formulation), enabling a fine-grained analysis of ToM reasoning. In total, our design yields: $(3\ \text{LLMs} \times 2\ \text{variants} \times 20\ \text{scenarios}) + (52\ \text{humans} \times 10\ \text{scenarios each}) = 640$ scenario-level responses, corresponding to 1920 evaluated reasoning components. This scale supports systematic model-model and model-human comparisons as outlined in Section 6.1. It also aligns with established ToM research practices for LLMs that emphasize carefully designed reasoning scenarios over exhaustive task enumeration. For example, Kosinski [20] uses 20 scenarios, and Strachan et al. [19] employ an average of 12 scenarios per ToM task (e.g., false-belief, faux pas, and hinting). Participants reported approximately 10–15 min per scenario, reflecting the multistep reasoning involved in interpreting indirect instructions, inferring latent goals, and formulating action plans. This indicates that each scenario constitutes a concrete Instruction Inference task rather than a simple prompt-response item.

5.2. Evaluation metrics

We employed five metrics to evaluate the ToM capabilities of Tomcat and the participants. We define an additional metric, *Instruction Accuracy*, just for the Tomcat variants and not for the participants. These metrics assess the ability to recognize the principal's intentions and evaluate the quality of actions and the plans generated by both groups. Whereas actions focus exclusively on execution, plans extend to include the interpretation of the principal's next steps, reflecting the principal's mental model. This distinction is crucial for assessing ToM capabilities, as the plans capture deeper cognitive understanding. Accordingly, the metrics are categorized into *actions* and *plans*, as detailed below:

Intent Accuracy assesses how well the generated response aligns with the principal's goal. It is scored as 1 if the goal matches and 0 otherwise.

Action Feasibility determines whether the suggested actions can be executed in the grid, regardless of efficiency. It is calculated as:

$$\text{Action Feasibility} = \frac{|A_{\text{feasible}}|}{|A_{\text{total}}|}, \text{ where} \tag{3}$$

A_{feasible} is the set of feasible actions, and $A_{\text{total}} = A_{\text{ground-truth}} \cup A_{\text{incorrect}}$ is the union of ground-truth and incorrect or unnecessary actions.

Action Optimality measures the ratio of optimal actions correctly performed to total actions.

Plan Feasibility measures the ratio of actions within a plan that are executable in the environment, regardless of efficiency. That is:

$$\text{Plan Feasibility} = \frac{|A_{\text{feasible_plan}}|}{|A_{\text{total_plan}}|}, \text{ where} \tag{4}$$

$A_{\text{feasible_plan}}$ is the set of feasible actions in the plan, and $A_{\text{total_plan}} = A_{\text{ground-truth_plan}} \cup A_{\text{incorrect_plan}}$ is the union of ground-truth and incorrect or unnecessary actions in the plan.

Plan Optimality assesses whether the actions in the plan are in the correct sequence. If correct, 1, else 0.

Instruction Accuracy evaluates whether Tomcat correctly identifies the instruction type (clear or unclear). If correct, 1, else 0.

5.3. Task structure and difficulty

We characterize the 20 problem scenarios in Table 3. The first column represents the scenario number and the "Grid" column shows the grid identifier derived from [3] dataset. Grids "3", "6", "12", and "13" appear twice, each paired with a different instruction, producing distinct grid-instruction scenarios.

Grid Features. We report basic structural properties of each grid layout, including the grid dimensions and the number of keys and doors present in the environment. These quantities determine the size of the navigation space and the number of objects present in the collaborative task.

Spatial Obstruction. We quantify the navigation constraints of the grid using two heuristics: *detour ratio* and *corridor tightness*.

For a target key location k and agent start position m, the detour ratio is defined as:

$$\rho(m,k) = \frac{d_{\text{sp}}(m,k)}{\max(1, d_{\text{man}}(m,k))}, \text{ where} \tag{5}$$

d_{sp} denotes the shortest-path distance on the grid (computed via BFS) and d_{man} denotes the Manhattan distance. For instance, in Scenario 1 (Fig. 8b, Grid 3), two red keys serve as candidate targets: k_1 at (1, 3) and k_2 at (1, 7), yielding $\rho(m,k_1) = 6/4 = 1.50$ and $\rho(m,k_2) = 2/2 = 1.00$ respectively. The aggregate detour ratio for the scenario, ρ_{agg}, is computed by averaging $\rho(m,k)$ over the number of keys *fetched by the agent*.

$$\rho_{\text{agg}} = \frac{1}{|T|} \sum_{k \in T} \rho(m,k), \text{ where} \tag{6}$$

T is the set of target keys fetched by the agent in a scenario. Thus, $\rho_{\text{agg}} = (1.50 + 1.00)/2 = 1.25$ for Scenario 1.

To capture narrow navigation corridors induced by walls, we measure corridor tightness for each cell, c, along the agent's shortest path, $P(m,k)$, from its start position m to a target key k. The corridor tightness is defined as:

$$\beta(m,k) = \frac{|\{c \in P(m,k) : \deg(c) \leq 2\}|}{|P(m,k)|}, \text{ where} \tag{7}$$

Table 3
Scenario Characterization: 20 unique grid-instruction problem scenarios from the dataset, showing systematic variation across grid metrics (dimensions, keys, doors), obstruction level, ambiguity types (deictic references, implicit goal inference), plan depth (subgoals), and empirical difficulty measured by human participants' accuracy in recognizing the principal's intent.

		Grid Features				Ambiguity			
No.	Grid	Dims	Keys	Doors	Obstr.	Deict.	Impl.	Sub.	Acc.
1	3	10×12	5	5	Moderate	✓	✓	7	0.84
2	5	9×10	3	5	High	✗	✓	7	0.92
3	6	11×11	4	7	High	✗	✓	5	0.68
4	7	11×11	5	5	High	✗	✗	6	0.92
5	10	10×11	4	7	High	✗	✓	12	0.88
6	12	7×9	2	3	Low	✓	✓	4	0.76
7	13	7×11	3	5	Moderate	✗	✓	4	0.84
8	15	9×8	2	3	Moderate	✓	✓	5	0.88
9	18	8×8	2	4	Moderate	✓	✓	5	0.84
10	20	8×8	2	4	Moderate	✓	✓	5	0.80
11	3	10×12	5	5	Moderate	✓	✓	7	0.74
12	6	11×11	4	7	High	✗	✓	5	0.70
13	8	12×13	5	4	High	✗	✗	7	0.81
14	9	9×11	3	5	High	✗	✓	7	0.56
15	10b	10×11	4	7	High	✗	✓	12	0.93
16	11	8×12	3	5	Moderate	✓	✓	7	0.78
17	12	7×9	2	3	Low	✓	✗	3	0.81
18	13	7×11	3	5	Moderate	✓	✓	4	0.78
19	17	8×8	2	3	Moderate	✓	✓	5	0.93
20	19	7×10	3	5	Moderate	✗	✓	8	0.70

$|P(m,k)|$ denotes the number of cells along the agent's shortest path to the required keys, and $\deg(c)$ denotes the number of traversable neighboring cells of c in the grid's 4-neighborhood. Cells with $\deg(c) \leq 2$ correspond to narrow passages where movement is highly constrained. Thus, this heuristic measures the fraction of path cells lying in corridor-like regions of the grid. Continuing with Scenario 1, the path to k_1 spans 7 cells, of which 4 have $\deg(c) \leq 2$, yielding $\beta(m,k_1) = 4/7 \approx 0.57$; the path to k_2 spans 3 cells, of which 2 are narrow, yielding $\beta(m,k_2) = 2/3 \approx 0.67$. The aggregate corridor tightness, β_{agg}, is computed by averaging $\beta(m,k)$ over the number of target keys.

$$\beta_{\text{agg}} = \frac{1}{|T|} \sum_{k \in T} \beta(m,k), \text{ where} \tag{8}$$

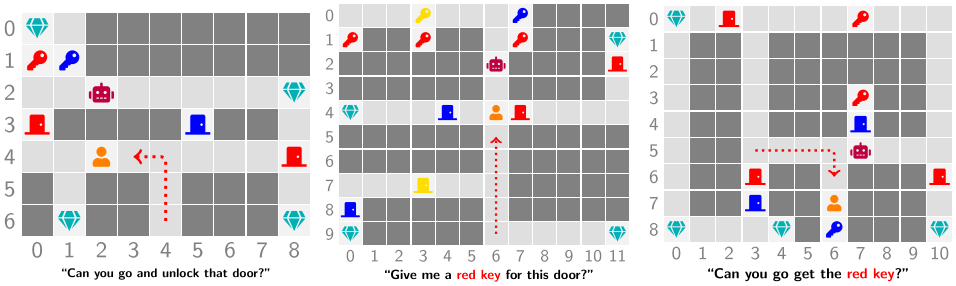
T is the set of target keys fetched by the agent in a scenario. Thus, $\beta_{\text{agg}} = (0.57 + 0.67)/2 \approx 0.62$ for Scenario 1.

Based on ρ_{agg} and β_{agg}, scenarios are categorized into *Low*, *Moderate*, and *High* obstruction levels.

$$O = \begin{cases} \text{Low}, & \rho_{\text{agg}} \leq 1.15 \ \wedge \ \beta_{\text{agg}} < 0.5 \\ \text{Moderate}, & \rho_{\text{agg}} \leq 1.5 \ \wedge \ 0.5 \leq \beta_{\text{agg}} < 0.8 \\ \text{High}, & \text{otherwise.} \end{cases} \tag{9}$$

The thresholds reflect interpretable levels of spatial constraint in the grid environment. Values of $\rho_{\text{agg}} \leq 1.15$ indicate near-Manhattan navigation with minimal detours, $1.15 < \rho_{\text{agg}} \leq 1.5$ indicates moderate detours around walls, and $\rho_{\text{agg}} > 1.5$ indicates substantial path detours caused by obstacles. Further, β_{agg} distinguishes open paths ($\beta_{\text{agg}} < 0.5$), corridor-dominated navigation ($0.5 \leq \beta_{\text{agg}} < 0.8$), and strongly constrained paths ($\beta_{\text{agg}} \geq 0.8$), yielding the Low, Moderate, and High obstruction categories. Scenario 1 thus falls in the Moderate category ($\rho_{\text{agg}} = 1.25$, $\beta_{\text{agg}} = 0.62$). Fig. 8 shows a representative grid from each obstruction level.

Ambiguity and Planning Depth. We annotate each scenario for two forms of ambiguity: *deictic ambiguity*, where the instruction contains spatial references such as, *this*, *that*, or *there*, and *implicit goal inference*, where the agent must infer the principal's intended objective from contextual cues such as the principal's movement, the available resources (e.g., keys or doors), and the division of labor in the collaborative task. For instance, Scenario 17 (Fig. 8a) contains deictic ambiguity but not implicit goal inference: the instruction "Can you go and unlock *that* door?" uses the spatial reference *that*, yet the target door can be identified from the principal's movement toward the red door. Scenario 1 (Fig. 8b) contains both forms: the instruction "Give me a red key for *this* door?" uses *this* to indicate the red door near which the principal is standing, and it implies collecting a single red key; however, two red doors block the path to the target gem, so the agent must infer that two red keys in the grid are required to fulfill the principal's goal. Scenario 14 (Fig. 8c) contains implicit goal inference without deictic ambiguity: the instruction "Can you go get the red key?" is spatially unambiguous, but the grid contains two red keys—one accessible and one locked behind a blue door. Since the principal is observed moving toward the blue key, the agent infers that while the principal collects the blue key, it should fetch and pass the accessible red key to the principal who then unlocks both doors to reach the gem. We also report the number of subgoals required in the corresponding plan, which reflects the depth of the cooperative task.



(a) Scenario 17, Grid 12. Low obstruction ($\rho_{agg} = 1.00$, $\beta_{agg} = 0.25$) with deictic ambiguity, but no implicit goal inference.

(b) Scenario 1, Grid 3. Moderate obstruction ($\rho_{agg} = 1.25$, $\beta_{agg} = 0.62$) with deictic ambiguity and implicit goal inference.

(c) Scenario 14, Grid 9. High obstruction ($\rho_{agg} = 2.20$, $\beta_{agg} = 1.00$) with implicit goal inference, but no deictic ambiguity.

Fig. 8. Example scenarios in the Doors, Keys, and Gems world, showing spatial obstruction levels (low, moderate, high) and ambiguity types (deictic, implicit goal inference).

Table 4
Performance comparison between LLMs and participants across metrics.

Agents	Method	Intent Acc. (%)	Act. Feas. (%)	Act. Opt. (%)	Plan Feas. (%)	Plan Opt. (%)
GPT-4o	CP	50.00	78.75	55.00	60.92	25.00
	Fs-CoT	**85.00**	**97.50**	**92.50**	**91.78**	**75.00**
DeepSeek-R1	CP	60.00	86.00	73.50	69.90	55.00
	Fs-CoT	80.00	88.50	73.50	80.45	70.00
Gemma-3-27B	CP	5.00	60.42	30.42	32.64	5.00
	Fs-CoT	30.00	75.00	51.25	49.78	30.00
Human	Participants	80.38	89.26	86.22	79.02	71.15

Table 5
Results of McNemar's exact test with Cohen's g for binary metrics.

LLMs	Metric	Comparison	p	Cohen's g
GPT-4o	Intent Accuracy	Fs-CoT vs CP	0.039	0.778
	Plan Optimality	Fs-CoT vs CP	0.002	1.000
	Instruction Accuracy	Fs-CoT vs CP	0.250	1.000
DeepSeek-R1	Intent Accuracy	Fs-CoT vs CP	0.125	1.000
	Plan Optimality	Fs-CoT vs CP	0.375	0.600
	Instruction Accuracy	Fs-CoT vs CP	0.125	1.000
Gemma-3-27B	Intent Accuracy	Fs-CoT vs CP	0.063	1.000
	Plan Optimality	Fs-CoT vs CP	0.063	1.000
	Instruction Accuracy	Fs-CoT vs CP	0.375	0.600

Difficulty. We report human participants' mean accuracy in recognizing the principal's intent for each scenario as an indicator of task difficulty.

6. Results and discussion

Table 4 reveals that Tomcat with Fs-CoT consistently outperformed its CP counterpart in all metrics, regardless of the underlying LLM. Among the Fs-CoT variants, GPT-4o outperformed, DeepSeek-R1 performed comparably to, and Gemma-3-27B performed worse than human participants. In contrast, Tomcat with CP consistently underperformed relative to participants in all metrics, although DeepSeek-R1 exhibited performance that was moderately close to, but still below, that of participants and its Fs-CoT variant. In general, DeepSeek-R1 was the strongest among the CP variants, whereas the GPT-4o Fs-CoT variant achieved the best overall performance. We substantiate these findings with further analyses below.

6.1. Statistical tests

To substantiate these findings, we conducted statistical significance tests for each metric between the three groups—two Tomcat variants and participants—separately for each LLM. We conducted a Kruskal-Wallis test followed by post hoc Dunn tests with Holm-

Table 6
Results of Chi-square test with Cramér's V for binary metrics.

LLMs	Metric	Comparison	p	Cramér's V
GPT-4o	Intent Accuracy	Participants vs CP	0.003	0.129
		Participants vs Fs-CoT	0.823	0.010
	Plan Optimality	Participants vs CP	< 0.001	0.179
		Participants vs Fs-CoT	0.903	0.005
DeepSeek-R1	Intent Accuracy	Participants vs CP	0.053	0.083
		Participants vs Fs-CoT	1.000	0.000
	Plan Optimality	Participants vs CP	0.192	0.056
		Participants vs Fs-CoT	1.000	0.000
Gemma-3-27B	Intent Accuracy	Participants vs CP	< 0.001	0.330
		Participants vs Fs-CoT	< 0.001	0.220
	Plan Optimality	Participants vs CP	< 0.001	0.259
		Participants vs Fs-CoT	< 0.001	0.158

Table 7
Kruskal-Wallis and Dunn post hoc test results for continuous metrics. Dunn test pairwise comparisons include adjusted p-values using Holm-Bonferroni correction and effect sizes (r) using Rank-biserial correlation.

LLMs	Metric	p	Dunn-Test	Comparison	Adjusted p	r
GPT-4o	Action Feasibility	0.003	Yes	Fs-CoT vs CP	0.005	0.353
				Participants vs CP	0.003	0.245
				Participants vs Fs-CoT	0.332	−0.072
	Action Optimality	< 0.001	Yes	Fs-CoT vs CP	< 0.001	0.545
				Participants vs CP	< 0.001	0.452
				Participants vs Fs-CoT	0.752	−0.025
	Plan Feasibility	0.003	Yes	Fs-CoT vs CP	0.004	0.550
				Participants vs CP	0.004	0.336
				Participants vs Fs-CoT	0.277	−0.110
DeepSeek-R1	Action Feasibility	0.254	No	–	–	–
	Action Optimality	0.017	Yes	Fs-CoT vs CP	1.000	0.000
				Participants vs CP	0.122	0.177
				Participants vs Fs-CoT	0.122	0.177
	Plan Feasibility	0.445	No	–	–	–
Gemma-3-27B	Action Feasibility	< 0.001	Yes	Fs-CoT vs CP	0.485	0.175
				Participants vs CP	< 0.001	0.365
				Participants vs Fs-CoT	< 0.001	0.295
	Action Optimality	< 0.001	Yes	Fs-CoT vs CP	0.154	0.285
				Participants vs CP	< 0.001	0.638
				Participants vs Fs-CoT	< 0.001	0.465
	Plan Feasibility	< 0.001	Yes	Fs-CoT vs CP	0.165	0.325
				Participants vs CP	< 0.001	0.607
				Participants vs Fs-CoT	< 0.001	0.407

Bonferroni correction for the continuous metrics: action feasibility, action optimality, and plan feasibility. In addition, we calculated the Rank-biserial correlation (r) to quantify the effect sizes for pairwise comparisons, and the results are summarized in Table 7. We performed McNemar's exact test to compare the performance of the two Tomcat variants on the binary metrics—intent accuracy, plan optimality, and instruction accuracy—leveraging their paired data structure. To quantify effect size, we report Cohen's g. We conducted Chi-square tests and calculated Cramér's V as an effect size to compare the performance of the participants and the two Tomcat variants across the binary metrics, given their independent data. The results of these tests for the binary metrics are shown in Tables 5 and 6. The following sections present a comprehensive analysis of performance in all metrics, statistical significance results, qualitative analysis, and responses to research questions.

6.2. Intent accuracy

The GPT-4o Fs-CoT variant achieved an improvement of 70% over its CP counterpart and 5.74% over the participants in intent recognition accuracy. In contrast, neither of the DeepSeek-R1 Tomcat variants outperformed the participants; however, the Fs-CoT variant achieved results comparable to the participants and demonstrated a 33.33% improvement over its CP variant. Similarly,

both Gemma-3-27B Tomcat variants underperformed relative to the participants, though its Fs-CoT variant achieved a substantial improvement of 500% over its CP counterpart.

Significance tests: We assessed statistical significance for intent accuracy using McNemar's exact test to compare Tomcat variants (Table 5) and Chi-square tests to compare variants with participants (Table 6). We report results for each LLM below:

- **GPT-4o:** We observed a significant difference between Fs-CoT and CP ($p = .039$), with a large effect size ($g = 0.78$), favoring Fs-CoT. We found substantial significant differences between participants and CP ($p = .0026$) with a small effect ($V = 0.13$), indicating CP's underperformance, whereas we observed no significant difference between participants and Fs-CoT ($p = .823$; $V = 0.01$).
- **DeepSeek-R1:** We observed no significant difference between Fs-CoT and CP ($p = .125$), despite a large effect size ($g = 1$), supporting Fs-CoT. Compared to the participants, we did not observe significant differences for either variant ($p = .053$ for CP; $p = 1$ for Fs-CoT), though CP approached significance with a small effect size ($V = 0.08$).
- **Gemma-3-27B:** We observed no significant difference between Fs-CoT and CP ($p = .063$), although the result was marginally nonsignificant with a large effect size ($g = 1$), favoring Fs-CoT. However, the test revealed that both variants differed significantly from the participants ($p < .001$), with moderate to small effect sizes ($V = 0.33$ for CP; $V = 0.22$ for Fs-CoT).

Qualitative analysis: In examples such as *Can you unlock this red door?* (scenario 9), where both red and blue doors must be unlocked to reach the gem, the CP variants of GPT-4o and DeepSeek-R1 often misinterpret the instruction, treating the request as an isolated task rather than part of a coordinated effort to retrieve the gem beyond the red and blue doors. In contrast, their Fs-CoT counterparts correctly infer the underlying goal and generate responses aligned with that intention. Further, the Gemma-3-27B Tomcat variants generally fail to recognize the principal's ultimate objective. For instance, in following the instruction, *I'll get the blue key. Can you pick up a red key?* (scenario 13), both variants perform the literal action of passing the red key without acknowledging the principal's next steps of gem retrieval. Although the Gemma-3-27B Fs-CoT variant shows a slight improvement in intent recognition, it does not generate a coherent plan in most cases. These patterns suggest that the CP variants of GPT-4o and DeepSeek-R1, as well as both Gemma-3-27B Tomcat variants, struggle with latent goal inference when instructions are indirect or rely on shared context. However, the Fs-CoT variants of GPT-4o and DeepSeek-R1 benefit from reasoning exemplars that bridge intention and action.

Participant analysis: Most participants correctly identified the principal's intent. On average, three to four participants per instruction failed to do so, possibly due to weaker ToM reasoning or misinterpretation of the task. Interestingly, even participants who typically performed well made mistakes in certain instances. For example, in scenario 14 (Fig. 8c), the grid contained two red keys, one easily accessible but far from the principal-agent pair, and another located behind a locked blue door. The principal issued an instruction *Can you go get the red key?* as it was moving toward a blue key, implying a multistep plan to unlock both the red and blue doors in order to retrieve a gem placed beyond them. This meant the appropriate response was to retrieve the accessible red key, pass it to the principal, and support the shared objective of unlocking the doors and retrieving the gem. However, some participants instead attempted to retrieve the red key behind the blue door or focused only on gems behind a single red door, missing the principal's broader intention. Consequently, the average participants' performance was the worst for this scenario (Table 3). In contrast, both Fs-CoT variants of GPT-4o and DeepSeek-R1 correctly inferred the intended plan and coordinated their actions accordingly. Apart from these rare lapses, participants generally exhibited strong performance in inferring the principal's latent goals even under unclear instructions.

Takeaways: These findings underscore the effectiveness of Fs-CoT for LLMs like GPT-4o and DeepSeek-R1 in enabling Tomcat to infer the principal's desired gem from incomplete or ambiguous instructions, unlike CP, which lacks reasoning exemplars. The significance tests indicate that the Fs-CoT variants of GPT-4o and DeepSeek-R1 perform comparably with the participants. However, the Gemma-3-27B Fs-CoT variant performed worse than the participants and other LLMs. This may stem from its smaller model size, which could limit its capacity for complex reasoning required for Instruction Inference.

6.3. Action feasibility and optimality

For action feasibility, the GPT-4o Fs-CoT variant achieved an improvement of 23.81% over its CP counterpart and 9.23% over the participants. The DeepSeek-R1 Tomcat variants performed comparably to the participants, though with slightly lower performance. Both Gemma-3-27B Tomcat variants failed to reach the performance level of the participants, though its Fs-CoT variant showed an improvement of 24.13% over CP.

For action optimality, the GPT-4o Fs-CoT variant demonstrated a considerable gain of 68.18% over CP and 7.28% over the participants. The DeepSeek-R1 Tomcat variants performed equally and yielded lower performance than the participants. Similarly, both Gemma-3-27B Tomcat variants underperformed relative to the participants, though its Fs-CoT variant achieved a notable gain of 68.47% over CP.

Significance tests: We assessed statistical significance for action feasibility and optimality using the Kruskal-Wallis test, followed by Dunn's post hoc comparisons where applicable (Table 7). We report results per LLM below:

- **GPT-4o:** For *action feasibility*, the Kruskal-Wallis test identified significant differences across groups ($p = .003$). The Dunn tests showed that CP underperformed both Fs-CoT ($p = .005$, $r = 0.353$) and participants ($p = .003$, $r = 0.245$), with moderate to small effect sizes, respectively. However, the test revealed no significant difference between participants and Fs-CoT ($p = .332$, $r = -0.07$).

Similarly, for *action optimality*, the Kruskal-Wallis test revealed notable significant differences across groups ($p < .001$). The Dunn tests demonstrated that CP substantially underperformed relative to both Fs-CoT ($p < .001$, $r = 0.545$) and participants ($p < .001$, $r = 0.452$), with large to moderate effect sizes, respectively. In contrast, the test revealed no significant difference between participants and Fs-CoT ($p = .752$, $r = -0.025$).

- **DeepSeek-R1:** For *action feasibility*, the Kruskal-Wallis test revealed no significant differences across groups ($p = .254$); therefore, we did not conduct the Dunn tests.

  For *action optimality*, the test revealed significant differences across groups ($p = .017$). However, post hoc comparisons did not indicate significant differences between any of the pairs. That is, CP, Fs-CoT, and participants performed comparably, though participants scored higher than the Tomcat variants.
- **Gemma-3-27B:** The Kruskal-Wallis tests for both *action feasibility* and *action optimality* revealed significant differences across groups ($p < .001$ for both). Dunn tests showed that participants outperformed both CP ($p < .001$, $r = 0.365$ for feasibility; $p < .001$, $r = 0.638$ for optimality) and Fs-CoT ($p < .001$, $r = 0.295$ for feasibility; $p < .001$, $r = 0.465$ for optimality), with moderate to large effect sizes. However, we observed no significant differences between Fs-CoT and CP in either metric ($p = .485$, $r = 0.175$ for feasibility; $p = .154$, $r = 0.285$ for optimality).

Qualitative analysis: The CP variants of GPT-4o and Gemma-3-27B often produce actions that are technically feasible but suboptimal. For example, in scenario 4, *I'm picking up the yellow key. Can you get a red key?*, multiple red keys are available on the grid. However, the CP variants for both LLMs retrieve a poorly positioned red key that is feasible but not efficient. In contrast, the Fs-CoT variants select the more strategically placed red key, demonstrating an improved understanding of optimality.

By comparison, both DeepSeek-R1 Tomcat variants execute actions similarly in most scenarios. For instance, in scenario 1 (Fig. 8b)—*Give me a red key for this door?*—retrieving the gem requires two red keys. However, both variants retrieve only the nearest key and generate a follow-up action claiming the second key is inaccessible due to walls, which is incorrect. Consequently, the DeepSeek-R1 Fs-CoT variant fails to improve over CP in terms of action feasibility and optimality. Although both variants struggled in specific edge cases, the overall performance of the CP variant was already strong in these metrics, which may have limited the observable gains from Fs-CoT.

Participant analysis: Most participants who correctly inferred the principal's intent were also able to execute the actions optimally. In contrast to the suboptimal behavior of CP variants in the mentioned LLM cases—choosing a well-positioned red key or retrieving two red keys—participants consistently selected more efficient options. In both examples where the CP variants underperformed, human participants executed the tasks optimally.

The few failure cases among participants typically stemmed from a misinterpretation of the task setup or difficulty interpreting unclear instructions. For example, in scenario 18 *Can you help me unlock the blue door there?*, where two blue doors were present, a small number of participants unlocked the incorrect one. This occurred even though the principal's movements clearly implied that the blue door adjacent to the red door was part of a multistep plan to retrieve a gem. Such failures were rare, originating primarily under ambiguous instructions and affecting only a handful of participants. Overall, participant performance demonstrated strong action feasibility and optimality, particularly when the principal's intent was clearly understood.

Takeaways: These results suggest that the Fs-CoT variants of GPT-4o and Gemma-3-27B generate more optimal and error-free actions over its CP counterpart. This improvement can be attributed to the structured reasoning provided by Fs-CoT demonstrations and the additional rules in the common ground component. However, this benefit does not extend to DeepSeek-R1, as both of its Tomcat variants behave similarly in action execution. Further, the statistical tests indicate that the GPT-4o Fs-CoT variant and both DeepSeek-R1 Tomcat variants perform comparably to participants, whereas both Gemma-3-27B Tomcat variants underperformed.

6.4. Plan feasibility and optimality

For plan feasibility, the GPT-4o Fs-CoT variant achieved an improvement of 50.66% over its CP counterpart and 16.14% over the participants. The DeepSeek-R1 Fs-CoT variant had a gain of 16.56% over CP and performed slightly better than the participants. The Gemma-3-27B Fs-CoT variant also exhibited an improvement of 52.51% over CP, though both variants fell short of participant performance.

For plan optimality, the GPT-4o Fs-CoT variant demonstrated a substantial gain of 200% over CP and a modest gain of 5.41% over the participants. The DeepSeek-R1 Fs-CoT variant improved by 27.27% over CP but remained slightly below participant performance. Further, both Gemma-3-27B Tomcat variants achieved worse results than the participants, though the Fs-CoT variant achieved a notable gain of 500% over CP.

Significance tests: We evaluated statistical significance for plan optimality using McNemar's exact test to compare Tomcat variants (Table 5) and Chi-square tests to compare each variant with participants (Table 6). For plan feasibility, we used the Kruskal-Wallis test, followed by Dunn's post hoc comparisons where applicable (Table 7). We report results per LLM below:

- **GPT-4o:** For *plan feasibility*, the Kruskal-Wallis test revealed significant differences across groups ($p = .003$). The Dunn tests indicated that CP underperformed both Fs-CoT ($p = .004$, $r = 0.55$) and participants ($p = .004$, $r = 0.336$), with large to moderate effect sizes, respectively. However, there was no significant difference between participants and Fs-CoT ($p = .277$, $r = -0.11$).

  For *plan optimality*, McNemar's exact test found a significant difference between Fs-CoT and CP ($p = .002$, $g = 1$), with a large effect size. The Chi-square test demonstrated a notable significant difference between participants and CP ($p < .001$, $V = 0.179$), with a small effect, but no significant difference between participants and Fs-CoT ($p = .903$, $V = 0.005$).

Table 8
Instruction identification metrics for Tomcat.

LLMs	Method	Precision	Recall	F1-score	Accuracy
GPT-4o	CP	0.60	**1.00**	0.75	0.60
	Fs-CoT	0.71	**1.00**	**0.83**	**0.75**
DeepSeek-R1	CP	0.58	0.92	0.71	0.55
	Fs-CoT	**0.73**	0.92	0.81	**0.75**
Gemma-3-27B	CP	0.56	0.83	0.67	0.50
	Fs-CoT	0.67	0.83	0.74	0.65

- **DeepSeek-R1:** For *plan feasibility*, the Kruskal-Wallis test revealed no significant differences across groups ($p = .445$); hence, Dunn tests were not performed.

  For *plan optimality*, McNemar's exact test indicated no significant difference between Fs-CoT and CP ($p = .375$, $g = 0.6$), despite a large effect size. Similarly, Chi-square results revealed no significant differences between participants and either variant ($p = .192$ for CP, $p = 1$ for Fs-CoT).
- **Gemma-3-27B:** For *plan feasibility*, the Kruskal-Wallis test revealed significant differences across groups ($p < .001$). The Dunn tests showed that participants outperformed both CP ($p < .001$, $r = 0.607$) and Fs-CoT ($p < .001$, $r = 0.407$), with large to moderate effect sizes, respectively. We observed no significant difference between Fs-CoT and CP ($p = .165$, $r = 0.325$), despite a moderate effect size favoring Fs-CoT.

  For *plan optimality*, McNemar's exact test found no significant difference between Fs-CoT and CP ($p = .063$), although the result approached significance with a large effect size ($g = 1$), supporting Fs-CoT. However, the Chi-square test showed that participants outperformed both CP ($p < .001$, $V = 0.259$) and Fs-CoT ($p < .001$, $V = 0.158$), with moderate to small effect sizes, respectively.

Qualitative analysis: The CP variants of GPT-4o and DeepSeek-R1 often generate plans that are feasible but lack goal-directed coherence. Consider the example in scenario 18, *Can you help me unlock the blue door there?*, where multiple blue doors are present. Based on the principal's movements, the blue door adjacent to the red door must be unlocked to retrieve the gem. The GPT-4o CP variant passes the key to the principal without unlocking the door, whereas the DeepSeek-R1 CP variant unlocks a different blue door. In contrast, their Fs-CoT counterparts generate plans that enable the principal to proceed seamlessly through both doors and retrieve the gem.

In another example (scenario 13), *I'll get the blue key. Can you pick up a red key?*, multiple red keys are available, but the one closest to the blue door must be selected for efficient coordination. The GPT-4o CP variant retrieves a distant red key, failing to generate a coherent plan for gem retrieval, whereas its Fs-CoT counterpart collects the optimal key. In contrast, both DeepSeek-R1 Tomcat variants execute the action correctly by picking up the optimal red key. This suggests that the DeepSeek-R1 CP variant generates more coherent plans than the GPT-4o CP variant. These observations indicate that both Fs-CoT variants of GPT-4o and DeepSeek-R1—and, in some cases, the DeepSeek-R1 CP variant—model long-term dependencies and shared intentions, resulting in plans that are both feasible and optimal.

However, the Gemma-3-27B Tomcat variants in both examples execute only the immediate action without generating a plan that considers the principal's intended next steps. Although the Fs-CoT variant improves plan feasibility and optimality for Gemma-3-27B, its overall planning performance trails substantially behind the other LLMs.

Participant analysis: In both examples discussed—unlocking the correct blue door and retrieving the red key adjacent to the blue door—participants' plans were generally feasible and aligned with the principal's objective. However, a few responses were sub-optimal, primarily because they omitted references to the principal's follow-up actions for gem retrieval. Even when participants identified the correct key or door, their responses occasionally lacked the broader coordination necessary for optimal planning.

Additional failures occurred when some participants misunderstood the principal's intent entirely, leading to incorrect goal identification and less effective action choices in planning. Nevertheless, participant performance consistently surpassed the CP variants, which frequently failed to generate coherent plans. In terms of plan feasibility and optimality, the Fs-CoT variants of GPT-4o and DeepSeek-R1 closely matched participant performance.

Takeaways: These findings indicate that across all LLMs, Tomcat with Fs-CoT outperformed its CP counterpart in generating feasible and optimal plans. Although we observed no significant difference in the DeepSeek-R1 Tomcat variants, its Fs-CoT variant exhibited substantial gains over CP. Further, since effective planning requires anticipating the principal's future actions, Fs-CoT variants generally demonstrated stronger ToM reasoning than CP. Additionally, the significance tests show that the GPT-4o Fs-CoT variant and both DeepSeek-R1 Tomcat variants performed on par with human participants and exhibited more robust ToM reasoning, whereas both Gemma-3-27B Tomcat variants lagged behind. We attribute this comparable performance to the structured reasoning exemplars in Fs-CoT, which provide a *step-by-step* process for inferring the principal's intention. The underperformance of Gemma-3-27B may be due to its smaller model size, which limits its ability to handle the complexity of ToM reasoning.

Table 9
Mean performance of intent accuracy and planning metrics of each Tomcat variant stratified by obstruction level (n denotes the number of scenarios per level).

LLMs	Method	Metric	Low ($n=2$)	Moderate ($n=10$)	High ($n=8$)
GPT-4o	CP	Intent Accuracy	0.50	0.40	0.63
		Plan Feasibility	0.83	0.56	0.61
		Plan Optimality	0.50	0.30	0.13
	Fs-CoT	Intent Accuracy	0.50	0.90	0.88
		Plan Feasibility	0.63	0.93	0.98
		Plan Optimality	0.50	0.70	0.88
DeepSeek-R1	CP	Intent Accuracy	0.50	0.60	0.63
		Plan Feasibility	0.83	0.65	0.73
		Plan Optimality	0.50	0.50	0.63
	Fs-CoT	Intent Accuracy	0.50	0.80	0.88
		Plan Feasibility	0.75	0.73	0.91
		Plan Optimality	0.50	0.60	0.88
Gemma-3-27B	CP	Intent Accuracy	0.50	0.00	0.00
		Plan Feasibility	0.63	0.27	0.32
		Plan Optimality	0.50	0.00	0.00
	Fs-CoT	Intent Accuracy	0.50	0.30	0.25
		Plan Feasibility	0.63	0.51	0.45
		Plan Optimality	0.50	0.30	0.25

6.5. Instruction accuracy

Table 8 presents Tomcat's instruction identification metrics with Fs-CoT and CP across all LLMs. Both demonstrate excellent recall, demonstrating their ability to recognize incomplete or ambiguous instructions. However, their precision is imperfect. For GPT-4o and DeepSeek-R1, the Fs-CoT variant occasionally misclassified clear instructions as unclear, whereas their CP counterpart underperformed further, misclassifying several clear instructions as unclear. Gemma-3-27B shows similar trends, with lower precision across both variants, though Fs-CoT still achieves higher accuracy and F1-score than CP. This misclassification hurt CP's performance on other metrics. In contrast, despite its lower precision, the Fs-CoT variants excelled in other metrics, suggesting that its enhanced ToM capabilities, bolstered by reasoning exemplars, enable it to better infer principal's intentions and optimize actions and plans.

Further, McNemar's exact test did not reveal statistically significant differences in instruction accuracy between Fs-CoT and CP for any of the LLMs. Specifically, the results were as follows: GPT-4o ($p = .25$, $g = 1.00$), DeepSeek-R1 ($p = .125$, $g = 1.00$), and Gemma-3-27B ($p = .375$, $g = 0.60$). Despite the lack of significance, the large or moderate effect sizes suggest that Tomcat with Fs-CoT outperforms CP in instruction identification, which is further corroborated by Table 8.

6.6. Impact of obstruction level

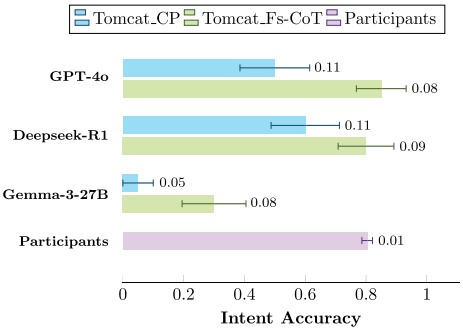
Table 9 reports mean intent accuracy and planning metrics for each Tomcat variant, stratified by obstruction level. Because the Low obstruction level contains only two scenarios ($n = 2$), binary metric scores in that column are constrained to multiples of 0.5 and offer limited comparative value; therefore, the following discussion focuses on the Moderate and High levels.

For GPT-4o, the CP variant's intent accuracy and plan feasibility increase from Moderate to High obstruction, whereas its plan optimality declines across the same range. In contrast, the Fs-CoT variant's plan feasibility and plan optimality rise from Moderate to High, whereas its intent accuracy shows a marginal decline over the same range.

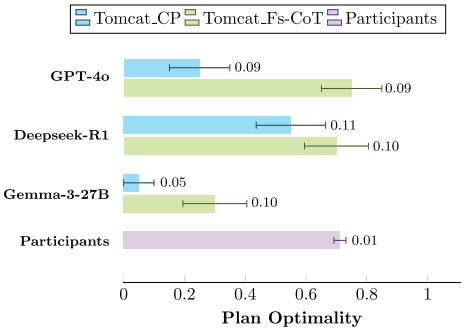
For DeepSeek-R1, both variants exhibit consistent increases across all three metrics from Moderate to High obstruction, with the Fs-CoT variant recording higher absolute values throughout. For Gemma-3-27B, the CP variant records zero intent accuracy and plan optimality at both obstruction levels, whereas its plan feasibility shows a slight increase from Moderate to High. In contrast, the Fs-CoT variant exhibits a slight uniform decline across all three metrics over the same range. The direction and magnitude of change from Moderate to High are inconsistent across models for intent recognition and planning metrics.

Table 9 shows that of the 18 combinations of LLMs, variants, and metrics, 11 increase from Moderate to High, five decrease (four slightly), and two are unchanged. Performance, therefore, tends to increase as the grid becomes more obstructed. We explain this counterintuitive direction using the grid features and the two forms of ambiguity defined in Section 5.3, which move in opposite directions across the levels.

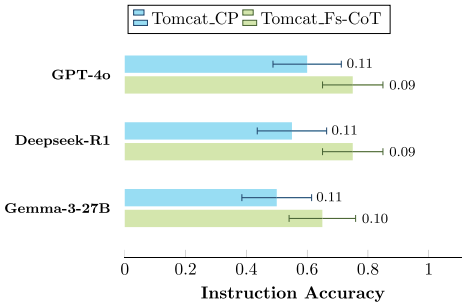
Grid features increase across the obstruction levels. Across Low, Moderate, and High, the mean number of keys is 2.00, 3.00, and 4.00; the mean number of doors is 3.00, 4.40, and 5.88; and the mean number of subgoals is 3.50, 5.70, and 7.62 (Table 3). Conversely, the two forms of ambiguity decrease over the same range. All ten Moderate scenarios require implicit goal inference, and eight contain deictic ambiguity. In contrast, six of the eight High scenarios require implicit goal inference, and none contains deictic ambiguity. Implicit goal inference is present at both levels, making deictic ambiguity the distinguishing factor.



(a) Intent accuracy across LLMs and participants.



(b) Plan optimality across LLMs and participants.



(c) Instruction accuracy across LLMs.

Fig. 9. Binary metrics with standard error of mean (SEM). Participants were not tasked with classifying the instruction type.

This separation matters because deictic ambiguity is the one form that the grid cannot resolve independently. Identifying "that door" requires reading the principal's movement, whereas "the blue door" is recoverable from the layout alone. The High obstruction scenarios, therefore, carry less ambiguity than the Moderate ones. Their additional keys and doors add navigation rather than inference. Tomcat's performance follows the ambiguity rather than the grid features. Further, the obstruction levels do not separate difficulty for participants either. The mean human intent accuracy is 78.50%, 81.30%, and 80.00% across Low, Moderate, and High, respectively.

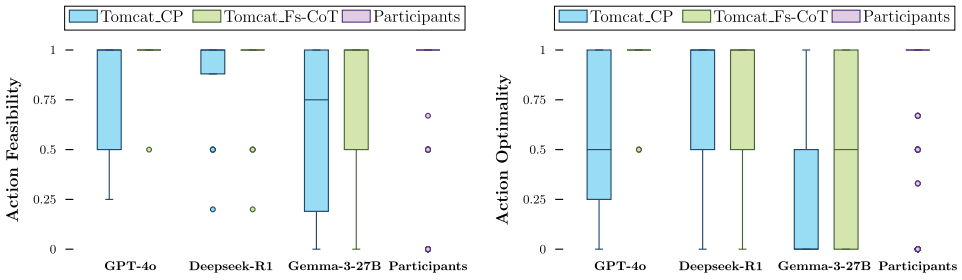
Two properties of the heuristics bound our conclusions. First, both ρ_{agg} and β_{agg} are computed along the shortest path to a target key. They presuppose the goal and measure the cost of executing a plan rather than choosing one. Second, both are scale-invariant. A large grid with a single straight, unobstructed corridor yields $\rho_{\text{agg}} = 1.00$ and a low β_{agg}, indistinguishable from a small open room. We therefore restrict our claim: spatial obstruction does not predict performance for Tomcat or for our participants, whereas instruction type does. This claim applies within the range this corpus spans: grids from 7×9 to 12×13, two to five keys, and four candidate gems in every scenario. Section 10.2 discusses grid scale, which lies outside what these heuristics measure.

6.7. Metrics visualization

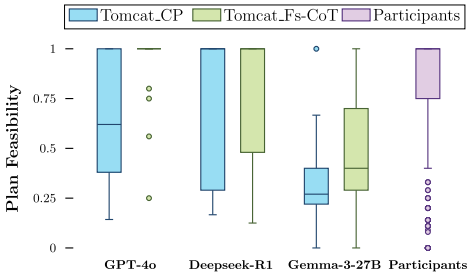
In Fig. 9, the standard error of the mean (SEM) for Tomcat with CP is generally higher than for both Tomcat with Fs-CoT and the participants across all LLMs, indicating greater variability and less reliable average performance. Notably, for the Gemma-3-27B Tomcat variants, the SEM for Fs-CoT for intent accuracy (Fig. 9a) and plan optimality (Fig. 9b) is slightly higher than that of CP, suggesting that despite its improved accuracy, Fs-CoT exhibits more inconsistent outcomes on this LLM. For GPT-4o, the SEM for both variants is identical in plan optimality (Fig. 9b), indicating that Fs-CoT achieves better central performance than CP, yet their variability remains comparable. In contrast, the consistently lower SEMs observed for the Fs-CoT variants of GPT-4o and DeepSeek-R1, as well as for the participants, reflect more stable performance, making their reported means more robust.

In Fig. 10a, for action feasibility, the CP variants of GPT-4o and DeepSeek-R1 show a relatively narrow interquartile range (IQR) with higher medians, indicating low variability and consistently high performance. In contrast, the Gemma-3-27B CP variant exhibits a wider IQR and greater variability, reflecting inconsistent performance. The Fs-CoT variants of GPT-4o and DeepSeek-R1, along with the participants, achieve near-ceiling performance, as shown by their high medians and minimal variability, with only a few minor outliers. However, the Gemma-3-27B Fs-CoT variant shows a noticeably wider IQR and increased variability than the other Fs-CoT variants.

In Fig. 10b, for action optimality, the GPT-4o CP variant and both Gemma-3-27B Tomcat variants exhibit wider IQRs with lower medians, indicating greater variability and overall weaker performance. In contrast, the GPT-4o Fs-CoT variant, both DeepSeek-R1 Tomcat variants, and the participants display higher medians and fewer outliers. Notably, the IQRs for the GPT-4o Fs-CoT variant and



(a) Action feasibility across LLMs and participants. (b) Action optimality across LLMs and participants.



(c) Plan feasibility across LLMs and participants.

Fig. 10. Box plots of continuous metrics with the IQR and median collapsed into a single line, indicating near-optimal performance.

Table 10
Statistical outcomes across all metrics. Here, ✔ denotes $p < .05$, ✖ indicates no significant differences (or that the test could not be conducted), ‡ marks cases where both p-value approaches significance ($0.05 < p < .1$) and the effect was large, and † denotes cases where only one of these conditions (near-significance or large effect) was met.

LLM	Metric	Fs-CoT vs. CP	Part. vs. CP	Part. vs. Fs-CoT
GPT-4o	Intent Accuracy	✔ (g = .78)	✔ (V = .13)	✖
	Action Feasibility	✔ (r = .35)	✔ (r = .25)	✖
	Action Optimality	✔ (r = .55)	✔ (r = .45)	✖
	Plan Feasibility	✔ (r = .55)	✔ (r = .34)	✖
	Plan Optimality	✔ (g = 1.0)	✔ (V = .18)	✖
	Instruction Acc.	† (g = 1.0)	–	–
DeepSeek-R1	Intent Accuracy	† (g = 1.0)	† (p = .053)	✖
	Action Feasibility	✖	✖	✖
	Action Optimality	✖	✖	✖
	Plan Feasibility	✖	✖	✖
	Plan Optimality	† (g = 0.6)	✖	✖
	Instruction Acc.	† (g = 1.0)	–	–
Gemma-3-27B	Intent Accuracy	‡ (g = 1.0)	✔ (V = .33)	✔ (V = .22)
	Action Feasibility	✖	✔ (r = .37)	✔ (r = .30)
	Action Optimality	✖	✔ (r = .64)	✔ (r = .47)
	Plan Feasibility	✖	✔ (r = .61)	✔ (r = .41)
	Plan Optimality	‡ (g = 1.0)	✔ (V = .26)	✔ (V = .16)
	Instruction Acc.	† (g = 0.6)	–	–

the participants are tightly collapsed, suggesting that most values are concentrated around near-optimal performance. The DeepSeek-R1 Tomcat variants exhibit narrow IQRs with high medians, reflecting stable and high performance.

In Fig. 10c, for plan feasibility, the GPT-4o CP variant exhibits a wider IQR and a lower median, indicating variability and weaker performance. In contrast, its Fs-CoT variant shows a higher median and a collapsed IQR, reflecting greater consistency and improvement. The DeepSeek-R1 CP variant shows a wider IQR than GPT-4o CP but a higher median, suggesting better overall performance despite variability. Its Fs-CoT variant, however, displays a tighter IQR and a high median, indicating stable and better performance. The Gemma-3-27B CP variant has a narrower IQR but a lower median than its Fs-CoT counterpart, suggesting that while the latter achieves better performance, it also exhibits higher variability. Participants show a somewhat wider IQR than GPT-4o Fs-CoT but maintain a high median with minimal outliers, indicating consistently strong performance.

6.8. Answering the research questions

We answer the following research questions by the summary statistics of performance in Table 4 and statistical significance tests in Table 10, and analyzing the visualized metrics, with findings reported separately for each LLM.

RQ_1: How do Tomcat's ToM capabilities on Instruction Inference compare to human performance?

- **GPT-4o:** Tomcat with CP underperformed participants in key metrics such as plan optimality and intent accuracy, which are critical indicators of ToM reasoning. In contrast, Tomcat with Fs-CoT demonstrated ToM capabilities on par with the participants, as indicated by the statistical significance testing.
- **DeepSeek-R1:** Tomcat with CP did not match participant performance in planning and intent recognition. Although the statistical tests revealed no significant differences, intent accuracy approached significance ($p = .053$), and participants showed gains over CP in planning metrics, suggesting weaker ToM reasoning for CP. For other metrics, however, CP performed comparably with participants. In contrast, Tomcat with Fs-CoT achieved substantial improvements over CP in both intent accuracy and planning, and performed on par with participants across all ToM-relevant metrics.
- **Gemma-3-27B:** Tomcat with CP underperformed participants across all metrics, particularly in intent recognition and planning. Despite gains in these metrics, the Fs-CoT variant also failed to reach participant-level performance. Statistical tests confirmed significant differences between participants and both variants, and although Fs-CoT improved over CP, the improvement remained insufficient. These findings suggest that Gemma-3-27B lacks the model capacity required for reliable ToM reasoning, even with structured exemplars.

RQ_2: Does Tomcat with Fs-CoT outperform Tomcat with CP on Instruction Inference?

- **GPT-4o:** Fs-CoT demonstrations substantially enhanced Tomcat's ToM capabilities and performance on Instruction Inference. By leveraging these few-shot exemplars, Tomcat effectively interpreted incomplete and ambiguous instructions, inferred human intentions, and optimized actions and plans, achieving notable improvements over Tomcat with CP, as evidenced by its performance across the metrics and significance testing.
- **DeepSeek-R1:** Fs-CoT demonstrations improved Tomcat's performance, particularly in intent recognition, planning, and instruction identification, where it achieved notable gains over CP. Although significance tests did not reveal statistical differences between Fs-CoT and CP, large effect sizes for intent and instruction accuracy, along with consistent improvements in planning metrics, suggest that Fs-CoT enabled more reliable interpretation of unclear instructions. Overall, Fs-CoT improved DeepSeek-R1 Tomcat's ToM capabilities, aligning its performance with that of the participants. Notably, the CP variant of DeepSeek-R1 was the strongest among all CP variants, particularly in the action metrics, where both variants performed similarly.
- **Gemma-3-27B:** Fs-CoT demonstrations improved Tomcat's performance on Instruction Inference, achieving substantial gains over CP across all metrics. Although the statistical tests revealed no significant differences between the two variants, intent accuracy and plan optimality approached significance ($p = .063$), and plan feasibility yielded a moderate effect size, all of which support improved ToM performance with Fs-CoT. However, these gains were less notable than the other LLMs, largely because CP performed poorly. However, despite these strong improvements, Tomcat with Fs-CoT still did not achieve participant-level performance. This outcome likely reflects limitations in Gemma-3-27B's smaller model size, which may constrain its capacity for complex ToM reasoning even when guided by structured exemplars.

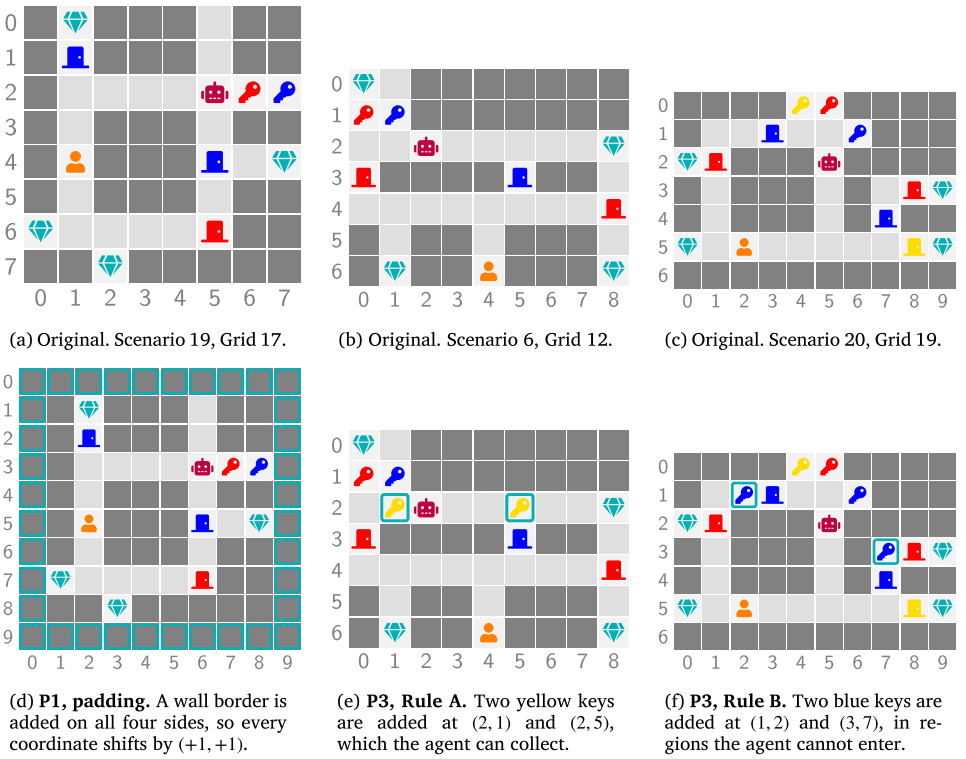
7. Environmental perturbations

Recent studies [46,47] suggest that LLMs suffer from data contamination by replicating learned patterns from their training data rather than demonstrating genuine reasoning. Further, LLMs may be sensitive to specific scenarios and produce variable responses across instances of the same problem. Our Instruction Inference task invites similar concerns. Both the CP and Fs-CoT variants include examples demonstrating how to interpret instructions and execute tasks (Fig. 2). Specifically, CP uses two problem walkthroughs, and Fs-CoT uses seven input-output exemplars. The evaluated LLMs might succeed by recognizing game-specific patterns in these examples, such as recurring coordinates, color words, or object counts, rather than by reasoning over the environment. We therefore construct four perturbation categories for the 20 grid–instruction pairs and explain their design as follows:

P1: Marginal padding. We add a wall border on all four sides of the grid, so every coordinate shifts by $(+1, +1)$. Fig. 11a shows an instance of the original grid, and Fig. 11d shows the padded grid. The optimal action and plan sequences remain unchanged. Only the numeric encoding of each position differs, which isolates dependence on absolute coordinates.

P2: Color permutation. We apply a cyclic color permutation to keys, doors, and the scenario text, including the instruction: red→blue→yellow→red. Fig. 12a shows an instance of the original grid, and Fig. 12d shows the permuted grid. The optimal action and plan sequences remain unchanged. Only the color of the keys and doors changes, which isolates dependence on specific color tokens.

P3: Distractor keys. We add two keys to each of the 16 grids that cannot be part of the optimal plan. Our grids use three colors for keys and doors: red, blue, and yellow. Eight grids carry exactly two of these colors, and the remaining eight carry all three. We adopt one rule for each case. *Rule A (two colors)* adds two keys of the missing color in the agent's own region, so the agent can collect them,



(a) Original. Scenario 19, Grid 17.

(b) Original. Scenario 6, Grid 12.

(c) Original. Scenario 20, Grid 19.

(d) **P1, padding.** A wall border is added on all four sides, so every coordinate shifts by $(+1, +1)$.

(e) **P3, Rule A.** Two yellow keys are added at $(2, 1)$ and $(2, 5)$, which the agent can collect.

(f) **P3, Rule B.** Two blue keys are added at $(1, 2)$ and $(3, 7)$, in regions the agent cannot enter.

Fig. 11. Marginal padding and distractor keys. The upper row shows the original scenario and the lower row the perturbed counterpart. Padding renumbers every coordinate, and both distractor rules add keys. Both preserve the optimal action and plan sequences.

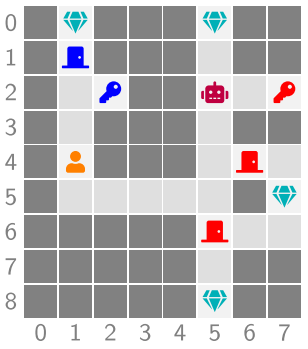
but they unlock nothing. *Rule B (three colors)* adds two keys in a region the agent cannot enter, so the agent cannot collect them at all. Fig. 11b and e show an original grid and its Rule A perturbation, and Fig. 11c and f do the same for Rule B. Neither rule moves an existing object nor places a key on the principal's route. The optimal action and plan sequences remain unchanged. Only the number of keys differs, which isolates dependence on object counts, and Rule B further tests whether an unreachable key distracts the agent.

P4: Key relocation. We move every key to a different empty cell within its own door-bounded region, at least two steps from its original position. We do not move any key that the principal retrieves in any scenario. Therefore, the target gem and the principal's intent remain unchanged. However, the path length to the target keys changes. Consequently, the optimal action and plan sequences change. The relocation decreases the spatial obstruction in some scenarios and increases it in others, as measured by the detour ratio ρ_{agg} and corridor tightness β_{agg}. Some scenarios therefore become easier and others harder. Fig. 12b and e show a scenario becoming easier, and Fig. 12c and f show a scenario becoming harder. Unlike P1–P3, this perturbation alters the optimal action by design, which isolates whether the agent reasons about the grid at all.

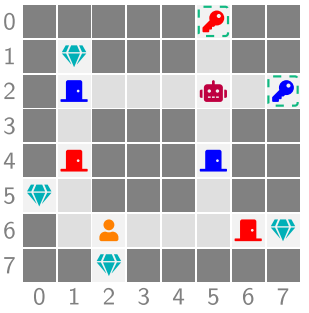
Re-run baseline. We re-ran the original experiments in Section 6 because the endpoints for the DeepSeek-R1 and Gemma-3-27B models used during the original study have been retired, and the available replacements serve quantized weights. We initially accessed DeepSeek-R1 via DeepSeek's native API and Gemma-3-27B via Google's AI Studio endpoint. Both services have since been discontinued. We now use DeepSeek-R1 through Novita and Gemma-3-27B through DeepInfra, both on OpenRouter. We pin the provider in each case and disable fallbacks, so no condition is silently rerouted to a different host. The token budget also changed for DeepSeek-R1, the sole reasoning model among the three. The original study capped generation at 512 tokens for all models. The current Novita endpoint counts reasoning and output tokens against the same limit. Consequently, the reasoning trace exhausts a 512-token cap, returning no answer. Thus, we increase the cap to 16,000 tokens, the ceiling for this provider. Generation then averages 8300 tokens, of which about 6500 are reasoning, and no call reaches the ceiling. We therefore make all comparisons in this section against the re-run, which shares its endpoint, provider, and numerical precision with all perturbed conditions.

Scale. The re-run and the four perturbed categories together give five conditions, yielding $(5 \text{ conditions} \times 3 \text{ LLMs} \times 2 \text{ variants} \times 20 \text{ scenarios}) = 600$ scenario-level responses, corresponding to 1800 evaluated reasoning components. A single seeded script generates the 64 perturbed grids ($4 \text{ perturbations} \times 16 \text{ grids}$), thus the set is reproducible from the original scenarios.

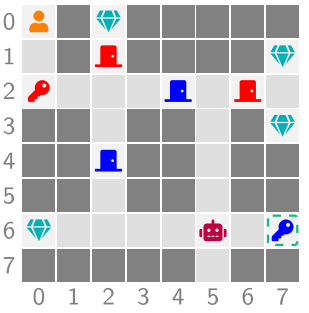
To summarize, P1–P3 preserve the ground-truth optimal action and plan sequences. P1 alters object locations, P2 alters colors, and P3 alters key counts. P4 deliberately alters the optimal action and plan sequences. Thus, a model that reasons over the environment should remain unaffected, whereas a model that relies on game-specific patterns from the provided prompt examples should degrade.



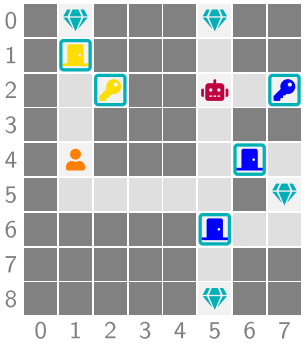
(a) Original. Scenario 8, Grid 15. *"Can you open the red door there?"*



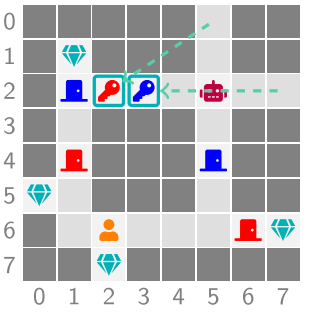
(b) Original. Scenario 9, Grid 18. The two keys sit at $(0, 5)$ and $(2, 7)$, requiring separate trips.



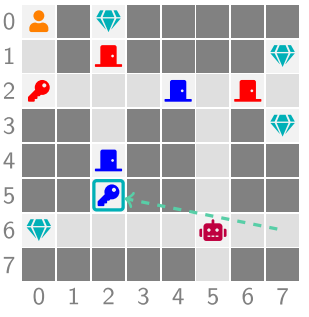
(c) Original. Scenario 10, Grid 20. The blue key sits at $(6, 7)$, two steps from the agent.



(d) **P2, color.** The cyclic color permutation red→blue→yellow→red re-colors keys, doors, and the instruction alike: *"Can you open the blue door there?"*



(e) **P4, easier.** Both keys move to adjacent cells at $(2, 2)$ and $(2, 3)$, so a single leftward sweep collects both.



(f) **P4, harder.** The blue key moves to $(5, 2)$, doubling the agent-to-key distance and landing on the far side of the agent.

Fig. 12. Color permutation and key relocation. The upper row shows the original scenario and the lower row the perturbed counterpart. The permutation preserves the grid structure and optimal plan, whereas relocation alters the optimal action and plan sequences, making a scenario easier or harder.

7.1. Performance and statistical analysis

Table 11 reports all six metrics across the six conditions for each LLM and prompting variant. Table 12 reports the accompanying significance tests.

We evaluate each condition on the same 20 grid–instruction pairs with the same LLM. Consequently, each scenario under a perturbation pairs with that same scenario under the re-run. Therefore, each scenario serves as its own control. Accordingly, we apply Cochran's Q test to the binary metrics and the Friedman test to the continuous metrics. We evaluate each test across the five conditions on $n = 20$ matched scenarios, and report Kendall's W as the effect size.

We ran 36 omnibus tests in total (three LLMs × two variants × six metrics). One test is undefined: plan optimality for the CP variant of Gemma-3-27B. This metric remains identical across all five conditions. No test attains $p < .05$ in any configuration. Because no test attains significance, we omit post hoc comparisons. Among the 35 defined tests, the smallest p-value is 0.267 and the largest Kendall's W is 0.065. These values indicate negligible agreement in how the conditions rank the scenarios. We detail each configuration below.

GPT-4o CP: This configuration ranks in the middle among the three CP configurations. It scores above Gemma-3-27B and below DeepSeek-R1 on every metric. Every perturbation leaves it unaffected.

- **Re-run:** Migrating to the current endpoints leaves five of the six metrics unchanged. Plan optimality falls from 0.250 to 0.200, i.e., exactly one scenario out of 20. Plan feasibility rises by 0.004. The re-run therefore provides a faithful baseline.
- **Perturbations:** The P1–P4 mean departs from the re-run by at most 0.015 on any metric. The P1-P4 mean action optimality remains identical to the re-run. Nonetheless, between 12 and 14 of the 20 scenarios change on at least one metric in every condition. Some scenarios improve over the baseline, whereas others degrade. However, these changes cancel out in the aggregate.
- **Insights:** Two failure modes appear. Padding (P1) produces coordinate-anchoring failures, in which the model reproduces a structurally correct plan whose coordinates belong to the unpadded grid. The distractor (P3) failures on grids 10 and 10b are reachability failures. For instance, the agent collects the key at $(5, 5)$. However, a door blocks this key, making it unreachable to

Table 11
Tomcat's perturbation performance, including the original and re-run baseline. P1–P4 mean remains closely aligned with the re-run for each configuration.

Model & Variant	Condition	Intent Accuracy	Action Feasibility	Action Optimality	Plan Feasibility	Plan Optimality	Instruction Accuracy
	Original	0.500	0.788	0.550	0.609	0.250	0.600
	Re-run	0.500	0.788	0.550	0.613	0.200	0.600
	P1: Padding	0.500	0.771	0.508	0.591	0.150	0.600
GPT-4o	P2: Color	0.450	0.804	0.525	0.604	0.200	0.550
CP	P3: Distractor	0.500	0.733	0.542	0.587	0.200	0.600
	P4: Relocation	0.500	0.788	0.625	0.609	0.200	0.600
	P1–P4 mean	0.488	0.774	0.550	0.598	0.188	0.588
	Original	0.850	0.975	0.925	0.918	0.750	0.750
	Re-run	0.850	0.975	0.875	0.930	0.750	0.750
	P1: Padding	0.800	0.975	0.813	0.878	0.700	0.700
GPT-4o	P2: Color	0.850	0.925	0.829	0.897	0.650	0.650
Fs-CoT	P3: Distractor	0.850	0.967	0.867	0.935	0.750	0.800
	P4: Relocation	0.850	0.950	0.863	0.902	0.700	0.700
	P1–P4 mean	0.838	0.954	0.843	0.903	0.700	0.713
	Original	0.600	0.860	0.735	0.699	0.550	0.550
	Re-run	0.750	0.875	0.725	0.793	0.600	0.650
	P1: Padding	0.750	0.867	0.713	0.780	0.600	0.550
DeepSeek-R1	P2: Color	0.750	0.900	0.746	0.798	0.600	0.700
CP	P3: Distractor	0.750	0.850	0.808	0.781	0.650	0.600
	P4: Relocation	0.700	0.958	0.858	0.822	0.700	0.600
	P1–P4 mean	0.738	0.894	0.781	0.795	0.638	0.613
	Original	0.800	0.885	0.735	0.805	0.700	0.750
	Re-run	0.800	0.943	0.877	0.869	0.800	0.800
	P1: Padding	0.850	0.905	0.850	0.888	0.800	0.850
DeepSeek-R1	P2: Color	0.750	0.958	0.833	0.856	0.700	0.750
Fs-CoT	P3: Distractor	0.850	0.950	0.838	0.889	0.750	0.900
	P4: Relocation	0.900	0.945	0.888	0.908	0.800	0.850
	P1–P4 mean	0.838	0.940	0.852	0.885	0.763	0.838
	Original	0.050	0.604	0.304	0.326	0.050	0.500
	Re-run	0.100	0.675	0.467	0.412	0.050	0.600
	P1: Padding	0.100	0.650	0.450	0.421	0.050	0.600
Gemma-3-27B	P2: Color	0.100	0.704	0.454	0.416	0.050	0.550
CP	P3: Distractor	0.100	0.642	0.367	0.375	0.050	0.500
	P4: Relocation	0.150	0.708	0.483	0.430	0.050	0.600
	P1–P4 mean	0.113	0.676	0.439	0.411	0.050	0.563
	Original	0.300	0.750	0.513	0.498	0.300	0.650
	Re-run	0.400	0.775	0.638	0.655	0.350	0.750
	P1: Padding	0.400	0.755	0.543	0.644	0.300	0.650
Gemma-3-27B	P2: Color	0.450	0.705	0.573	0.631	0.350	0.650
Fs-CoT	P3: Distractor	0.400	0.775	0.638	0.636	0.350	0.650
	P4: Relocation	0.450	0.792	0.679	0.658	0.400	0.750
	P1–P4 mean	0.425	0.757	0.608	0.642	0.350	0.675

the agent. As a result, action feasibility falls to 0.0. The correct action was to collect the reachable yellow key at $(9, 0)$ or $(9, 10)$. Performance changes on individual scenarios are also directional. Scenarios 7 and 13, on grids 13 and 8, improve under all four perturbations. In contrast, scenarios 6 and 11 strictly degrade.

GPT-4o Fs-CoT: This configuration is the strongest overall. It is also the most stable across individual scenarios.

- **Re-run:** The re-run reproduces the published result on four metrics. Plan feasibility rises by 0.012 and action optimality falls by 0.050.
- **Perturbations:** The P1–P4 mean sits at or below the re-run on all six metrics, by at most 0.050. Only five to ten scenarios change per condition. This represents the narrowest movement of any configuration. The exemplars therefore provide stability alongside accuracy. The distractor condition (P3) yields the strongest planning metrics among the perturbations and re-run.
- **Insights:** The single scenario preventing P3 from leading on every metric is scenario 20 on grid 19, where the agent plans to collect a blue key at $(1, 2)$. That cell holds a distractor placed in a region the agent cannot enter. Consequently, action feasibility falls to 0.333. This represents the same reachability failure observed under CP on grids 10 and 10b. However, it now occurs under a different variant, a different grid, and a different color. Scenario 3 on grid 6 improves under perturbations P1–P3. Meanwhile, Scenario 5 on grid 10b improves over the baseline.

DeepSeek-R1 CP: The strongest CP configuration. The perturbations mildly improve its performance.

Table 12
Omnibus repeated-measures tests across the five conditions (the re-run baseline and P1–P4) on $n = 20$ matched scenarios with four degrees of freedom. Because no test attains $p < .05$, we omit post hoc comparisons. Cochran's Q is undefined for plan optimality for Gemma-3-27B CP because the metric is identical across all five conditions.

Model & Variant	Metric	Omnibus Test	Statistic	p	Kendall's W	Post hoc
GPT-4o CP	Intent Accuracy	Cochran's Q	0.308	0.989	0.004	✗
	Plan Optimality	Cochran's Q	0.471	0.976	0.006	✗
	Instruction Accuracy	Cochran's Q	4.000	0.406	0.050	✗
	Action Feasibility	Friedman (χ^2)	0.276	0.991	0.003	✗
	Action Optimality	Friedman (χ^2)	1.818	0.769	0.023	✗
	Plan Feasibility	Friedman (χ^2)	1.175	0.882	0.015	✗
GPT-4o Fs-CoT	Intent Accuracy	Cochran's Q	0.471	0.976	0.006	✗
	Plan Optimality	Cochran's Q	1.037	0.904	0.013	✗
	Instruction Accuracy	Cochran's Q	5.200	0.267	0.065	✗
	Action Feasibility	Friedman (χ^2)	2.286	0.683	0.029	✗
	Action Optimality	Friedman (χ^2)	1.985	0.739	0.025	✗
	Plan Feasibility	Friedman (χ^2)	2.233	0.693	0.028	✗
DeepSeek-R1 CP	Intent Accuracy	Cochran's Q	0.320	0.989	0.004	✗
	Plan Optimality	Cochran's Q	1.143	0.887	0.014	✗
	Instruction Accuracy	Cochran's Q	2.167	0.705	0.027	✗
	Action Feasibility	Friedman (χ^2)	3.240	0.519	0.041	✗
	Action Optimality	Friedman (χ^2)	2.864	0.581	0.036	✗
	Plan Feasibility	Friedman (χ^2)	1.280	0.865	0.016	✗
DeepSeek-R1 Fs-CoT	Intent Accuracy	Cochran's Q	2.737	0.603	0.034	✗
	Plan Optimality	Cochran's Q	1.280	0.865	0.016	✗
	Instruction Accuracy	Cochran's Q	2.261	0.688	0.028	✗
	Action Feasibility	Friedman (χ^2)	1.041	0.904	0.013	✗
	Action Optimality	Friedman (χ^2)	0.437	0.979	0.006	✗
	Plan Feasibility	Friedman (χ^2)	2.485	0.647	0.031	✗
Gemma-3-27B CP	Intent Accuracy	Cochran's Q	0.500	0.974	0.006	✗
	Plan Optimality	Cochran's Q	–	–	–	✗
	Instruction Accuracy	Cochran's Q	3.556	0.470	0.044	✗
	Action Feasibility	Friedman (χ^2)	0.825	0.935	0.010	✗
	Action Optimality	Friedman (χ^2)	1.848	0.764	0.023	✗
	Plan Feasibility	Friedman (χ^2)	3.691	0.449	0.046	✗
Gemma-3-27B Fs-CoT	Intent Accuracy	Cochran's Q	0.333	0.988	0.004	✗
	Plan Optimality	Cochran's Q	0.667	0.955	0.008	✗
	Instruction Accuracy	Cochran's Q	3.000	0.558	0.038	✗
	Action Feasibility	Friedman (χ^2)	1.364	0.851	0.017	✗
	Action Optimality	Friedman (χ^2)	0.221	0.994	0.003	✗
	Plan Feasibility	Friedman (χ^2)	0.892	0.926	0.011	✗

- **Re-run:** The re-run improves over the original results on five of the six metrics. The largest gains appear in intent accuracy (0.600 → 0.750) and plan feasibility (0.699 → 0.793). We attribute this gain to the increased token budget rather than to the model.
- **Perturbations:** Relative to that re-run baseline the perturbations are mildly beneficial. On the P1–P4 mean, action optimality rises by 0.056 and plan optimality rises by 0.038, whereas instruction accuracy falls by 0.037. Between nine and 14 scenarios change per condition, thus the aggregate stability again reflects cancellation rather than invariance.
- **Insights:** The gains concentrate in key relocation (P4) which raises plan optimality from 0.600 to 0.700 and action optimality from 0.725 to 0.858. Unlike GPT-4o, this configuration never commits to a distractor key (P3). Scenarios 12 and 16, on grids 6 and 11, improve under all four perturbations. Scenario 17 on grid 12 degrades under P1 and P4, whereas scenario 20 on grid 19 degrades under P2 and P3.

DeepSeek-R1 Fs-CoT: This configuration performs comparably to GPT-4o Fs-CoT on the planning metrics. It remains equally unaffected by the perturbations.

- **Re-run:** The re-run improves on the original result on five of the six metrics, with action optimality rising from 0.735 to 0.877 and plan optimality from 0.700 to 0.800. Intent accuracy remains unchanged.
- **Perturbations:** The P1–P4 mean moves by at most 0.038 on any metric. It moves upward on instruction and intent accuracy, and downward on plan optimality. Between seven and 12 scenarios change per condition.
- **Insights:** Similar to its CP counterpart, the agent never commits to a distractor key (P3) in this configuration. Scenarios 17 and 18, on grids 12 and 13 improve under all four perturbations, whereas scenario 15 on grid 10b strictly degrades.

Gemma-3-27B CP: This is the weakest configuration in the study on the planning metrics.

- **Re-run:** The re-run improves on the original result on five of the six metrics, with action optimality rising from 0.304 to 0.467. Plan optimality remains unchanged at 0.050, the floor for this configuration.
- **Perturbations:** The P1–P4 mean departs from the re-run by at most 0.037. Between ten and 13 scenarios change per condition. The distractor condition (P3) produces the weakest results. It lowers instruction accuracy by 0.100 and action optimality by 0.100. Key relocation (P4) provides the strongest condition among the perturbations. It increases intent accuracy, plan feasibility, and both action metrics over the baseline.
- **Insights:** P3 induces the highest rate of reachability failures in this configuration. For instance, in scenario 20 on grid 19, the agent commits to the blue key at $(1, 2)$. This cell holds one of the two unreachable distractors. Cochran's Q is undefined for plan optimality because the metric remains identical across all five conditions. In few scenarios, the agent confabulates key locations. It names a key that does not exist at the specified grid coordinates. Scenario 3 on grid 6 improves under all four perturbations, whereas scenario 1 on grid 3 degrades under P1–P3.

Gemma-3-27B Fs-CoT: The exemplars substantially improve this configuration. However, it exhibits the widest movement across individual scenarios of any configuration in the study.

- **Re-run:** The re-run improves on the original result across all six metrics, with plan feasibility rising from 0.498 to 0.655 and action optimality from 0.512 to 0.637.
- **Perturbations:** The P1–P4 mean departs from the baseline by at most 0.075, on instruction accuracy. This represents the largest such departure in the study. Between nine and 16 scenarios change per condition. P4 exhibits the strongest performance, whereas P1 demonstrates the weakest performance relative to the baseline.
- **Insights:** The agent commits to a distractor key (P3) on scenarios 4 and 20. More distinctively, it grounds actions in coordinates drawn from the exemplars rather than the provided grid. This behavior demonstrates game-specific pattern memorization. An action of the form *collect* ⟨*color*⟩ *key at* (r, c) names a cell holding no such key in three of the 20 scenarios under color permutation (scenarios 2, 16, and 18) and key relocation (scenarios 3, 10, and 12). In contrast, this error never occurs under the re-run. The same measure is zero for GPT-4o and DeepSeek-R1. This memorization behavior is therefore specific to the smaller model.

7.2. Perturbation takeaways

The perturbation conditions P1–P3 show that performance does not depend on the specific coordinates, color words, or object counts in the scenarios. No configuration's P1–P4 mean departs from its own re-run by more than 0.075 on any metric, and no omnibus test approaches significance. P4 demonstrates that this stability does not arise from the models ignoring their environment. By construction, P4 alters the optimal action and plan sequences, and performance shifts appropriately in response. Without P4, the invariance in P1–P3 could simply describe a model that ignores the grid entirely.

The perturbations also preserve the ranking between prompting variants. Fs-CoT exceeds CP on all 30 metric-by-condition comparisons for GPT-4o and Gemma-3-27B. It also exceeds CP on 28 of 30 comparisons for DeepSeek-R1. For DeepSeek-R1, intent accuracy ties under P2, and action feasibility drops by 0.013 under P4. The Fs-CoT advantage established in Section 6 therefore survives every perturbation across every LLM.

Gemma-3-27B presents a partial exception on individual scenarios. Although its omnibus tests show no significance, it exhibits specific qualitative failures. Gemma-3-27B suffers the highest number of reachability failures. In these cases, the agent attempts to retrieve unreachable keys. This failure occurs more frequently under CP than Fs-CoT. Further, Gemma-3-27B with Fs-CoT sometimes generates actions directly from the exemplars. This causes the agent to attempt to collect keys from empty coordinates, demonstrating game-specific pattern memorization. This confabulation occurs across four to eight scenarios for Gemma-3-27B with Fs-CoT, and occasionally under its CP counterpart. GPT-4o and DeepSeek-R1 do not exhibit this sensitivity. Therefore, the perturbation results suggest these errors stem from the underlying models rather than the prompting strategy.

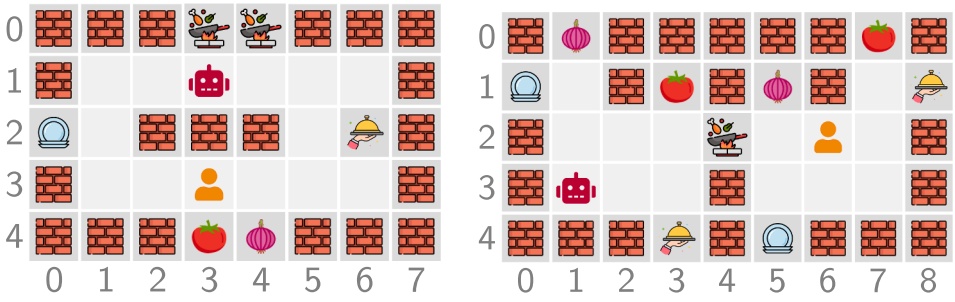
8. Environment generalizability

The perturbation study (Section 7) demonstrates that both Tomcat variants remain robust to environmental variations. This suggests that reliance on game-specific patterns stems from the underlying models rather than the framework. Intra-environment perturbations alone cannot establish whether Tomcat generalizes to entirely distinct domains. To address this, we implement the Instruction Inference task in a second environment, *Overcooked*, and evaluate our framework within this new context. Crucially, the instruction taxonomy, the six evaluation metrics, the two prompting variants, and the underlying prompt skeleton remain strictly unchanged to ensure a robust comparison.

8.1. The Overcooked environment

Overcooked is a cooperative multiagent domain [48]. Two agents share a kitchen to fulfill pending soup orders. We evaluate Tomcat's ToM capabilities across seven layouts adapted from Carroll et al. [48]. Fig. 13 shows two of the seven layouts. Appendix B shows all seven. The objects in this environment include pots, serving stations, counters, dish dispensers carrying plates, and ingredient dispensers carrying ten tomatoes or onions.

The interaction model matches the Doors, Keys, and Gems environment. A principal moves and issues an instruction. The agent must infer the intent and act. To complete a pending soup order, the agents must place the necessary ingredients into the pot. Cooking



(a) **Counter Circuit.** A central counter block divides the kitchen. Both agents can reach every object. Navigation requires detouring around the central block.

(b) **Mirrored Halves.** A central counter divides the kitchen. Each half houses a complete set of stations, thus neither agent has to depend on the other.

Fig. 13. Two of the seven Overcooked kitchens. The brick cells ( ) are counters, which cannot be walked on but can hold an item; light cells are floor. Icons are the agent ( ), the principal ( ), pots ( ), onion ( ) and tomato ( ) dispensers, the dish dispenser ( ), and the serving station ( ). Coordinates are given as (row, column) with a top-left origin, exactly as the agent receives them.

starts automatically once the required ingredients are placed. After cooking finishes, an agent retrieves a plate from the dish dispenser. The agent plates the soup and delivers it to a serving station. Unlike the Doors, Keys, and Gems environment, agents cannot occupy the same cell. Narrow corridors create physical bottlenecks. Agents must coordinate paths and yield right-of-way to avoid blocking each other. Three structural differences make this a distinct test rather than a simple relabeling:

The principal acts: Both parties move, carry, and hand items to each other. A plan therefore represents a partially ordered sequence over two actors rather than a linear sequence over one.

The goal is not in the grid: A kitchen looks identical whether the pending order requires one onion or three tomatoes. The order queue arrives separately. The agent must reconcile this queue with the principal's instruction. Consequently, an instruction can remain incomplete or ambiguous in ways the physical layout alone cannot resolve.

Regions can be disjoint: Several layouts physically partition the agents. Agents may transfer items over shared counters. The principal's position dictates which counter to use. In Fig. 13b, the agent can place an object at location $(3, 4)$. The principal can then retrieve it. Fig. B.1b in the appendix illustrates the extreme case of physical separation.

The instruction distribution follows Section 3.3 and Table 2. It contains eight clear and 12 unclear instructions. The unclear instructions split evenly between incomplete and ambiguous. This matches the proportions of the Doors, Keys, and Gems corpus. The interpretation pipeline in Fig. 2 remains unchanged. The single substantive change is the output format. Tomcat now answers in JSON rather than free text. We detail this modification next.

8.2. Framework

The prompt retains the component order of Section 4: 1) common ground, 2) response generation, and 3) demonstration exemplars. Only the response generation component changes. The action vocabulary now comprises six verbs—*Get*, *Place*, *Pass*, *Put*, *Plate*, and *Deliver*. These apply to four items (*onion*, *tomato*, *dish*, *soup*) and eight object types: onion dispenser, tomato dispenser, dish dispenser, pot, serving station, counter, human, and agent.

The response generation component now specifies a JSON format (Fig. 14). This replaces the free-text template of Fig. 5. The JSON object carries the same four fields the original study elicited in prose. The prompt asks nothing new of the model. The structured format simply enables automated annotation. Emitting valid JSON is a capability that varies across models. We therefore report format compliance alongside the evaluation metrics.

CP supplies the same two problem walkthroughs. These step through the initial, observed, and completed kitchen states exactly as in Fig. 6. **Fs-CoT** supplies seven input–output exemplars and the four additional reasoning rules of Section 4.

8.3. Experiments

We first develop a corpus using seven layouts (Fig. B.1 in the appendix). We then select four frontier reasoning models to evaluate alongside the three models from the Doors, Keys, and Gems domain. We evaluate Tomcat's ToM capabilities using the same intent, action, and planning metrics (Section 5.2).

Corpus construction: We develop 20 scenarios over the seven layouts. Five are standard Overcooked-AI configurations. We add two layouts to test mechanisms the standard set lacks: strict complementarity (Fig. B.1b) and a single-tile choke point (Fig. B.1c). The set spans the three sources of underspecification the task requires. First, one shared region with a single set of stations (Fig. B.1a and f).

Response Generation: Overcooked

Based on your understanding of the game rules, current kitchen layout, pending order(s), human actions, instruction…delimited by triple backticks:

```
‘‘‘
Pending order(s): 3-onion soup
Human Action: The human moves right from (2,6) to (2,7), toward the serving station, and
gives the instruction.
Instruction: Grab one from the dispenser and get it cooking.
‘‘‘
```

Return a SINGLE JSON object and nothing else. Put it between the two markers below, each on its own line, and write nothing after the closing marker.

```
###ANSWER-START###
{
"type": "Clear or Unclear",
"type_reasoning": "<two or three sentences …>",
"intent": "<two or three sentences …>",
"intent_target": {
"order": "<single order being served now, e.g., 2-onion soup>",
"ingredients": ["<every ingredient that goes into the pot>"],
"pot": [[row, col]], "serving": [[row, col]]
},
"response": "<two or three sentences of reasoning …>",
"actions": [{"actor": "agent|human", "verb": "Get|Place|Pass|Put|Plate|Deliver",
"item": "onion|tomato|dish|soup", "object": "O|T|D|P|S|X|h|m",
"row": …, "col": …}]
}
###ANSWER-END###
```

Fig. 14. Excerpt of the response generation component for Overcooked.

Second, duplicated stations where a deictic reference does not resolve (Fig. B.1g). Third, divided kitchens where a hand-off counter must be inferred (Figs. 13b and B.1b). Fig. 13a introduces a fourth condition. An interior counter block creates narrow pathways. Here, travel cost and collision avoidance, rather than reachability, dictate sub-task allocation. Agents must coordinate routes to prevent blocking each other. Ground truth is authored by hand and checked programmatically for object identity, region reachability, and a carry rule tracking what each agent holds. When two plans complete an order equally well, we record both. This ensures a model is not penalized for choosing an equivalent solution.

Models and protocol: We evaluate seven models. Three carry over from the original study: GPT-4o, DeepSeek-R1, and Gemma-3-27B. We access GPT-4o via the OpenAI API. We access DeepSeek-R1 and Gemma-3-27B via OpenRouter using the Novita and DeepInfra providers, respectively. We cap DeepSeek-R1 at 16,000 output tokens, the maximum limit supported by Novita. We introduce four frontier models for this experiment: *Claude Opus 5*, *GPT-5.6 Sol*, *Gemini 3.1 Pro*, and *DeepSeek V4 Pro*. We set the reasoning parameter to the highest available level for each new model. Each model answers all 20 scenarios under both variants. This yields 280 total samples spanning 2.03 M prompt tokens and 834 K completion tokens.

Annotation: The six metrics match those in Section 5.2 and are computed automatically. Every predicted action falls into exactly one of four buckets. An action is *matched* if it perfectly aligns with the ground truth. It is *inefficient* if it reaches the same goal through an interchangeable object. It is *incorrect* if it is executable but does not advance the goal. It is *infeasible* if it fails the grid, region, or carry check. Let $D = |A_{\text{groud-truth}}| + |A_{\text{incorrect}}| + |A_{\text{infeasible}}|$. Feasibility is the proportion of D that executes. Action optimality is the proportion that matches. A missing action is not added to D because the ground truth already counts it. An inefficient but feasible action is excluded from D. As in Section 4, *Pass* falls inside the plan bracket and outside the action bracket. A prediction is scored against whichever accepted plan it matches. Because both agents act concurrently, a re-ordered sequence that still executes is scored as optimal. All 280 outputs parsed on the first attempt. Consequently, no result below is confounded by format drift.

8.4. Overcooked results

Table 13 reports the six metrics for every model under each variant. Each cell represents a mean over the same 20 scenarios. Both variants answer the same scenarios with the same model. Consequently, each scenario serves as its own control. Because we compare only two groups, we use the two-group paired procedures defined in Section 6.1. We use McNemar's exact test with Cohen's g for the binary metrics (Table 14). We use the Wilcoxon signed-rank test with rank-biserial r for the continuous metrics (Table 15).

Instruction Inference transfers: Six of the seven models achieve at least 0.90 intent accuracy under Fs-CoT. The Fs-CoT variant of Claude Opus 5 and Gemini 3.1 Pro attain perfect action feasibility and action optimality. Claude Opus 5 with Fs-CoT matches an

Table 13
Tomcat performance on Overcooked using CP and Fs-CoT variants across seven models ($n = 20$ scenarios).

Model	Variant	Intent Accuracy	Action Feasibility	Action Optimality	Plan Feasibility	Plan Optimality	Instruction Accuracy
Claude Opus 5	CP	0.950	0.943	0.927	0.963	0.900	1.000
	Fs-CoT	1.000	1.000	1.000	1.000	1.000	0.950
Gemini 3.1 Pro	CP	1.000	1.000	1.000	0.984	0.900	0.950
	Fs-CoT	1.000	1.000	1.000	0.994	0.950	0.950
DeepSeek V4 Pro	CP	1.000	0.980	0.960	0.981	0.850	1.000
	Fs-CoT	1.000	0.990	0.975	0.977	0.850	0.950
DeepSeek-R1	CP	0.950	0.957	0.906	0.955	0.800	0.950
	Fs-CoT	1.000	1.000	0.950	0.988	0.900	0.900
GPT-5.6 Sol	CP	0.900	0.958	0.858	0.944	0.750	0.950
	Fs-CoT	1.000	0.994	0.915	0.948	0.700	0.950
GPT-4o	CP	0.850	0.896	0.854	0.876	0.600	0.650
	Fs-CoT	0.900	0.941	0.868	0.829	0.550	0.600
Gemma-3-27B	CP	0.650	0.539	0.467	0.482	0.250	0.400
	Fs-CoT	0.850	0.740	0.623	0.576	0.250	0.700

Table 14
Binary metrics on Overcooked, McNemar's exact test with Cohen's g, paired on the same 20 scenarios. A positive g favours Fs-CoT. Markers follow Table 10.

Model	Metric	p	Effect	CP vs. Fs-CoT
Claude Opus 5	Intent Accuracy	1.000	$g = +1.00$	†
	Plan Optimality	0.500	$g = +1.00$	†
	Instruction Accuracy	1.000	$g = -1.00$	†
Gemini 3.1 Pro	Intent Accuracy	1.000	$g = +0.00$	✗
	Plan Optimality	1.000	$g = +1.00$	†
	Instruction Accuracy	1.000	$g = +0.00$	✗
DeepSeek V4 Pro	Intent Accuracy	1.000	$g = +0.00$	✗
	Plan Optimality	1.000	$g = +0.00$	✗
	Instruction Accuracy	1.000	$g = -1.00$	†
DeepSeek-R1	Intent Accuracy	1.000	$g = +1.00$	†
	Plan Optimality	0.688	$g = +0.33$	†
	Instruction Accuracy	1.000	$g = -0.33$	†
GPT-5.6 Sol	Intent Accuracy	0.500	$g = +1.00$	†
	Plan Optimality	1.000	$g = -0.20$	✗
	Instruction Accuracy	1.000	$g = +0.00$	✗
GPT-4o	Intent Accuracy	1.000	$g = +1.00$	†
	Plan Optimality	1.000	$g = -0.14$	✗
	Instruction Accuracy	1.000	$g = -1.00$	†
Gemma-3-27B	Intent Accuracy	0.289	$g = +0.50$	†
	Plan Optimality	1.000	$g = +0.00$	✗
	Instruction Accuracy	0.146	$g = +0.50$	†

accepted plan on all 20 scenarios. However, the evaluation does not hit a performance ceiling. Models with less reasoning capacity, such as Gemma-3-27B, still struggle to resolve unclear instructions and synthesize optimal plans. This gap proves the metrics do not saturate. They successfully separate frontier capabilities from baseline models across different instruction types. Testing solely in the original gridworld could not have demonstrated this robust, cross-domain transferability.

The taxonomy dictates task difficulty: Nearly every model handles clear instructions well. The unclear subtypes separate model performance. Both GPT-4o variants classifies all 12 incomplete scenarios as clear. However, it correctly classifies 0.94 of the genuinely clear scenarios, matching Claude Opus 5. On six of those 12 incomplete scenarios, GPT-4o's plan is nonetheless correct. It recovers the unstated ingredient count from the order queue. It simultaneously asserts the instruction is complete. This separates task competence from the recognition of an incomplete instruction.

No variant difference reaches significance: None of the 42 comparisons in Tables 14 and 15 yields $p < .05$. However, overall aggregate performance favors the Fs-CoT variant for five of the seven models, as demonstrated in Table 16 via mean gaps and metric win rates. This narrow performance margin represents a weaker separation than observed in the original study. The CP baseline is far stronger in this new domain. For example, plan optimality for GPT-4o with CP rises from 0.250 in Doors, Keys, and Gems to

Table 15
Continuous metrics on Overcooked, Wilcoxon signed-rank test with rank-biserial r, paired on the same 20 scenarios. A positive r favours Fs-CoT. Markers follow Table 10.

Model	Metric	p	Effect	CP vs. Fs-CoT
Claude Opus 5	Action Feasibility	0.180	$r = +0.10$	✗
	Action Optimality	0.180	$r = +0.10$	✗
	Plan Feasibility	0.180	$r = +0.10$	✗
Gemini 3.1 Pro	Action Feasibility	1.000	$r = +0.00$	✗
	Action Optimality	1.000	$r = +0.00$	✗
	Plan Feasibility	0.317	$r = +0.05$	✗
DeepSeek V4 Pro	Action Feasibility	0.655	$r = +0.00$	✗
	Action Optimality	0.715	$r = +0.09$	✗
	Plan Feasibility	0.686	$r = -0.05$	✗
DeepSeek-R1	Action Feasibility	0.180	$r = +0.10$	✗
	Action Optimality	0.596	$r = +0.09$	✗
	Plan Feasibility	0.168	$r = +0.11$	✗
GPT-5.6 Sol	Action Feasibility	0.655	$r = +0.00$	✗
	Action Optimality	0.465	$r = +0.06$	✗
	Plan Feasibility	1.000	$r = -0.03$	✗
GPT-4o	Action Feasibility	0.180	$r = +0.05$	✗
	Action Optimality	0.596	$r = +0.00$	✗
	Plan Feasibility	0.398	$r = -0.07$	✗
Gemma-3-27B	Action Feasibility	0.126	$r = +0.25$	✗
	Action Optimality	0.256	$r = +0.21$	✗
	Plan Feasibility	0.562	$r = +0.17$	✗

Table 16
Summary of Fs-CoT impact by model on the Overcooked domain. Mean Gap is the average difference across all six metrics, calculated as $\frac{1}{6}\sum$(Fs-CoT − CP). Up/Flat/Down indicates the number of metrics (out of six) where Fs-CoT outperformed, tied, or underperformed CP.

Model	Reasoning	Mean Gap	Up ▲	Flat =	Down ▼	Verdict
Gemma-3-27B	✗	0.158	5	1	0	Fs-CoT clearly helps
Claude Opus 5	✓	0.044	5	0	1	Fs-CoT helps mildly
DeepSeek-R1	✓	0.037	5	0	1	Fs-CoT helps mildly
GPT-5.6 Sol	✓	0.025	4	1	1	Fs-CoT helps mildly
Gemini 3.1 Pro	✓	0.010	2	4	0	No real difference
DeepSeek V4 Pro	✓	−0.005	2	2	2	No real difference
GPT-4o	✗	−0.006	3	0	3	No real difference

0.600 in Overcooked. This stronger baseline leaves less headroom for the exemplars to improve performance. Detecting these smaller, incremental gains would require more statistical power than the 20 scenarios afford. However, the two carried-over models for which the original study showed a positive effect do reproduce it. Gemma-3-27B Fs-CoT improves on five of six metrics, and DeepSeek-R1 Fs-CoT improves by +0.100 on plan optimality.

8.5. Insights

This section distills the broader takeaways from our evaluation in the Overcooked domain. Table 16 summarizes the aggregate impact of the prompting variants and highlights three key findings.

The result is not an artifact of one gridworld: The task, the taxonomy, and the six metrics successfully transfer to a new environment. This environment features a different object vocabulary, a different action grammar, an acting principal, and an implicit goal. Combined with the perturbation study, this proves a key point. The framework relies neither on a specific surface encoding nor on a single gridworld domain.

Model generation matters more than prompting strategy here: The performance spread across models is much larger than the spread between prompting variants for any single model. For example, with CP, plan optimality ranges from 0.250 to 0.900 depending on the model. Excluding Gemma-3-27B, no model exhibits a mean gap larger than 0.05 between variants. In this domain, the choice of model dictates performance more than the prompting strategy.

Longer deliberation is not what separates the models: Output length does not correlate with performance. Gemini 3.1 Pro averages roughly 760 tokens—comparable to Gemma-3-27B—yet achieves perfect action optimality. In contrast, DeepSeek V4 Pro averages roughly 19,000 tokens, twenty-five times more than Gemini 3.1 Pro, yet does not score higher. Neither model hit its token ceiling, ruling out truncation. Ultimately, model performance is driven by inference quality rather than deliberation length.

Table 17
Optimization (abbreviated as "Opt.") objectives, signals and the effect of Fs-CoT across all evaluated LLMs.

Model	Opt. Objective	Opt. Signal	Effect of Fs-CoT
DeepSeek-R1	Self-Discovery	Outcome-based	Neutral (Reasoning Interference)
GPT-4o	Alignment	Preference-based	Positive (Instruction Alignment)
Gemma-3-27B	Simulation	Distillation-based	Positive (Pattern Activation)

9. Theoretical assumptions of LLMs and Fs-CoT

To interpret the performance variations observed across the LLMs, we outline theoretical assumptions grounded in their respective post-training optimization objectives and reasoning behaviors.

9.1. Post-training regimes of the three named systems

This section analyzes the three LLMs from the original study. For each model, we isolate its specific post-training objective and use it to predict how the system will respond to reasoning exemplars.

DeepSeek-R1: Self-Directed Reasoning through GRPO. DeepSeek-R1 is optimized via Reinforcement Learning (Group Relative Policy Optimization) [49] using outcome-based rewards. It is trained to converge on an internally derived reasoning trajectory. Guo et al. [49] find that introducing few-shot exemplars acts as *contextual noise* for reasoning-intensive tasks, forcing the model to reconcile its optimized internal path with an external template. This likely creates a "reasoning collision" that degrades performance, suggesting that zero-shot prompting may act as a practical performance ceiling for DeepSeek-R1. However, our empirical results offer a more nuanced perspective. Although we observed no statistical differences between the Fs-CoT and CP variants, the Fs-CoT variant showed moderate to substantial absolute gains across all metrics, except *action feasibility*. These findings suggest that the claim by Guo et al. [49]—"Few-shot prompting consistently degrades its performance"—may not hold universally in complex, task-oriented environments like instruction inference, as we observed no performance degradation using Fs-CoT.

GPT-4o: Structural Alignment via RLHF. Conversely, GPT-4o exhibits a direct correlation between performance and prompt alignment. As a product of dense Supervised Finetuning (SFT) and Reinforcement Learning from Human Feedback (RLHF) [50], GPT-4o is strongly optimized for instruction following. It treats the few-shot context as a command structure or a formatting instruction, which aligns with its training to act as a mimicking assistant. The exemplars serve as guiding constraints that effectively reduce the search space for an optimal response [51]. Consistent with this, we observed that GPT-4o achieved the highest scores across all metrics when provided with Fs-CoT prompting over its counterpart.

Gemma-3-27B: Simulation via Knowledge Distillation. Gemma's performance is likely driven by teacher-student mimicry. Using knowledge distillation from larger models (e.g., Gemini), its reward functions [52] encourage behavior that approximates the reasoning patterns of the "Teacher" model. Unlike DeepSeek-R1's self-discovery, Gemma is optimized to replicate high-quality reasoning patterns provided during post-training. Consequently, Fs-CoT prompting succeeds by providing the teacher-like context Gemma was specifically trained to simulate. In this study, we observed the success of the Gemma-3-27B Fs-CoT variant over its CP variant, suggesting that explicit reasoning exemplars effectively activate these distilled capabilities.

Table 17 breaks down the optimization objectives and impact of using Fs-CoT for the Door, Keys, and Gems domain.

9.2. Generalization beyond the three named systems

The three preceding accounts describe specific models, limiting their predictive power for unseen systems. We therefore reframe them as claims about post-training paradigms rather than how specific models behave. Each paradigm dictates how a model utilizes an exemplar, a property directly tied to its training signal.

Under *outcome-reward reinforcement learning*, a model receives rewards for reaching a correct final state, leaving it free to discover its own path. Exemplars therefore constrain a previously optimized trajectory, yielding a neutral to mildly negative predicted effect for Fs-CoT. Under *preference alignment*, a model receives rewards for producing human-preferred responses, making it highly sensitive to demonstrated form. It treats exemplars as strict specifications, yielding a positive predicted effect. Under *knowledge distillation*, a model learns to reproduce a larger teacher's reasoning. Exemplars supply the exact teacher-like context it was trained to imitate. The predicted effect is positive and largest for small models lacking the capacity to reconstruct that context unaided.

The Overcooked experiment introduces four unseen models to test these hypotheses. Table 18 records the dominant paradigm, the expected effect, and the observed mean gap between CP and Fs-CoT.

Six of the seven models behave as predicted. GPT-4o is the exception, but it provides valuable insights into why this happens. The alignment paradigm predicted a large gain for GPT-4o, which it delivered in the original gridworld. However, its mean gap in Overcooked is -0.006, essentially zero. This does not invalidate the paradigm hypothesis, but rather exposes its limit. The hypothesis

Table 18
Post-training paradigm, the predicted effect of Fs-CoT, and the observed mean gap from CP to Fs-CoT across the six metrics in Overcooked.

Model	Dominant paradigm	Predicted effect	Observed	Agrees?
Gemma-3-27B	Knowledge Distillation	Positive, large	+0.158	Yes
Claude Opus 5	Alignment + reasoning	Positive, small	+0.044	Yes
DeepSeek-R1	Outcome-reward RL	Neutral	+0.037	Yes
GPT-5.6 Sol	Alignment + reasoning	Positive, small	+0.025	Yes
Gemini 3.1 Pro	Alignment + reasoning	Positive, small	+0.010	Yes
DeepSeek V4 Pro	Outcome-reward RL	Neutral	−0.005	Yes
GPT-4o	Preference alignment	Positive	−0.006	Attenuated

predicts how a model responds to exemplars, whereas it cannot account for available performance headroom. GPT-4o establishes a far stronger CP baseline in Overcooked, meaning this attenuation reflects a lack of headroom rather than a paradigm contradiction.

Two further observations qualify this framework. First, the two DeepSeek models, trained under the same paradigm one generation apart, move in the identical direction. This demonstrates a paradigm prediction holding across a generational boundary. Second, modern training paradigms are increasingly mixed. Current-generation systems combine preference alignment with explicit reasoning phases. Their small positive movements reflect a model that accepts a demonstrated format, whereas it still protects its own optimized trajectory. Therefore, we only predict if exemplars will yield a positive effect, rather than how large that effect will be. As explicit reasoning phases become standard, the behavioral differences between outcome-reward and alignment paradigms may eventually disappear.

9.3. Surface sensitivity and data contamination

LLM reasoning is often criticized for being sensitive to surface details or for suffering from data contamination [46]. Our perturbation study addresses both concerns simultaneously. Conditions P1–P3 shift coordinates, permute colors, and add irrelevant (distractor) keys. These perturbations alter the surface formatting and break any exact memorized instruction–grid pairings. Simple data recall or surface-level pattern matching would predict severe degradation under these modifications. However, performance remains statistically indistinguishable from the baseline, with Kendall's W remaining below 0.07 (Section 7.1). To definitively rule out environment-specific advantages, Section 8 removes the original environment altogether. The Overcooked domain introduces a completely different object vocabulary, action grammar, and visual representation. Despite this domain shift, the framework successfully transfers. Our metrics continue to effectively separate both models and instruction types.

10. Discussion

In Doors, Keys, and Gems, our evaluation demonstrated that Fs-CoT, an instantiation of Tomcat that integrates structured reasoning exemplars, consistently outperformed its CP counterpart across multiple LLMs on most metrics, highlighting the benefits of exemplar-based ToM reasoning for instruction understanding. Across instruction identification, action, and planning metrics, Fs-CoT improved Tomcat's ability to model shared intentions and respond appropriately. The Fs-CoT variants of GPT-4o and DeepSeek-R1 achieved performance comparable to that of human participants, particularly in intent recognition and planning, where reasoning about the principal's future steps is crucial.

Notably, the DeepSeek-R1 CP variant performed closely to both its Fs-CoT counterpart and the participants. This is largely owing to the DeepSeek-R1 model being aligned to reason independently in its post-training phase. For Gemma-3-27B, Fs-CoT yielded substantial improvements over CP, but its performance remained below human levels. This gap likely stems from its smaller model capacity, which may constrain its ability to generalize from exemplars and reason about complex, multistep tasks. These findings confirm the value of embedding structured ToM exemplars in collaborative agents and highlight the importance of model capacity in supporting Instruction Inference.

Reimplementing Instruction Inference in Overcooked (Section 8) reinforces our central claim. The task, taxonomy, and metrics successfully transfer to a domain with a different vocabulary, action grammar, acting principal, and implicit goals. Together with the perturbation study, this confirms the framework's robustness against both surface-level variations and complete domain shifts.

Our secondary claim regarding Fs-CoT requires qualification. Although Fs-CoT's advantage was substantial in Doors, Keys, and Gems, it lacks statistical significance in Overcooked. However, the trend still favors Fs-CoT in five of the seven models. This attenuated effect is driven by a much stronger CP baseline; for instance, GPT-4o's plan optimality under CP rises from 0.250 to 0.600. Thus, exemplar-based prompting provides the most benefit where the baseline leaves sufficient headroom, and the magnitude of the gain depends on both the environment and the model.

Finally, two observations shape how we compare these systems. First, model choice outweighs prompting strategy: plan optimality under CP varies widely across models (from 0.250 to 0.900), whereas prompting variants alter performance by less than 0.05 for nearly all models. Second, deliberation length does not predict success. Output token counts vary by over an order of magnitude without tracking performance, confirming that inference quality, rather than sheer verbosity, is what truly separates these systems.

10.1. Threats to validity

Despite the promising results, some factors may limit the generalizability of our findings. First, a small number of participants appeared to misunderstand the task requirements. Although such a variation is expected in user studies and reflects real-world variance, it may have introduced noise in our human baseline. Second, only two annotators evaluated the LLM completions. Although they followed a strict rubric and the criteria were objective, involving additional annotators could have reduced potential bias and increased confidence in the evaluation results.

10.2. Directions in addressing limitations

We identify some limitations that open avenues for further investigation.

Participant sampling. The study involved 52 participants, divided into two groups to balance cognitive load. Expanding to more diverse populations and ensuring exposure to all instructions would improve reliability.

Domain diversity. Our experiment is limited to two domains of human-agent collaboration. Future work can benefit from incorporating diverse domains of human-agent collaboration.

Scale of the environment. Our obstruction heuristics, ρ_{agg} and β_{agg}, are scale-invariant because they compute proportional detours and corridor densities rather than absolute distances. A large grid with one straight, unobstructed corridor yields $\rho_{\text{agg}} = 1.00$ and a low β_{agg}, making it indistinguishable from a small open room. Further, we compute both heuristics along the shortest path to a target key. They presuppose the goal and measure the cost of executing a plan rather than choosing one. Therefore, our claim that instruction ambiguity drives performance more than spatial obstruction applies only within the range this corpus spans: grids from 7×9 to 12×13 with two to five keys. We maintain the environment in a regime where pathfinding is not the binding constraint. This ensures the variation in performance stems from instruction interpretation. A substantially larger environment (e.g., a 30×30 grid) would likely degrade planning. Such degradation could stem either from misinterpreting the instruction or from failing to search the expanded space. Future work could separate these factors by extending the corpus to larger grids and employing scale-sensitive obstruction measures.

Alternative techniques. Beyond exemplars, other ToM enhancement strategies such as reward-based finetuning, chain-of-reasoning distillation, or belief-tracking modules, remain unexplored in our setup.

10.3. Directions in addressing opportunities

Our results offer a promising outlook on human-agent collaboration. Tomcat with Fs-CoT demonstrates the potential of AI agents to serve as capable collaborators by accurately interpreting incomplete or ambiguous instructions, inferring human intent, and generating feasible and optimal actions and plans, often on par with human participants. These findings motivate future work on extending Instruction Inference by incorporating pragmatics to address a broader spectrum of instruction types, including nonverbal cues, humor, sarcasm, irrelevance, invalidity, and incompleteness. Further, to ensure robust generalization across LLMs of varying capacities, especially lower capacity models like Gemma-3-27B, future research should explore model-specific adaptations and scaling strategies.

The development of such capabilities aligns with the projected evolution of LLMs from transactional chatbots to relational cognitive agents. As post-training scaling laws begin to emphasize multistep reasoning, agentic planning, and metacognition, future architectures are expected to incorporate explicit recursive reasoning modules [53–55] for self-reflection and reinforcement learning strategies for finetuning ToM reasoning [56]. Such advancements will likely enable even smaller-scale models to move beyond literal pattern matching toward functional ToM, where they can maintain persistent models of a user's mental state across extended collaborative sessions. Additionally, our findings regarding DeepSeek-R1 invite a re-evaluation of the Fs-CoT strategy. Contrary to the developers' claims that few-shot prompting degrades performance, we observed no such degradation, suggesting that DeepSeek-R1's reasoning capabilities may be more robust to contextual examples than previously reported. Finally, we aim to evaluate Tomcat in real-world collaborative domains featuring richer dialogue, multiagent coordination, and dynamic environments.

10.4. Reproducibility

The code, dataset, task description, and results for our experiments are available at: https://github.com/fardinsaad/Tomcat-LLM.

CRediT authorship contribution statement

Fardin Saad: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Pradeep K. Murukannaiah:** Writing – review & editing, Visualization, Validation, Supervision, Methodology, Formal analysis, Data curation, Conceptualization; **Munindar P. Singh:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Munindar P. Singh reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was partially supported by the National Science Foundation (grant IIS-2116751).

Data availability

In the Reproducibility section (Section 10.4), we state that the complete experimental code, dataset, and results are available via the provided GitHub repository link.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.artint.2026.104621.

Appendix A. Doors, Keys, and Gems prompts

A.1. Common ground component

This component establishes the shared task knowledge available to both Tomcat variants. Fig. A.1 outlines the background and chain-of-thought framing, and Fig. A.2 specifies the objectives and the rules of the game.

General Chain-of-Thought and Background

You assist a human in a cooperative planning domain called Doors, Keys, and Gems, set in a gridworld. The human attempts to retrieve a specific gem, and you assist the human. You help by unlocking doors using the correct keys or passing the keys to the human. Think of it as a team game where a human gives callouts to their teammate (you), expecting collaboration to achieve their goal. The challenge for you is correctly understanding human instructions, which may require some inferencing due to a lack of specificity in the instructions.

Background: This collaborative game is played in a grid containing the following objects: keys, doors, gems, walls, empty spaces, a human, and you. The colors of keys and doors can be red, blue, or yellow.

Figure 1: Grid Configuration

```
[['.' '.' '.' 'y' '.' '.' '.' 'b' 'W' 'W' 'W' 'W']
 ['r' 'W' 'W' 'r' 'W' 'W' '.' 'r' 'W' 'W' 'W' 'g']
 ['W' 'W' 'W' 'W' 'W' 'W' 'm' 'W' 'W' 'W' 'W' 'R']
 ['W' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['g' '.' '.' '.' 'B' '.' '.' 'R' '.' '.' '.' '.']
 ['W' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' 'W']
 ['.' '.' '.' 'Y' '.' '.' '.' 'W' 'W' 'W' 'W' 'W']
 ['B' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' 'W']
 ['g' 'W' 'W' 'W' 'W' 'W' 'h' '.' '.' '.' '.' 'g']]
```

Fig. A.1. Common ground component part-1.

Objectives and Rules

Human's objective: The human's goal is to retrieve a specific gem by providing only a single instruction. However, the human's instruction could be ambiguous or lack specificity, adding a challenge for you to correctly infer the gem desired by the human. For instance, consider the configuration in Figure 1. While some gems, like the gem at (9, 11), may be easily accessible (i.e., don't require a key), the human may instead be aiming for a more challenging gem, such as the red gem placed behind two doors, and can provide instruction as "Get the red key." Here, the human is directing you to collect a red key to unlock the red door and get the gem. But the gem is blocked by two red doors. Thus, you must collect and pass two red keys to the human. You need to infer this even when not explicitly stated based on the context (i.e., the movement of the human).

Your objective: Your primary goal is to assist the human in retrieving the desired gem by fetching the necessary keys or unlocking doors while prioritizing minimal movement. Upon receiving the instruction, you infer which gem the human intends to collect and take only the essential actions—either collecting and passing keys or unlocking doors—to enable the human to access the gem as efficiently and quickly as possible. As a result, your movements are optimized to ensure that the number of steps taken is minimal.

Rules of the Game:
- For each grid configuration (problem), the human provides a single instruction called a callout.
- The instructions (callouts) can be of two types: (1) either directing you to collect a key (or keys) and passing them to the human or (2) instructing you to unlock a door (or doors).
- Only the key of the same color as the door can unlock the door. However, once a key is used to unlock a door, it cannot be reused for any other door of the same color. For instance, one red key can only unlock one red door.
- If you collect one or more keys, they are immediately passed to the human, depending on the grid configuration. However, if you are blocked by a door (or doors) and cannot pass the key directly, you will either collect the necessary key(s) to unlock the door or instruct the human to retrieve the key if it is beyond your reach. This ensures you can overcome obstacles and continue assisting the human efficiently.
- If the callout instructs to unlock a door or doors, you must first collect the corresponding key or keys before proceeding to unlock the door (or doors). In these instances, you do not pass the key to the human.
- Gems are randomly placed within the grid.
- You prioritize efficient, optimal, and obstacle-free routes to achieve the human's objective as quickly and effectively as possible. When instructed to collect key(s) or unlock the door(s), you don't gather all the keys or unlock all the doors on the grid. Instead, you perform only the actions necessary to assist the human in retrieving the gem with minimal steps.
- The objects, such as gems, keys, and doors, have no associated cost (i.e., they are all the same). Your primary goal is to acquire the necessary keys or unlock the appropriate doors, ensuring the human can obtain the desired gem most efficiently, i.e., requiring minimal movements to execute the instruction.
- The human's instruction may not always be explicit, requiring you to infer the intended action based on the context and the grid configuration. The instructions may be ambiguous or lacking specificity.

Fig. A.2. Common ground component part-2.

A.2. Response generation component

This component defines the specific problem instance and dictates the required output format. Fig. A.3 illustrates the grid layout and object positions exactly as the agent receives them, whereas Fig. A.4 presents the template used to elicit the instruction type, the natural language response, and the action sequence.

Response Generation: Grid Structure

The Current Grid Layout for the Keys, Doors, and, Gems:

```
[['.' '.' '.' 'y' '.' '.' '.' 'b' 'W' 'W' 'W' 'W']
 ['r' 'W' 'W' 'r' 'W' 'W' '.' 'r' 'W' 'W' 'W' 'g']
 ['W' 'W' 'W' 'W' 'W' 'W' 'm' 'W' 'W' 'W' 'W' 'R']
 ['W' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['g' '.' '.' '.' 'B' '.' '.' 'R' '.' '.' '.' '.']
 ['W' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' 'W']
 ['.' '.' '.' 'Y' '.' '.' '.' 'W' 'W' 'W' 'W' 'W']
 ['B' 'W' 'W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W' 'W']
 ['g' 'W' 'W' 'W' 'W' 'W' 'h' '.' '.' '.' '.' 'g']]
```

Below are the locations of all objects on the grid, displayed as (row, column) coordinates, along with their total count:
My position (Labeled as 'm'): (2, 6)
Human (Labeled as 'h'): (9, 6)
Red keys (Labeled as 'r'): (1, 0), (1, 3), (1, 7) −−> Total Red keys: 3
Yellow key (Labeled as 'y'): (0, 3) −−> Total Yellow key: 1
Blue key (Labeled as 'b'): (0, 7) −−> Total Blue key: 1
Red doors (Labeled as 'R'): (2, 11), (4, 7) −−> Total Red doors: 2
Yellow door (Labeled as 'Y'): (7, 3) −−> Total Yellow door: 1
Blue doors (Labeled as 'B'): (4, 4), (8, 0) −−> Total Blue doors: 2
Gems (Labeled as 'g'): (1, 11), (4, 0), (9, 0), (9, 11) −−> Total Gems: 4
Walls (Labeled as 'W'): (0, 8), (0, 9), (0, 10), (0, 11), (1, 1), (1, 2), (1, 4), (1, 5), (1, 8), (1, 9), (1, 10), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 7), (2, 8), (2, 9), (2, 10), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 7), (3, 8), (3, 9), (3, 10), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 7), (5, 8), (5, 9), (5, 10), (5, 11), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 7), (6, 8), (6, 9), (6, 10), (6, 11), (7, 7), (7, 8), (7, 9), (7, 10), (7, 11), (8, 1), (8, 2), (8, 3), (8, 4), (8, 5), (8, 7), (8, 8), (8, 9), (8, 10), (8, 11), (9, 1), (9, 2), (9, 3), (9, 4), (9, 5) −−> Total Walls: 73
Empty spaces (Labeled as '.'): (0, 0), (0, 1), (0, 2), (0, 4), (0, 5), (0, 6), (1, 6), (3, 6), (3, 11), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 8), (4, 9), (4, 10), (4, 11), (5, 6), (6, 6), (7, 0), (7, 1), (7, 2), (7, 4), (7, 5), (7, 6), (8, 6), (9, 7), (9, 8), (9, 9), (9, 10) −−> Total Empty spaces: 31

Fig. A.3. Response generation component part-1.

Response Generation: Type, Response, Actions

Based on your understanding of the game rules, current grid layout, human actions, instruction, and the labeled examples provided below, generate the instruction type (clear or unclear), response, and optimal actions for the following human actions and instruction, delimited by triple backticks:

```
```
Human Action: The human moves to the right from their current position at (5,0) until they reach at (5,4) and gives the instruction.
Instruction: Can you pass me the blue key?
Type: <Identify the instruction type and provide chain-of-thought reasoning for the type based on the current grid configuration>
Response: <Generate chain-of-thought reasoning for the response based on the current grid configuration>
Actions: <Provide optimal actions from the response based on the instruction and the current grid configuration>
```
```

Fig. A.4. Response generation component part-2.

A.3. Demonstrations: CP

Tomcat with CP receives two problem walkthroughs, each stepping through the initial, observed, and completed grids before providing a worked example. Figs. A.5–A.8 present the first walkthrough, whereas Figs. A.9–A.12 detail the second.

Demonstration Exemplars: CP Problem-1 (Initial Grid)

Use the following problems and examples, delimited by triple quotes, to understand how to generate the appropriate type, response, and actions for each instruction and human actions.
’’’
—Problem (1) demonstrating initial, observed, completed grids with figures and the example.
Figure 2: Initial Grid Configuration

```
[['r' '.' '.' '.' 'm' 'W' 'W' 'g']
 ['y' '.' 'W' 'W' '.' 'W' 'W' '.']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' '.']
 ['.' 'R' '.' '.' '.' '.' 'h' '.']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' 'Y']
 ['Y' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['g' 'W' 'g' 'W' 'W' 'W' 'W' 'g']]
```

Object Positions for Figure 2:
My position (Labeled as ’m’): (0, 4)
Human (Labeled as ’h’): (3, 6)
Red key (Labeled as ’r’): (0, 0) −−> Total Red key: 1
Yellow key (Labeled as ’y’): (1, 0) −−> Total Yellow key: 1
Red door (Labeled as ’R’): (3, 1) −−> Total Red door: 1
Yellow doors (Labeled as ’Y’): (5, 7), (6, 0) −−> Total Yellow doors: 2
Gems (Labeled as ’g’): (0, 7), (7, 0), (7, 2), (7, 7) −−> Total Gems: 4
Walls (Labeled as ’W’): (0, 5), (0, 6), (1, 2), (1, 3), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 3), (2, 5), (2, 6), (4, 1), (4, 3), (4, 4), (4, 5), (4, 6), (5, 1), (5, 3), (5, 4), (5, 5), (5, 6), (6, 1), (6, 3), (6, 4), (6, 5), (6, 6), (7, 1), (7, 3), (7, 4), (7, 5), (7, 6) −−> Total Walls: 32
Empty spaces (Labeled as ’.’): (0, 1), (0, 2), (0, 3), (1, 1), (1, 4), (1, 7), (2, 4), (2, 7), (3, 0), (3, 2), (3, 3), (3, 4), (3, 5), (3, 7), (4, 0), (4, 2), (4, 7), (5, 0), (5, 2), (6, 2), (6, 7) −−> Total Empty spaces: 21

–Initial Grid: This grid in Figure 2 displays the initial positions of you, human, keys (red, yellow), doors (red, yellow), and gems.

Fig. A.5. Demonstration exemplar component for Tomcat with CP part-1.

Demonstration Exemplars: CP Problem-1 (Observed Grid)

Figure 3: Observed Grid Configuration

```
[['r' '.' '.' '.' 'm' 'W' 'W' 'g']
 ['y' '.' 'W' 'W' '.' 'W' 'W' '.']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' '.']
 ['.' 'R' 'h' '.' '.' '.' '.' '.']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' 'Y']
 ['Y' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['g' 'W' 'g' 'W' 'W' 'W' 'W' 'g']]
```

Object Positions for Figure 3:
My position (Labeled as 'm'): (0, 4)
Human (Labeled as 'h'): (3, 2)
Red key (Labeled as 'r'): (0, 0) −−> Total Red key: 1
Yellow key (Labeled as 'y'): (1, 0) −−> Total Yellow key: 1
Red door (Labeled as 'R'): (3, 1) −−> Total Red door: 1
Yellow doors (Labeled as 'Y'): (5, 7), (6, 0) −−> Total Yellow doors: 2
Gems (Labeled as 'g'): (0, 7), (7, 0), (7, 2), (7, 7) −−> Total Gems: 4
Walls (Labeled as 'W'): (0, 5), (0, 6), (1, 2), (1, 3), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 3), (2, 5), (2, 6), (4, 1), (4, 3), (4, 4), (4, 5), (4, 6), (5, 1), (5, 3), (5, 4), (5, 5), (5, 6), (6, 1), (6, 3), (6, 4), (6, 5), (6, 6), (7, 1), (7, 3), (7, 4), (7, 5), (7, 6) −−> Total Walls: 32
Empty spaces (Labeled as '.'): (0, 1), (0, 2), (0, 3), (1, 1), (1, 4), (1, 7), (2, 4), (2, 7), (3, 0), (3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (4, 0), (4, 2), (4, 7), (5, 0), (5, 2), (6, 2), (6, 7) −−> Total Empty spaces: 21

–Observed Grid: The human moves left toward the red door in Figure 3 from position (3,6) to (3,2) and gives the instruction, "Can you pass me the red key?" From this, you infer the human's intentions based on their movement and location–specifically, which gem they are aiming to collect.

Fig. A.6. Demonstration exemplar component for Tomcat with CP part-2.

Demonstration Exemplars: CP Problem-1 (Completed Grid)

Figure 4: Completed Grid Configuration

```
[['r' '.' '.' '.' 'm' 'W' 'W' 'g']
 ['y' '.' 'W' 'W' '.' 'W' 'W' '.']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' '.']
 ['.' '.' '.' 'm' '.' '.' '.' '.']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' 'Y']
 ['.' 'W' '.' 'W' 'W' 'W' 'W' '.']
 ['h' 'W' 'g' 'W' 'W' 'W' 'W' 'g']]
```

Object Positions for Figure 4:
My position (Labeled as 'm'): (3, 3)
Human (Labeled as 'h'): (7, 0)
Yellow door (Labeled as 'Y'): (5, 7) −−> Total Yellow door: 1
Gems (Labeled as 'g'): (0, 7), (7, 2), (7, 7) −−> Total Gems: 3
Walls (Labeled as 'W'): (0, 5), (0, 6), (1, 2), (1, 3), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 3), (2, 5), (2, 6), (4, 1), (4, 3), (4, 4), (4, 5), (4, 6), (5, 1), (5, 3), (5, 4), (5, 5), (5, 6), (6, 1), (6, 3), (6, 4), (6, 5), (6, 6), (7, 1), (7, 3), (7, 4), (7, 5), (7, 6) −−> Total Walls: 32
Empty spaces (Labeled as '.'): (0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 0), (1, 1), (1, 4), (1, 7), (2, 4), (2, 7), (3, 0), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (3, 7), (4, 0), (4, 2), (4, 7), (5, 0), (5, 2), (6, 0), (6, 2), (6, 7) −−> Total Empty spaces: 26

–Completed Grid: You infer that the human intends to collect the gem beyond the red door. However, you notice that after the red door, there is also a yellow door blocking the path to the gem. In response, in Figure 4, you collect both the red and yellow keys and pass them to the human. The human then uses the corresponding keys to unlock both doors and successfully retrieves the gem.

Fig. A.7. Demonstration exemplar component for Tomcat with CP part-3.

Demonstration Exemplars: CP Problem-1 (Example)

Example from this grid configuration.
Human Action: The human moves left from their current position at (3,6) to (3,2), which is adjacent to the red door at (3,1). Upon arriving at (3,2), they provide the instruction.
Instruction: Can you pass me the red key?
Type: Unclear. Based on the instruction and the grid, I infer that the human wants to collect the gem at (7,0), beyond a red door. However, before reaching the gem, a yellow door is blocking the path. Therefore, I must collect both the red and yellow keys, and since the instruction doesn't mention picking up the yellow key, this instruction lacks specificity and is thus unclear.
Response: There are two yellow doors, one red door, one red key, and one yellow key on the grid. The instruction asks to collect a red key, so the desired gem the human wants to collect is placed beyond the red door at (3,1). However, a yellow door at (6,0) blocks the path to the gem at (7,0). Thus, I will collect both the red and yellow keys. To minimize movement, I will first pick up the red key at (0,0) followed by the yellow key at (1,0), as this strategy reduces the number of steps from my current position at (0,4). I will then pass them to the human at (3,2) to unlock the doors and collect the gem.
Actions:
1) Collect: red_key at (0,0).
2) Collect: yellow_key at (1,0).
3) Pass: red_key and yellow_key to the human at (3,2).
4) Unlock: human unlocks the Red_door at (3,1) and the Yellow_door at (6,0).
5) Retrieve: human retrieves gem at (7,0).

Fig. A.8. Demonstration exemplar component for Tomcat with CP part-4.

Demonstration Exemplars: CP Problem-2 (Initial Grid)

—Problem (2) demonstrating initial, observed, completed grids with figures and the example.
Figure 5: Initial Grid Configuration

```
[['W' 'W' 'b' 'W' 'W' 'W' 'r' 'W' 'W']
 ['W' 'r' '.' 'r' 'W' 'b' '.' 'b' 'W']
 ['W' 'W' '.' 'W' 'W' 'W' '.' 'W' 'W']
 ['W' 'W' '.' '.' 'm' '.' '.' 'W' 'W']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'g']
 ['h' '.' '.' '.' '.' '.' 'B' 'B' '.']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'g']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' 'R' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' 'R' 'W' 'W' 'W' 'W']
 ['g' '.' '.' '.' '.' '.' '.' '.' 'g']]
```

Object Positions for Figure 5:
My position (Labeled as 'm'): (3, 4)
Human (Labeled as 'h'): (5, 0)
Red keys (Labeled as 'r'): (0, 6), (1, 1), (1, 3) −−> Total Red keys: 3
Blue keys (Labeled as 'b'): (0, 2), (1, 5), (1, 7) −−> Total Blue keys: 3
Red doors (Labeled as 'R'): (8, 4), (9, 4) −−> Total Red doors: 2
Blue doors (Labeled as 'B'): (5, 6), (5, 7) −−> Total Blue doors: 2
Gems (Labeled as 'g'): (4, 8), (6, 8), (10, 0), (10, 8) −−> Total Gems: 4
Walls (Labeled as 'W'): (0, 0), (0, 1), (0, 3), (0, 4), (0, 5), (0, 7), (0, 8), (1, 0), (1, 4), (1, 8), (2, 0), (2, 1), (2, 3), (2, 4), (2, 5), (2, 7), (2, 8), (3, 0), (3, 1), (3, 7), (3, 8), (4, 0), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (6, 0), (6, 1), (6, 2), (6, 3), (6, 5), (6, 6), (6, 7), (7, 0), (7, 1), (7, 2), (7, 3), (7, 5), (7, 6), (7, 7), (7, 8), (8, 0), (8, 1), (8, 2), (8, 3), (8, 5), (8, 6), (8, 7), (8, 8), (9, 0), (9, 1), (9, 2), (9, 3), (9, 5), (9, 6), (9, 7), (9, 8) −−> Total Walls: 59
Empty spaces (Labeled as '.'): (1, 2), (1, 6), (2, 2), (2, 6), (3, 2), (3, 3), (3, 5), (3, 6), (4, 4), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 8), (6, 4), (7, 4), (10, 1), (10, 2), (10, 3), (10, 4), (10, 5), (10, 6), (10, 7) −−> Total Empty spaces: 24

–Initial Grid: This grid in Figure 5 displays the initial positions of you, human, keys (red, blue), doors (red, blue), and gems.

Fig. A.9. Demonstration Exemplars component for Tomcat with CP part-4.

Demonstration Exemplars: CP Problem-2 (Observed Grid)

Figure 6: Observed Grid Configuration

```
[['W' 'W' 'b' 'W' 'W' 'W' 'r' 'W' 'W']
 ['W' 'r' '.' 'r' 'W' 'b' '.' 'b' 'W']
 ['W' 'W' '.' 'W' 'W' 'W' '.' 'W' 'W']
 ['W' 'W' '.' '.' 'm' '.' '.' 'W' 'W']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'g']
 ['.' '.' '.' '.' 'h' '.' 'B' 'B' '.']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'g']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' 'R' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' 'R' 'W' 'W' 'W' 'W']
 ['g' '.' '.' '.' '.' '.' '.' '.' 'g']]
```

Object Positions for Figure 6:
My position (Labeled as 'm'): (3, 4)
Human (Labeled as 'h'): (5, 4)
Red keys (Labeled as 'r'): (0, 6), (1, 1), (1, 3) −−> Total Red keys: 3
Blue keys (Labeled as 'b'): (0, 2), (1, 5), (1, 7) −−> Total Blue keys: 3
Red doors (Labeled as 'R'): (8, 4), (9, 4) −−> Total Red doors: 2
Blue doors (Labeled as 'B'): (5, 6), (5, 7) −−> Total Blue doors: 2
Gems (Labeled as 'g'): (4, 8), (6, 8), (10, 0), (10, 8) −−> Total Gems: 4
Walls (Labeled as 'W'): (0, 0), (0, 1), (0, 3), (0, 4), (0, 5), (0, 7), (0, 8), (1, 0), (1, 4), (1, 8), (2, 0), (2, 1), (2, 3), (2, 4), (2, 5), (2, 7), (2, 8), (3, 0), (3, 1), (3, 7), (3, 8), (4, 0), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (6, 0), (6, 1), (6, 2), (6, 3), (6, 5), (6, 6), (6, 7), (7, 0), (7, 1), (7, 2), (7, 3), (7, 5), (7, 6), (7, 7), (7, 8), (8, 0), (8, 1), (8, 2), (8, 3), (8, 5), (8, 6), (8, 7), (8, 8), (9, 0), (9, 1), (9, 2), (9, 3), (9, 5), (9, 6), (9, 7), (9, 8) −−> Total Walls: 59
Empty spaces (Labeled as '.'): (1, 2), (1, 6), (2, 2), (2, 6), (3, 2), (3, 3), (3, 5), (3, 6), (4, 4), (5, 0), (5, 1), (5, 2), (5, 3), (5, 5), (5, 8), (6, 4), (7, 4), (10, 1), (10, 2), (10, 3), (10, 4), (10, 5), (10, 6), (10, 7) −−> Total Empty spaces: 24

–Observed Grid: The human moves to the right from their current position in Figure 6 from (5,0) to (5,4), and provides an instruction, "Can you pass me the red keys?" From this, you infer the human's intentions based on their movement and location—specifically, which gem they are aiming to collect.

Fig. A.10. Demonstration exemplar component for Tomcat with CP part-5.

Demonstration Exemplars: CP Problem-2 (Completed Grid)

Figure 7: Completed Grid Configuration

```
[['W' 'W' 'b' 'W' 'W' 'W' 'r' 'W' 'W']
 ['W' '.' '.' ',' 'W' 'b' '.' 'b' 'W']
 ['W' 'W' '.' 'W' 'W' 'W' '.' 'W' 'W']
 ['W' 'W' '.' '.' '.' '.' '.' 'W' 'W']
 ['W' 'W' 'W' 'W' 'm' 'W' 'W' 'W' 'g']
 ['.' '.' '.' '.' ',' '.' 'B' 'B' '.']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'g']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W']
 ['W' 'W' 'W' 'W' '.' 'W' 'W' 'W' 'W']
 ['h' '.' '.' '.' '.' '.' '.' '.' 'g']]
```

Object Positions for Figure 7:
My position (Labeled as 'm'): (4, 4)
Human (Labeled as 'h'): (10, 0)
Red key (Labeled as 'r'): (0, 6) −−> Total Red key: 1
Blue keys (Labeled as 'b'): (0, 2), (1, 5), (1, 7) −−> Total Blue keys: 3
Red doors (Labeled as 'R'): (8, 4), (9, 4) −−> Total Red doors: 2
Blue doors (Labeled as 'B'): (5, 6), (5, 7) −−> Total Blue doors: 2
Gems (Labeled as 'g'): (4, 8), (6, 8), (10, 8) −−> Total Gems: 3
Walls (Labeled as 'W'): (0, 0), (0, 1), (0, 3), (0, 4), (0, 5), (0, 7), (0, 8), (1, 0), (1, 4), (1, 8), (2, 0), (2, 1), (2, 3), (2, 4), (2, 5), (2, 7), (2, 8), (3, 0), (3, 1), (3, 7), (3, 8), (4, 0), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6), (4, 7), (6, 0), (6, 1), (6, 2), (6, 3), (6, 5), (6, 6), (6, 7), (7, 0), (7, 1), (7, 2), (7, 3), (7, 5), (7, 6), (7, 7), (7, 8), (8, 0), (8, 1), (8, 2), (8, 3), (8, 5), (8, 6), (8, 7), (8, 8), (9, 0), (9, 1), (9, 2), (9, 3), (9, 5), (9, 6), (9, 7), (9, 8) −−> Total Walls: 59
Empty spaces (Labeled as '.'): (1, 1), (1, 2), (1, 3), (1, 6), (2, 2), (2, 6), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 8), (6, 4), (7, 4), (10, 1), (10, 2), (10, 3), (10, 4), (10, 5), (10, 6), (10, 7) −−> Total Empty spaces: 27

–Completed Grid: You infer that the human intends to retrieve the gem located beyond the two red doors on the grid. Observing that there are three red keys available, you strategically choose to collect the two keys positioned on the upper left of the grid, as they require the fewest steps from your current location. After gathering the keys, you pass them to the human, who then uses them to unlock both doors and successfully retrieve the gem.

Fig. A.11. Demonstration exemplar component for Tomcat with CP part-6.

Demonstration Exemplars: CP Problem-2 (Example)

Example from this grid configuration.
Human Action: The human moves to the right from their current position at (5,0) to (5,4), where they provide the instruction before continuing their movement.
Instruction: Can you pass me the red keys?
Type: Clear. Based on the instruction and the grid, I infer that the human wants to collect the gem at (10,0) or at (10,8), which is beyond two red doors. Since the human needs two red keys to unlock the doors and reach the gem, the instruction is clear.
Response: There are three red keys, three blue keys, two red doors, and two blue doors on the grid. The instruction asks to collect red keys to retrieve the gem at (10,0) or (10,8). I will collect the two closest red keys at (1,1) and (1,3), as this minimizes the steps required to retrieve them. Since the human continues moving after the instruction, they are likely heading downward from (5,4) to (7,4), adjacent to the red door at (8,4). I will pass the keys to the human at either (5,4) or (7,4), depending on their movements, enabling them to unlock the red doors at (8,4) and (9,4) to access the desired gem.
Actions:
1) Collect: red_key at (1,1).
2) Collect: red_key at (1,3).
3) Pass: red_keys to the human's future position either at (5,4) or (7,4).
4) Unlock: human unlocks the Red_doors at (8,4) or (9,4).
5) Retrieve: human retrieves gem at either (10,0) or (10,8).
' ' '

Fig. A.12. Demonstration exemplar component for Tomcat with CP part-7.

A.4. Demonstration: Fs-CoT

Tomcat with Fs-CoT receives seven input–output exemplars ($k = 7$) in place of the walkthroughs, each pairing a grid, a human action, and an instruction with the reasoning and actions they should produce. Figs. A.13–A.19 illustrate the seven exemplars in order.

Demonstration Exemplars: Fs-CoT Exemplar-1 (k=7)

Use the following problems and examples, delimited by triple quotes, to understand how to generate the appropriate type, response, and actions for each instruction and human actions.
'''
Human Action: The human moves downward from their current position at (6,4) to (8,4), where they provide the instruction.
Instruction: Can you pass me the blue key?
Type: Unclear. Based on the human's movement, instruction and the current grid configuration, I infer that the human moved to (8,4) to collect the gem at (8,9) located beyond the blue door and red door at (8,6) and (8,8) respectively since they asked for a blue key. To retrieve the gem the human needs to unlock both these doors. Since the human does not mention passing the red key, the instruction is unclear.
Response: The instruction asks to collect a blue key and based on the human movements and the grid configuration the desired gem they want to collect is placed at (8,9). But this gem is beyond the blue door and red door at (8,6) and (8,8) respectively. So I will collect one blue key and one red key located at (3,8) and (5,8). To minimize movement, I will first pick up the blue key at (3,8) followed by the red key at (5,8), as this strategy reduces the number of steps from my current position at (1,8) to the human at (8,4). Once both keys are collected, I will pass them to the human, allowing them to unlock blue door and red door at (8,6) and (8,8) respectively and retrieve the gem at (8,9).
Actions:
1) Collect: blue_key at (3,8).
2) Collect: red_key at (5,8).
3) Pass: blue_key and red_key to the human at (8,4).
4) Unlock: human unlocks the Blue_door at (8,6) and the Red_door at (8,8).
5) Retrieve: human retrieves gem at (8,9).

Fig. A.13. Demonstration exemplar component for Tomcat with Fs-CoT part-1.

Demonstration Exemplars: Fs-CoT Exemplar-2 (k=7)

Human Action: The human moves left from their current position at (0,5) to (0,3), where they provide the instruction before continuing their movement.
Instruction: On my way to pick up the blue key, can you find a yellow key?
Type: Clear. Based on the human's movement, instruction and the current grid configuration, I infer that the human moved to (0,3) to pick up the blue key at (1,0) and requested the yellow key to retrieve the gem at (2,10) located beyond both the blue and yellow doors at (4,7) and (3,10) respectively. Since both the keys are required to retrieve the gem at (2,10), the instruction is clear.
Response: The instruction asks to collect a yellow key while the human is picking up a blue key, so the desired gem the human wants to collect is placed beyond a blue and a yellow door. To get the gem at (2,10) the human needs to go through a blue door at (4,7) and a yellow door at (3,10), thus the gem at (2,10) must be the desired gem. Since the human is picking up the blue key at (1,0) based on their movement towards the left direction, I will collect the yellow key. The human after collecting the blue key will move to the blue door to unlock the blue door at (4,7), thus their future position can be (4,10) as they will keep moving towards the gem at (2,10). To minimize movement, I will pick up the yellow key at (8,10), as this strategy reduces the number of steps from my current position at (10,9). Once the key is collected, I will pass it to the human at (4,10) or their future position, after they unlock the blue door at (4,7). The human can then unlock the yellow door at (3,10) to retrieve the gem at (2,10).
Actions:
1) Collect: yellow_key at (8,10).
2) Unlock: human unlocks Blue_door at (4,7).
3) Pass: yellow_key to the human at (4,10) or their future position.
4) Unlock: human unlocks the Yellow_door at (3,10).
5) Retrieve: human retrieves gem at (2,10).

Fig. A.14. Demonstration exemplar component for Tomcat with Fs-CoT part-2.

Demonstration Exemplars: Fs-CoT Exemplar-3 (k=7)

Human Action: The human moves right from their current position at (3,3) to (3,5), where they provide the instruction before continuing their movement.
Instruction: I'll get the blue key. Can you pick up a red key?
Type: Clear. Based on the human's movement, instruction and the current grid configuration, I infer that the human moved to (3,5) to pick up the blue key at (3,8) and requested the red key to collect either the gem at (11,8) or the gem at (11,12) located behind the red and blue doors at (9,10) and (10,10) respectively. Since the red and blue keys are required for the human to reach either one of their desired gems, the instruction is clear.
Response: The instruction asks to collect a red key while the human is picking up a blue key, so the desired gem the human wants to collect is placed beyond a blue and a red door. To get the gem at either (11,8) or (11,12) the human needs to go through a red door at (9,10) and a blue door at (10,10), thus the gems at either (11,8) or (11,12) must be one of the desired gems. Since the human is picking up the blue key at (3,8) based on their movement towards the right direction, I will collect the red key. The human after collecting the blue key at (3,8) will move towards the red and blue doors at (9,10) and (10,10), thus their future position can be at (8,10), located just before both the doors. To minimize movement, I will pick up the red key located at (8,6), as this strategy reduces the number of steps from my current position at (6,3). Once the key is collected, I will pass it to the human at (8,10) or their future position, allowing them to unlock the both the doors and retrieve either one of the gems.
Actions:
1) Collect: red_key at (8,6).
2) Pass: red_key to the human at (8,10) or their future position.
3) Unlock: human unlocks Red_door at (9,9) and Blue_door at (10,10).
4) Retrieve: human retrieves a gem at either (11,8) or (11,12).

Fig. A.15. Demonstration exemplar component for Tomcat with Fs-CoT part-3.

Demonstration Exemplars: Fs-CoT Exemplar-4 (k=7)

Human Action: The human moves right from their current position at (5,3) to (5,6) and then moves downward to (6,6), where they provide the instruction before continuing their movement.
Instruction: Can you go get the red key?
Type: Unclear. Based on the human's movement, instruction and the current grid configuration, I infer that the human moved to (6,6) to pick up the blue key at (8,6) even though it was not mentioned in the instruction, and requested the red key to collect either the gem at (8,0) or (8,4) located behind the red and blue doors at (6,3) and (7,3) respectively. Since the human does not mention anything about a blue key, the instruction is unclear.
Response: The instruction asks to collect a red key while the human is likely picking up a blue key, so the desired gem the human wants to collect is placed beyond a blue and a red door. To get the gem at either (8,0) or (8,4) the human needs to go through a red door at (6,3) and a blue door at (7,3), thus the gem at either (8,0) or (8,4) must be the desired gem. Since the human is likely picking up the blue key at (8,6) based on their movement towards the downward direction, I will collect the red key. The human after collecting the blue key at (8,6) will move towards the red and blue doors at (6,3) and (7,3), thus their future position can be at (5,3), located just before both the doors. There are two red keys on the grid at (0,7) and at (3,7). Since the red key at (3,7) is blocked by a blue door, I will collect the red key at (0,7) such that it takes minimal steps. Once the red key is collected, I will pass it to the human at (5,3) or their future position, allowing them to unlock the doors and collect either one of the gems.
Actions:
1) Collect: red_key at (0,7).
2) Pass: red_key to the human at (5,3) or their future position.
3) Unlock: human unlocks Red_door at (6,3) and Blue_door at (7,3).
4) Retrieve: human retrieve a gem at either (8,0) or (8,4).

Fig. A.16. Demonstration exemplar component for Tomcat with Fs-CoT part-4.

Demonstration Exemplars: Fs-CoT Exemplar-5 (k=7)

Human Action: The human moves upward from their current position at (3,5) to (0,5), where they provide the instruction before continuing their movement.
Instruction: I will pick up this red key. Can you find a yellow one?
Type: Unclear. Based on the human's movement, instruction and the current grid configuration, I infer that the human moved to (0,5) to pick up the red key at (0,0) and requested for a yellow key to collect either one of the gems at (7,2), (7,4), or (7,6), respectively. Because to reach these gems it requires to unlock a yellow door first and then a red door. Since there are many possibilities to reach gems which will require both yellow and red keys, it is impossible to infer which gem the human wants, making this an unclear instruction.
Response: The instruction asks for a yellow key while the human is picking up a red key, so the desired gem the human wants to collect is placed beyond a yellow and a red door. To get the gems at (7,2), (7,4), or (7,6), the human needs to go through a yellow and a red door at. Therefore, based on the human's future position their desired gem can be any one of them. A red door at (6,2) is placed after a yellow door at (5,2). Two red doors at (6,4) and (6,6) are placed after a yellow door at (4,5). Therefore, the human's future position will be before the two yellow doors at (4,2) or (3,5), or it can be someplace else. To minimize movement, I will pick up the yellow key at (9,0) to reduce the number of steps from my current position at (9,5) to the human at (3,5) or (4,2) or their future position. Once I collect the yellow key, I will pass it to the human at (3,5) or (4,2), allowing them to unlock the doors and receive the gem.
Actions:
1) Collect: yellow_key at (9,0).
2) Pass: yellow_key to the human at (3,5) or (4,2) or their future position.
3) Unlock: human unlocks the Yellow_door at (5,2) or (4,5) and the Red_door at (6,2) or (6,4), (6,6).
4) Retrieve: human retrieves the gem at either (7,2), (7,4) or (7,6).

Fig. A.17. Demonstration exemplar component for Tomcat with Fs-CoT part-5.

Demonstration Exemplars: Fs-CoT Exemplar-6 (k=7)

Human Action: The human moves down from their current position at (4,4) to (5,4), and then proceeds to move left to collect the red key at (5,0). After collecting the key, they continue moving right until they reach (5,2), where they provide the instruction before continuing their movement.
Instruction: Can you help me unlock the blue door there?
Type: Clear. Based on the human's movement, instruction and the current grid configuration, I infer that the human picked up the red key at (5,0), moved to (5,2), and requested me to unlock blue door to retrieve the gem at (0,0), which is located beyond the blue and red doors at (2,4) and (2,0) respectively. To retrieve the gem the human needs to pass both these doors. Since the human picked up the red key and I just have to collect the blue key to unlock the blue door blocking the human's path to the gem, the instruction is clear.
Response: The instruction asks to unlock the blue door while the human has already picked up the red key at (5,0). The human's desired gem based on their actions and the grid configuration must be beyond a blue and a red door. The gem at (0,0) is the desired gem as it is placed beyond a blue door at (2,4) and a red door at (2,0). To minimize movement, I will collect the only accessible blue key at (0,8) and then move to unlock the blue door at (2,4) which is blocking the path to the gem at (0,0). The human can then unlock the red door at (2,0) to retrieve the gem at (0,0).
Actions:
1) Collect: blue_key at (0,8).
2) Unlock: Blue_door at (2,4).
3) Unlock: human unlocks Red_door at (2,0).
4) Retrieve: human retrieves gem at (0,0).

Fig. A.18. Demonstration exemplar component for Tomcat with Fs-CoT part-6.

Demonstration Exemplars: Fs-CoT Exemplar-7 (k=7)

Human Action: The human moves left from their current position at (6,2) to (6,1) and then upward to (5,1). Upon reaching (5,1), adjacent to the red door at (4,1), they provide the instruction.
Instruction: Can you unlock this red door?
Type: Unclear. Based on the human's movement, instruction and the current grid configuration, I infer that the human moved to (5,1) and asked to unlock the red door at (4,1) to retrieve the gem at (1,1) located behind a blue and red door at (2,1) and (4,1) respectively. To retrieve the gem the human needs to have both doors unlocked. However, since there is no mention about unlocking the blue door, the instruction is unclear.
Response: The instruction specifies unlocking a red door, indicating that the desired gem must be behind one. Based on the grid configuration, there are two gems on the grid, both blocked by a red and a blue door. Given that the human has moved to (5,1), adjacent to the red door at (4,1), they are likely targeting the gem at (1,1), which lies beyond the adjacent red door. However, the gem at (1,1) is also blocked by a blue door at (2,1), in addition to the red door at (4,1). To assist the human, I will collect both the red and blue keys. To minimize movement, I will first collect the red key at (0,5), followed by the blue key at (2,7), as this approach reduces the total steps from my current position at (2,5). After collecting both keys, I will proceed to unlock the blue door at (2,1) and then the red door at (4,1), enabling the human to access the gem at (1,1).
Actions:
1) Collect: red_key at (0,5).
2) Collect: blue_key at (2,7).
3) Unlock: Blue_door at (2,1) and Red_door at (4,1).
4) Retrieve: human retrieves gem at (1,1).
'''

Fig. A.19. Demonstration exemplar component for Tomcat with Fs-CoT part-7.

A.5. Overcooked

The Overcooked prompt keeps the component order of the Doors, Keys, and Gems prompt above. Only the domain content and the Response Generation envelope differ; Figs. A.20–A.22 show the parts that changed.

Common Ground: Overcooked

You assist a human in a cooperative cooking domain called Overcooked, set in a gridworld kitchen. The human and you must together prepare and deliver the soup orders that are pending, and you primarily assist the human. You help by fetching ingredients, loading the pot, taking a dish, plating a cooked soup, delivering it to a serving station, or handing an item to the human. Think of it as a team game where a human gives callouts to their teammate (you), expecting collaboration to achieve their goal. The challenge for you is correctly understanding human instructions, which may require some inferencing due to a lack of specificity in the instructions.

Background: This collaborative game is played in a grid kitchen containing the following objects: pots, onion dispensers, tomato dispensers, dish dispensers, serving stations, counters, empty floor, a human, and you.

Figure 1: Kitchen Configuration

```
[['X' 'X' 'P' 'X' 'X']
 ['T' '.' '.' 'm' 'O']
 ['X' 'h' '.' '.' 'X']
 ['X' 'D' 'X' 'S' 'X']]
```

Human's objective: The human wants the pending order completed, and says so in a single instruction. For instance, consider the configuration in Figure 1. Suppose two orders are pending, a 1-onion soup and then a 1-tomato soup, and the human says *"Can you pass the ingredient?"* This kitchen holds one onion dispenser at (1, 4) and one tomato dispenser at (1, 0), so *the ingredient* has two candidate referents and the sentence resolves neither. Because orders are served first come, first served, the active order is the onion soup. You must infer this from the order queue even though the instruction never names it. In other configurations the missing information is a count, a pot, or a serving station instead.

Your objective: infer what the human intends from their instruction, their movement and the pending order, then take only the actions needed to complete that order in as few steps as possible.

Fig. A.20. Common Ground component for Overcooked.

Objectives and Rules: Overcooked

Rules of the Game:

- For each kitchen configuration (problem), the human provides a single instruction called a callout.
- The pending order names the ingredients required. An order is one of: 1-, 2-, or 3-onion soup, 1-, 2-, or 3-tomato soup, 1-onion-1-tomato soup, or 2-onion-2-tomato soup. When two orders are pending, they are served first come, first served.
- An onion dispenser (O) holds 10 onions and a tomato dispenser (T) holds 10 tomatoes, so a dispenser never runs out during a single order.
- Movement is orthogonal only. You interact with an object when orthogonally adjacent to it, and must be empty-handed to pick an item up.
- Each of you carries exactly one item at a time. The human cannot accept a second item before disposing of the first.
- Cooking starts automatically as soon as the pot holds exactly the ingredients the order requires.
- Serving takes three steps in order: take a dish from a dispenser (D), plate the soup at the pot (P), deliver at a serving station (S).
- Counters (`X`) cannot be walked on, but an item can be placed on one and picked up from one. This is how the two of you hand items over when you cannot meet.
- Some kitchens divide the human and you into separate regions. There, check which objects are reachable by whom and route every transfer through a counter touching both regions.
- When you hand work over, hand over the most finished item you can. Sharing a region, plate the soup and hand over the soup; divided by a counter, leave the dish on the shared counter.
- All ingredients cost the same, so the only thing to minimise is movement.

Fig. A.21. Objectives and Rules component for Overcooked.

Order Heuristics and Output Contract: Overcooked

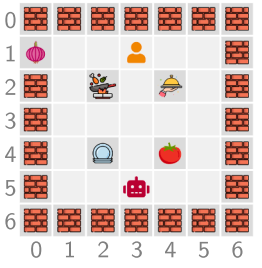
How an order is completed: for every order, put in one ingredient at a time until the pot holds exactly what the order needs, at which point cooking starts. Then take a dish, plate the soup at the pot, and deliver it at a serving station. So a 1-onion soup is 2 loading steps plus 3 serving steps, a 2-onion soup is 4 plus 3, a 1-onion-1-tomato soup is 4 plus 3 with one of each, and a 2-onion-2-tomato soup is 8 plus 3.

Rules for the JSON (envelope in Figure 14): use only the listed values, spelled exactly. `actor` is `agent` for you and `human` for the principal. `item` is what is carried; `object` is what it is taken from or put into, written as the layout character, so `X` is a counter, `h` the human and `m` you. Movement is implied and must not appear as an action. Which verb goes with which object: *Get* from O, T or D, or from X to lift an item left there; *Place* into P; *Put* onto X when you cannot meet; *Pass* to h or m when adjacent; *Plate* from P, consuming the dish and yielding soup; *Deliver* at S. List every action needed to finish the order, including the human's steps, ending with the delivery. `intent_target` restates the intent as bare values and must agree with the actions; name the one pot and station you commit to, and never name one that cannot be used.

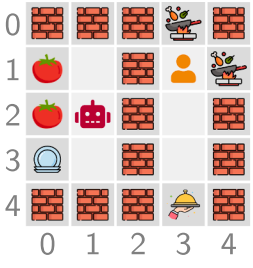
Fig. A.22. Order heuristics and the JSON output contract for Overcooked.

Appendix B. Overcooked layouts

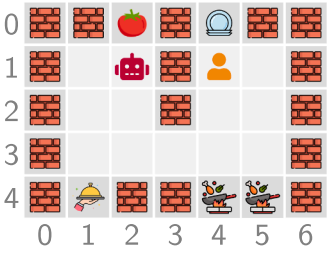
Fig. B.1 shows all seven Overcooked layouts. Maps 1–7 carry the 20 evaluation scenarios, three per map, except Maps 6 and 7, which carry two, and Map 5 carries four.



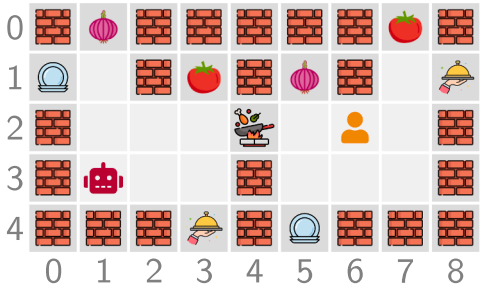
(a) Map 3: Open Room



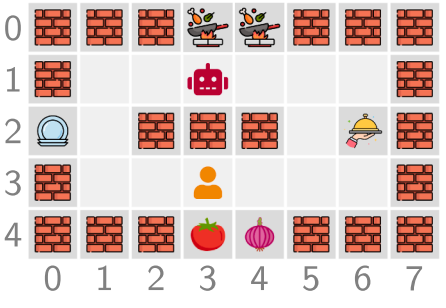
(b) Map 6: Forced Coordination



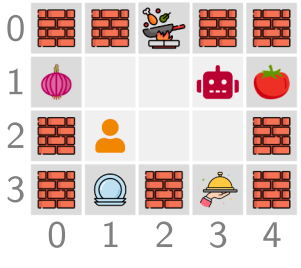
(c) Map 7: Bottleneck



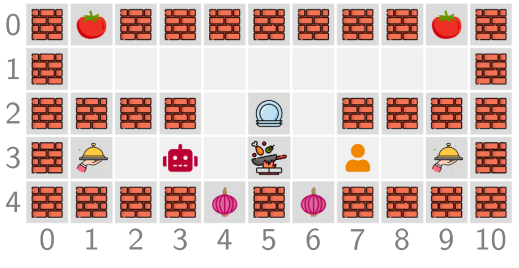
(d) Map 2: Mirrored Halves



(e) Map 4: Counter Circuit



(f) Map 1: Cramped Room



(g) Map 5: Twin Stations

Fig. B.1. The seven Overcooked layouts. The bricks/walls are counters and can be used for passing items; the remaining objects are the pot; onion, tomato, and dish dispensers; and the serving station.

References

[1] H.P. Grice, Logic and Conversation, in: P. Cole, J.L. Morgan (Eds.), Speech Acts, 3 in Syntax and Semantics, Academic Press, New York, 1975, pp. 41–58. https://doi.org/10.1163/9789004368811_003

[2] J.R. Searle, Indirect speech acts, in: P. Cole, J.L. Morgan (Eds.), Speech Acts, 3 in Syntax and Semantics, Academic Press, New York, 1975, pp. 59–82. Reprinted in [57]. https://doi.org/10.1163/9789004368811_004

[3] T. Zhi-Xuan, L. Ying, V. Mansinghka, J.B. Tenenbaum, Pragmatic instruction following and goal assistance via cooperative language-guided inverse planning, in: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, Auckland, New Zealand, 2024, pp. 2094–2103. https://doi.org/10.5555/3635637.3663074

[4] F. Saad, P.K. Murukannaiah, M.P. Singh, Gricean norms as a basis for effective collaboration, in: Proceedings of the 24th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), IFAAMAS, Detroit, 2025, pp. 1812–1820. https://doi.org/10.5555/3709347.3743817

[5] C. Qian, B. He, Z. Zhuang, J. Deng, Y. Qin, X. Cong, Y. Lin, Z. Zhang, Z. Liu, M. Sun, Tell me more! towards implicit user intention understanding of language model driven agents, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 1088–1113. https://doi.org/10.18653/v1/2024.acl-long.61

[6] M. Tomasello, M. Carpenter, J. Call, T. Behne, H. Moll, Understanding and sharing intentions: the origins of cultural cognition, Behav. Brain Sci. 28 (5) (2005) 675–691.

[7] A. Duranti, The Anthropology of Intentions, Cambridge University Press, 2015.

[8] D. Premack, G. Woodruff, Does the Chimpanzee have a theory of mind?, Behav. Brain Sci. 1 (4) (1978) 515–526. https://doi.org/10.1017/S0140525X00076512

[9] C. Frith, U. Frith, Theory of mind, Curr. Biol. 15 (17) (2005) R644–R645.

[10] E. Erdogan, F. Dignum, R. Verbrugge, P. Yolum, ToMA: computational theory of mind with abstractions for hybrid intelligence, J. Artif. Intell. Res. 82 (2025) 285–311. https://doi.org/10.1613/JAIR.1.16402

[11] J.-K. Chong, T.-H. Ho, C. Camerer, A generalized cognitive hierarchy model of games, Games Econ. Behav. 99 (2016) 257–274. https://doi.org/10.1016/j.geb.2016.08.007

[12] H. De Weerd, R. Verbrugge, B. Verheij, Higher-order theory of mind is especially useful in unpredictable negotiations, Auton. Agents Multi-Agent Syst. 36 (2) (2022) 30. https://doi.org/10.1007/s10458-022-09558-6

[13] Ş. Sarkadi, An arms race in theory-of-mind: deception drives the emergence of higher-level theory-of-mind in agent societies, in: IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), IEEE, Toronto, ON, Canada, 2023, pp. 1–10.

[14] S.A. Wu, R.E. Wang, J. Evans, J.B. Tenenbaum, D.C. Parkes, M. Kleiman-Weiner, Too many cooks: Bayesian inference for coordinating multi-agent collaboration, Top. Cogn. Sci. 13 (2) (2021) 414–432. https://doi.org/10.1111/tops.12525

[15] J. Richards, M. Wessel, What you need is what you get: theory of mind for an LLM-based code understanding assistant, in: International Conference on Software Maintenance and Evolution (ICSME), IEEE, Arizona, 2024, pp. 666–671. https://doi.org/10.1109/ICSME58944.2024.00070

[16] Y. Wan, J. Mao, J.B. Tenenbaum, HandMeThat: human-robot communication in physical and social environments, in: Advances in Neural Information Processing Systems 35 (NeurIPS 2022), Curran Associates, Inc., New York, 2022, pp. 12014–12026. https://doi.org/10.5555/3600270.3601143

[17] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, A. Garg, ProgPrompt: program generation for situated robot task planning using large language models, Auton. Robots 47 (8) (2023) 999–1012. https://doi.org/10.1007/s10514-023-10135-3

[18] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A.T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, N.J. Joshi, K. Jeffrey, R.J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, C.K. Fu, Do as i can, not as i say: grounding language in robotic affordances, in: Proceedings of the 6th Conference on Robot Learning, vol. 205, PMLR, 2023, pp. 287–318. https://proceedings.mlr.press/v205/ichter23a.html.

[19] J.W.A. Strachan, D. Albergo, G. Borghini, O. Pansardi, E. Scaliti, S. Gupta, K. Saxena, A. Rufo, S. Panzeri, G. Manzi, M.S.A. Graziano, C. Becchio, Testing theory of mind in large language models and humans, Nat. Hum. Behav. 8 (2024) 1285–1295. https://doi.org/10.1038/s41562-024-01882-z

[20] M. Kosinski, Evaluating large language models in theory of mind tasks, Proc. Natl. Acad. Sci. 121 (45) (2024) e2405460121. https://doi.org/10.1073/pnas.2405460121

[21] K. Gandhi, J.-P. Fraenken, T. Gerstenberg, N. Goodman, Understanding social reasoning in language models with language models, in: Proceedings of the 37th International Conference on Neural Information Processing Systems, Volume 36 of NIPS '23, Curran Associates, Inc., Red Hook, NY, USA, 2023, pp. 13518–13529.

[22] Y. Wu, Y. He, Y. Jia, R. Mihalcea, Y. Chen, N. Deng, Hi-ToM: a benchmark for evaluating higher-order theory of mind reasoning in large language models, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP, Association for Computational Linguistics, Singapore, 2023, pp. 10691–10706. https://doi.org/10.18653/v1/2023.findings-emnlp.717

[23] J.W.A. Strachan, D. Albergo, G. Borghini, O. Pansardi, E. Scaliti, S. Gupta, K. Saxena, A. Rufo, S. Panzeri, G. Manzi, M.S.A. Graziano, C. Becchio, Testing theory of mind in large language models and humans, Nat. Hum. Behav. 8 (2024) 1285–1295. https://doi.org/10.1038/s41562-024-01882-z

[24] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc., Red Hook, NY, USA, 2020, p. 159. https://dl.acm.org/doi/abs/10.5555/3495724.3495883.

[25] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E.H. Chi, Q.V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2022, p. 1800. https://doi.org/10.5555/3600270.3602070

[26] H. Kim, M. Sclar, X. Zhou, R. Bras, G. Kim, Y. Choi, M. Sap, FANToM: a benchmark for stress-testing machine theory of mind in interactions, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 14397–14413. https://doi.org/10.18653/v1/2023.emnlp-main.890

[27] C. Chan, J. Cheng, Y. Yim, Z. Deng, W. Fan, H. Li, X. Liu, H. Zhang, W. Wang, Y. Song, NegotiationToM: a benchmark for stress-testing machine theory of mind in negotiation scenarios, in: Findings of the Association for Computational Linguistics: EMNLP, Association for Computational Linguistics, Miami, Florida, 2024, pp. 4211–4241. https://doi.org/10.18653/v1/2024.findings-emnlp.244

[28] K. Chawla, J. Ramirez, R. Clever, G. Lucas, J. May, J. Gratch, CaSiNo: a corpus of campsite negotiation dialogues for automatic negotiation systems, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 3167–3185. https://doi.org/10.18653/v1/2021.naacl-main.254

[29] W. Street, LLM Theory of mind and alignment: opportunities and risks, in: Proceedings of the Workshop on Theory of Mind in Human-AI Interaction (ToMinHAI@CHI2024), ACM, Honolulu, HI, USA, 2024, pp. 1–7. Workshop paper; arXiv:2405.08154.

[30] W. Street, J.O. Siy, G. Keeling, A. Baranes, B. Barnett, M. McKibben, T. Kanyere, A. Lentz, R.I.M. Dunbar, LLMs achieve adult human performance on higher-order theory of mind tasks, CoRR (2024). https://arxiv.org/abs/2405.18870.

[31] M. Jamali, Z.M. Williams, J. Cai, Unveiling theory of mind in large language models: a parallel to single neurons in the human brain, CoRR (2023). arXiv:2309.01660.

[32] M. Rocha, H.H.d. Silva, A.S. Morales, S. Sarkadi, A.R. Panisson, Applying theory of mind to multi-agent systems: a systematic review, in: Intelligent Systems, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2023, pp. 367–381. https://doi.org/10.1007/978-3-031-45368-7_24

[33] M. Shvo, T.Q. Klassen, S.A. McIlraith, Resolving misconceptions about the plans of agents via theory of mind, in: Proceedings of the International Conference on Automated Planning and Scheduling, 32, 2022, pp. 719–729. https://doi.org/10.1609/icaps.v32i1.19862

[34] H. Shi, S. Ye, X. Fang, C. Jin, L. Isik, Y.-L. Kuo, T. Shu, MuMA-ToM: multi-modal multi-agent theory of mind, in: Proceedings of the AAAI Conference on Artificial Intelligence, 39, 2025, pp. 1510–1519. https://doi.org/10.1609/aaai.v39i2.32142

[35] Y. Yim, C. Chan, T. Shi, Z. Deng, W. Fan, T. Zheng, Y. Song, Evaluating and enhancing LLMs agent based on theory of mind in guandan: a multi-player cooperative game under imperfect information, in: Proceedings of the IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology, 2024, pp. 461–465. https://doi.org/10.1109/WI-IAT62293.2024.00074

[36] J.W.A. Strachan, O. Pansardi, E. Scaliti, M. Celotto, K. Saxena, C. Yi, F. Manzi, A. Rufo, G. Manzi, M.S.A. Graziano, S. Panzeri, C. Becchio, GPT-4o reads the mind in the eyes, CoRR abs/2410.22309 (2024). arXiv:2410.22309.

[37] M. Sap, R.L. Bras, D. Fried, Y. Choi, Neural theory-of-mind? On the limits of social intelligence in large LMs, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 3762–3780. https://doi.org/10.18653/v1/2022.emnlp-main.248

[38] T. Ullman, Large language models fail on trivial alterations to theory-of-mind tasks, CoRR abs/2302.08399 (2023). arXiv:2302.08399.

[39] M. Verma, S. Bhambri, S. Kambhampati, Theory of mind abilities of large language models in human-robot interaction: an illusion?, in: Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, 2024, pp. 36–45.

[40] M. Amirizaniani, E. Martin, M. Sivachenko, A. Mashhadi, C. Shah, Can LLMs reason like humans? Assessing theory of mind reasoning in LLMs for open-ended questions, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 34–44.

[41] H. Li, Y. Chong, S. Stepputtis, J. Campbell, D. Hughes, C. Lewis, K. Sycara, Theory of mind for multi-agent collaboration via large language models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 180–192. https://doi.org/10.18653/v1/2023.emnlp-main.13

[42] S. Zhang, X. Wang, W. Zhang, Y. Chen, L. Gao, D. Wang, W. Zhang, X. Wang, Y. Wen, Mutual theory of mind in human-AI collaboration: an empirical study with LLM-driven AI agents in a real-time shared workspace task, CoRR abs/2409.08811 (2024). https://arxiv.org/abs/2409.08811.

[43] I. Oguntola, J. Campbell, S. Stepputtis, K. Sycara, Theory of mind as intrinsic motivation for multi-agent reinforcement learning, CoRR abs/2307.01158 (2023). ICML Workshop on Theory of Mind in Communicating Agents. https://doi.org/10.48550/arXiv.2307.01158
[44] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, ReAct: synergizing reasoning and acting in language models, in: Proceedings of the 11th International Conference on Learning Representations (ICLR), OpenReview.net, Kigali, Rwanda, 2023, pp. 1–33. arXiv:2210.03629.
[45] E.V. Clark, Common Ground, Wiley Online Library, The Handbook of Language Emergence, 2015, pp. 328–353. https://doi.org/10.1002/9781118346136.ch15
[46] I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, M. Farajtabar, GSM-symbolic: understanding the limitations of mathematical reasoning in large language models, in: Proceedings of the 13th International Conference on Learning Representations (ICLR), 2025, pp. 94743–94765. https://proceedings.iclr.cc/paper_files/paper/2025/file/ec2e7a896f8250986b3907f57621ce94-Paper-Conference.pdf.
[47] P. Shojaee, I. Mirzadeh, K.A. Vahid, M. Horton, S. Bengio, M. Farajtabar, The illusion of thinking: understanding the strengths and limitations of reasoning models via the lens of problem complexity, in: Advances in Neural Information Processing Systems, 38, Main Conference, Curran Associates, Inc., 2025, pp. 108018–108059. https://proceedings.neurips.cc/paper_files/paper/2025/file/9b26ad15462c81548c0689188d2e8018-Paper-Conference.pdf. https://doi.org/10.52202/085713-3603
[48] M. Carroll, R. Shah, M.K. Ho, T. Griffiths, S. Seshia, P. Abbeel, A. Dragan, On the utility of learning about humans for human-AI coordination, in: Advances in Neural Information Processing Systems (NeurIPS), 32, Curran Associates, Inc., Vancouver, Canada, 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf.
[49] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al., DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning, Nature 645 (8081) (2025) 633–638. https://doi.org/10.1038/s41586-025-09422-z
[50] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C.L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P.F. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, in: Advances in Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., New York, 2022, pp. 27730–27744. https://doi.org/10.5555/3600270.3602281
[51] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, L. Zettlemoyer, Rethinking the role of demonstrations: what makes in-context learning work?, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 11048–11064. https://doi.org/10.18653/v1/2022.emnlp-main.759
[52] A. Rame, N. Vieillard, L. Hussenot, R. Dadashi-Tazehozi, G. Cideron, O. Bachem, J. Ferret, WARM: on the benefits of weight averaged reward models, in: Proceedings of the 41st International Conference on Machine Learning, Volume 235 of Proceedings of Machine Learning Research, PMLR, 2024, pp. 42048–42073. https://proceedings.mlr.press/v235/rame24a.html.
[53] J. Qi, Z. Xu, Y. Shen, M. Liu, D. Jin, Q. Wang, L. Huang, The art of SOCRATIC questioning: recursive thinking with large language models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 4177–4199. https://doi.org/10.18653/v1/2023.emnlp-main.255
[54] Y. Qu, T. Zhang, N. Garg, A. Kumar, Recursive introspection: teaching language model agents how to self-improve, in: Advances in Neural Information Processing Systems, vol. 37, Curran Associates, Inc., 2024, pp. 55249–55285. https://proceedings.neurips.cc/paper_files/paper/2024/file/639d992f819c2b40387d4d5170b8ffd7-Paper-Conference.pdf. https://doi.org/10.52202/079017-1754
[55] A.L. Zhang, T. Kraska, O. Khattab, Recursive language models, CoRR abs/2512.24601 (2025). https://doi.org/10.48550/arXiv.2512.24601
[56] A. Plaat, A. Wong, S. Verberne, J. Broekens, N. van Stein, T. Bäck, Multi-step reasoning with large language models: a survey, ACM Comput. Surv. 58 (6) (2025) 1–35. https://doi.org/10.1145/3774896
[57] A. P. Martinich (Ed.), The Philosophy of Language, Oxford University Press, New York, 1985.