Characterizing and assessing robust conversational interaction: Amendments in order specification

Simon Benigeri and Larry Birnbaum
Department of Computer Science
Northwestern University
Evanston, IL 60208 USA
{simon.benigeri, l-birnbaum}@northwestern.edu

Abstract

We assess the robustness of large language models (LLMs) on a simple task—fast-food order specification—in the face of user-initiated amendments. In amending an order, users must communicate, explicitly or implicitly, that a change is being made, the type of change, and what is being changed. We generate synthetic food orders with single amendments, controlling the type of amendment, which components are implicitly or incompletely specified, and the pragmatic inferences required to resolve them. On a dataset broadly covering the space of possible combinations, 7 flagship models average 97.7% accuracy. On a more targeted dataset with denser coverage of empirically difficult patterns, accuracy drops to 93.4%, with the hardest combinations dropping below 80% accuracy. We also test two further prompting methods: generating an order before interpreting amendments in a turn-taking interaction; and function-calling to parameterize explicit edit operations on the order itself. These somewhat improve performance (+0.2 pp and +3.2 pp respectively), but neither entirely eliminates the failure modes. We also find that asking clarification questions about implicit components improves accuracy on a subset of difficult examples (+5.4 pp), but models ask on only 11% of examples. Given the task's deliberate simplicity—small fixed menu, a single standard order, with a single unambiguous amendment—these results illustrate the challenge of robustly performing the pragmatic inferences that real-world conversational interaction demands.

1 Introduction

1.1 Motivation

Conversational interfaces for even the simplest practical tasks must take account of user goals, manage context, and apply task- and domain-specific constraints to correctly interpret input language (Budzianowski et al., 2018; Jurafsky and Martin, 2021). LLMs are increasingly deployed in such systems (Ouyang et al., 2022), but real-world interactions often challenge them. McDonald's, for example, discontinued an LLM-based drive-thru assistant after encountering errors such as failing to apply order changes and mishandling item combinations (Grantham-Philips, 2023; Kaplan, 2024; Bregel, 2024).

Such failures indicate a robustness gap in the face of natural conversational behaviors. In this work, we focus on one such behavior, user-initiated amendments to previously stated information, in a simple food order specification task.

1.2 Problem definition

Amendments in conversational interaction are common, driven by misspecified intent, shifting goals, or efforts to establish common ground (Sacks et al., 1978; Schegloff et al., 1977; Clark and Brennan, 1991; Clark, 1996), and interpreting them often requires pragmatic inference in the Gricean sense (Grice, 1975). For example, given an order of a cheeseburger, large fries, and a large Coca-Cola, the utterance "Wait, I'll have a Sprite instead," implies that the Sprite substitutes for the Coca-Cola (the only drink) and that it is sized large—because the speaker would have specified if otherwise.

In our setting, a user places a single standard food order from a small menu (∼30 items inspired by McDonald's (McDonald's Corporation, 2024)): a main, a side, a drink, and optionally a dessert. The user then provides a single, unambiguous amendment to that order. Given both utterances, the model must output the correct final order.

Interpreting an amendment requires establishing several facts: whether a change is being made; the type of change (addition, modification, etc.); which aspects or parameters of the order are affected (items, attributes); and the updated values. Users may state these completely and explicitly, or incompletely or implicitly in various ways, and exactly how a component is incompletely or im-

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 29321–29360
October 24-29, 2026 ©2026 Association for Computational Linguistics

plicitly specified determines the nature of the inferences required. Identifying which item is being changed requires *deictic reference resolution* when pointed to ("Make *that* a Sprite"), but *zero anaphora resolution* when omitted entirely ("I'll have a Sprite instead"). Similarly, inferring a new item's size from the replaced item's is a type of *inheritance* inference, with the original item serving as the "anchor" from which it is inherited.

1.3 Method

To systematically assess the abilities of LLMs to correctly perform these sorts of inferences (formalized in Section 3), we design structured templates to generate initial orders and amendments. The values used to configure each template control which components are implicitly or incompletely specified, and how, enabling us to systematically examine specific inferences and their combinations. Evaluation using such an artificially constructed dataset of this sort cannot, obviously, yield results reflecting real-world deployment (although it might indicate certain trends). Our goal rather is to characterize the space in a way that enables us to zero in on specific factors that pose difficulties for current models.

We evaluate in two phases, focusing our analysis primarily (although not exclusively) on a subset of 7 high-performing *flagship* models. A broad baseline of 23 models is first used in conjunction with a set of templates intended to cover the space of inference patterns, in order to identify empirically difficult inference profiles. A targeted evaluation (15 models) then stress-tests these profiles with denser sampling. Using the targeted dataset, we also compare additional prompting methods: a *two-step* approach that generates the order state before interpreting the amendment in a turn-taking paradigm, and *function calling* to parameterize explicit edit operations on the order itself (Section 4). On a subset of difficult examples, we also test a turn-taking setting where models can ask for clarification on implied or missing components.

1.4 Results and implications

Because each example has a known set of implicit components, failures can be mapped to the specific pragmatic inferences required. Flagship models[1] achieve 97.7% on the broad baseline, yet make errors even on trivially easy examples. For instance, given the order "a quarter pounder with cheese, large fries, and one small water" and the amendment "I'd like one more water. Make that medium," the correct interpretation is an addition of a medium water—but models including GPT-4o and Claude Sonnet 4.5 instead modify the existing water's size.

On the targeted dataset, flagship accuracy drops to 93.4%. Some components are harder to infer when implicit (target item: -4.3 pp vs. explicit; new item size: -13.1 pp), certain inference types are disproportionately difficult (deictic reference: 92.5%; inheritance: 83.7%), and the amendment type impacts difficulty further (inheritance + substitution: 79.9% vs. inheritance + addition: 89.0%). A 6.0-point gap between pass@k and pass^k (k=5) shows that models not only struggle with specific inferences but fail to make them consistently.

Two-step prompting yields +0.2 pp accuracy; function calling +3.2 pp—neither eliminates these failure modes (Section 4). Clarification dialogue improves accuracy by +5.4 pp, but models ask on only 11% of examples—and only 26% of the time when clarification is objectively required. The larger issue is *whether* they ask at all (Section 5).

Given the task's deliberate simplicity, these results expose a lack of robustness in the pragmatic inferences that real-world conversational interaction demands, where more complex orders, multiple amendments, and ambiguity co-occur.

Our contributions are: (1) a framework that decomposes amendments into components and maps how each is expressed to the inferences they require, together with a system for generating controlled test examples; (2) a broad evaluation of models showing that specific inferences remain disproportionately hard even on a deliberately simple task; (3) a comparison of prompting methods showing that none eliminates these failure modes; and (4) evidence that models can ask effective clarification questions but rarely do so, even when asking would clearly help, suggesting that improving this ability would directly benefit robustness in the face of a common but challenging conversational behavior: user-initiated amendments.

2 Related Work

While user-initiated amendments are common in conversation (Schegloff et al., 1977; Sacks et al., 1978; Clark and Brennan, 1991), LLM evaluation has largely overlooked them (Budzianowski et al., 2018; Zang et al., 2020). Balaraman et al. (2023)

[1]Metrics are averaged across the 7 flagship models (k=5 runs per example) unless noted; see Section 4 for the full list.

showed that GPT-3 struggled with third-person repair in conversational QA; Kim et al. (2022) found that synthetic amendments to MultiWOZ degraded dialogue-state tracking performance; and Lin et al. (2020) simulated user corrections in training data to improve model flexibility. Tian et al. (2021) address *model-initiated* corrections to dialogue state. Recent multi-turn evaluations reinforce the challenge: Laban et al. (2026) find that LLM performance degrades when instructions are communicated across turns, and Sirdeshmukh et al. (2025) show that a revision task which involves referential inferences over versioned edits, challenges frontier models. Our work focuses on a simpler setting and drills down on the specific pragmatic inferences that *user-initiated* amendments require.

Interpreting implicit or incompletely specified amendments requires Gricean inference (Grice, 1975): speakers assume listeners will derive unstated information from context, guided by maxims of quantity, quality, relevance, and manner. Recent work has begun evaluating LLMs on such reasoning: Yue et al. (2024) find that while GPT-4 reaches human-level accuracy on conversational implicature, most models lag far behind; Cong (2024) reports only minimal sensitivity to manner implicatures; and Krause and Vossen (2024) survey the broader role of Gricean maxims in NLP. These studies test whether LLMs can identify implicatures; we test whether and under what conditions they can apply them, specifically in order to properly interpret amendments.

To precisely probe the conditions under which models function robustly, we use controlled, synthetically generated datasets. Similar approaches to assess LLM capabilities include template-based workplace benchmarks (Styles et al., 2024), controlled error insertion for flaw detection (Xi et al., 2025), and synthetic data generation framed as mechanism design (Davidson et al., 2026).

3 Methodology

3.1 Task Setup

Given an order-amendment utterance pair, the model must produce the amended order as JSON (Appendix B.1):

> **Order:** "I'll have a cheeseburger, medium fries, and a medium Coca-Cola."
> **Amendment:** "Wait, make that a Sprite instead."
> **Output:** `[(cheeseburger, main, null, []), (fries, side, medium, []), (Sprite, drink, medium, [])]`

3.2 Generating Food Orders

Orders are sampled from a fixed menu (∼30 items) inspired by McDonald's (McDonald's Corporation, 2024) (Appendix A). Each consists of a main, a side, a drink, and optionally a dessert. Items may have available sizes and removable ingredients (e.g., a cheeseburger is a main with removable pickles, onions, and ketchup; fries are a side available in small, medium, or large). The order JSON specifies each item's name, category, size, and modifications. Order utterances are generated from templates that join an order intent (e.g., "I'll have") with phrases denoting items and their fields (Appendix B.1).

3.3 Amending Food Orders

Each example additionally involves one amendment, drawn from one of four edit operations: *deletion*, *addition*, *modification*, and *substitution*. The menu defines each item's category and the values its fields admit, constraining valid amendments (e.g., substitute items are sampled within the same category—a drink for a drink, items without size options cannot have their size changed, etc.).

An amendment is characterized by several components (Table 1): whether a change is being offered, the type of change, the target of the change (the item being changed, or for additions, the item being added), the relevant field, the field's new value, and—for substitutions and additions—the new item and its size.

Amendment utterances are generated from structured templates (Table 1). The values sampled from the menu and the template choices (e.g., whether to mention size) control which components are stated explicitly and which are left implicit. For example, "Replace my Coca-Cola with a medium Sprite" explicitly states the amendment type (substitution), target item (Coca-Cola), new item (Sprite), and new item size (medium). In contrast, "I want a Sprite instead" leaves amendment type, target item, and new item size implicit; resolving them requires domain knowledge from the menu—e.g., the Sprite must be replacing the Coca-Cola, not the cheeseburger or fries. All generated examples have exactly one correct interpretation (Appendix B.2).

3.4 Characterizing Inferences

Each component takes one of three values: *N/A* (not applicable for the operation), *explicit*, or *implicit*. Exactly *how* a component is left implicit determines the inference type required to resolve

Component	Operation				Inference required	Explicit utterance / Implicit utterances
	Del	Add	Mod	Sub		
amendment offer	✓	✓	✓	✓	—	I'm changing my order. Remove the fries. / Remove the fries.
amendment type	✓	✓	✓	✓	—	*Remove* the fries. . . / *I don't want* the fries.
target item	✓	✓‡	✓	✓	deictic ref.; indirect ref.; zero anaphora	. . . the *Cheeseburger* without pickles. / . . . *that* without pickles.; . . . the *burger* without pickles.; Wait...No pickles.
target field	—	✓	✓	—	—	Make my fries a larger *size*. / Make my fries larger.
target field value	—	✓†	✓†	—	directional	Change the fries to *large*. / Go up a size on the fries.
new item‡	—	—	—	✓	(always explicit)	Replace my Coca-Cola with a *Sprite*.
new item size	—	✓†	—	✓†	inheritance; directional	Add a *medium* fries. / Add another of those fries.; Replace my Coca-Cola with a *medium* Sprite. / . . . with a Sprite.; Add a *medium* fries. / Add another fries but *larger*.

Table 1: Amendment components and inferences required when implicit. Operation columns show the amendment type: Del = deletion, Add = addition, Mod = modification, Sub = substitution. Assumed order: cheeseburger, medium fries, medium Coca-Cola. Shaded rows = components prioritized for targeted dataset. † Whether explicit or implicit depends on the sampled items. ‡ For addition, target item is the item being added.

it. Inferring the target item of a modification, for example, may require *deictic reference* resolution ("Make *that* without pickles"), *indirect reference* ("the *burger* without pickles"), or *zero anaphora* ("Remove the pickles please"); the new item's size may require *property inheritance* from an anchor item, namely the item it is replacing ("I want a Sprite instead," replacing the Coca-Cola). The target field value may require *directional inference*: the user specifies a direction (+1 or −1 on the menu's ordered size scale) rather than an absolute value. "Go up a size on the fries" requires resolving medium to large given the scale {small, medium, large}.[2] Not all components are equally difficult when implicit: some are trivially resolvable, while others show significant accuracy drops. The focus of our evaluation is on identifying the components and inference types that empirically prove difficult (Appendix C.2).

Because a single amendment can leave multiple components implicit, each instance can require a *combination* of inferences that may compound in difficulty. We profile each example along three dimensions: the operation or type of amendment; which components are implicit; and the inference types required to resolve them. These profiles define the dimensions used both to generate the targeted dataset (§3.5), and to isolate combinations which pose difficulties (Section 4).

[2]This inference often co-occurs with zero anaphora on the field itself—"Make the fries larger" leaves both the field (size) and its value implicit, requiring the model to infer the field via zero anaphora and compute the resulting value directionally.

3.5 Datasets

We construct two datasets, the first to map the space, and the second to drill down and more clearly identify difficult aspects of the task.

Phase 1: Broad dataset (711 examples). Generated from all 624 templates (Appendix B.2), this dataset samples broadly across amendment types and inference requirements. Utterances are generated by uniformly sampling from the pool of templates and parameterizing them with values from the menu (Appendix C.1).

Phase 2: Targeted dataset (720 examples). Phase 1 results (Section 4) help identify which components and inference types are empirically difficult or worth testing further, but some combinations are represented too sparsely for reliable performance estimates. The targeted dataset prioritizes profiles built around these patterns with denser sampling.

Three components show significant accuracy drops in at least one implicit setting (shaded rows in Table 1): target item, new item size, and target field value. We ensure denser coverage of all inference types for these three components—including those that appear easy in the broad dataset—to determine which specific inferences drive difficulty. The remaining four components show smaller gaps; they are still represented in the targeted dataset but we do not prioritize their sampling or drill down by inference type (Appendix C.2).

Profiles pair each operation (amendment type)

with specific inference types for the relevant order components (e.g., substitution with deictic reference for target item, or property inheritance from an anchor for new item size), yielding 33 active profiles grouped into three bands by the number of amendment components incompletely or implicitly specified: 2 (30 examples each, testing whether combinations compound difficulty), 1 (20 each, ensuring coverage of each inference type), and 0 (baseline, 10 each), plus 50 no-amendment orders (Appendix C.2).

To test whether models ask for clarification (§5.2), we also draw a subset (n=324) and generate 50 underspecified examples.

Quality control. No two examples share the same order–utterance pair. After generating the datasets, we systematically exclude examples the ground truth of which cannot be reliably determined from the utterance alone—e.g., genuinely ambiguous references to the target item, disfluent utterances, etc. All dataset sizes above reflect post-exclusion counts (Appendix C.3).[3]

4 Experiments

4.1 Setup

We test a set of models across the datasets described in Section 3.5. The broad baseline helps identify which inference patterns (combinations of what needs to be inferred, and how) are empirically hard or worth testing further. The targeted dataset stress-tests these with denser sampling, and we then test additional approaches for performance gains.

Models. We evaluate 23 models spanning 7 providers on the broad dataset; 15 are tested on the targeted dataset.[4] To evaluate the state of the art, our analysis focuses on a subset of 7 high-performing *flagship* models from different families: GPT-5, GPT-4o, Claude Opus 4.6, Claude Sonnet 4.5, Gemini 2.5 Pro, DeepSeek v3.2, and Hermes 3 405B. (Full per-model results are in Appendix D; Implementation details in Appendix I.1.)

Prompting methods. All methods receive the same base context: instructions, the menu, the order and amendment utterances (Appendix I.2). The menu provides syntactic and semantic constraints (item categories, sizes, removable ingredients). **Standard**, the baseline, instructs the model to generate the final order JSON. We test two more methods on the targeted dataset. **Two-step** splits the task across two turns: the model first generates the order, then processes the amendment. **Function calling (FC)** (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024) requires the model to, given the order JSON, output a parameterized edit operation (`add`, `delete`, `modify`, or `substitute`); a deterministic executor then applies the returned operation to the current order to produce the final result. **Simulated FC (Sim. FC)** defines the operations and their constraints in the prompt (identical across models); **Native FC** enforces them through each provider's tool-calling API. Because Native FC implementation details vary across providers (Appendix I.1), we use Sim. FC as the primary experimental condition for function calling.

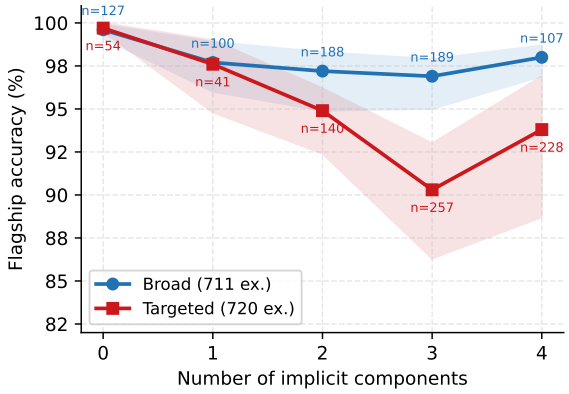

Figure 1: Avg. flagship accuracy (k=5 runs) vs. number of implicit components, Standard prompt on broad vs. targeted datasets. Shaded bands show the reliability gap. The accuracy drop in the targeted dataset shows its focus on empirically difficult inference patterns.

Evaluation. Each model is run k=5 times per example at default temperature. We report **average accuracy**, **pass@k**, **pass^k**, and the **reliability gap** (pass^k − pass@k). An output is correct if the generated JSON exactly matches the ground truth after normalization. Unless noted otherwise, metrics are averaged across *flagship* models.

4.2 Broad Baseline

Overall performance. Flagship models average 97.7% accuracy on the broad dataset (Table 2; full leaderboard in Appendix D), but a 2.5-point reli-

[3]Code and data are included as supplementary material and at `https://github.com/simon-benigeri/food-order-amendments`, released under CC-BY 4.0.

[4]OpenAI (OpenAI, 2025b,a, 2024), Anthropic (Anthropic, 2026, 2025b,a), Google (Google DeepMind, 2025), Meta/Nous (Grattafiori et al., 2024; Teknium et al., 2024), DeepSeek (DeepSeek-AI, 2024), and MiniMax (MiniMax, 2026).

ability gap shows that models that *can* solve an example don't always do so reliably. As one would expect, accuracy generally declines as more components are left implicit (Figure 1): from 99.6% at n=0 to 96.9% at n=3, with the reliability gap widening from 0.8 to 2.9 points. Interestingly, at n=4 both recover. Some n=4 examples are quite difficult ("*I want a Sprite instead of that*"—implicit offer, amendment type, target item, and new item size) while others are trivially resolvable ("*No pickles on that*"—the same four components).[5] What matters is not *how many* components are implicit, but *which ones are* and *how* they must be inferred.[6]

Difficulty is driven by component *and* inference. Table 3 breaks accuracy down by component (*what needs to be inferred*) and inference type (*how*). For **new item size**—what size the added or substituted item should be—property inheritance from an appropriate anchor (specifically, inferring the size from the existing item) drops flagship accuracy to 83.7% (n=55), 13.2 points below the explicit baseline. For **target item**—which item is being changed—deictic reference ("make *that* a Sprite") yields 92.5% (n=44), 4.3 points below the explicit baseline, while indirect reference is easy (99.0%, n=176). For **target field value**, directional inference ("go up a size on those fries"—i.e., modifying medium fries to large) is easy (99.0%, n=123, vs. 99.7% explicit). These patterns motivate the targeted dataset, which samples challenging component–inference pairs more densely (Appendix C.2).

4.3 Targeted Evaluation

Denser sampling sharpens the patterns. On the targeted dataset, with denser coverage of the empirically difficult combinations of target item, new item size, and target field value—and the inferences they require—as determined from the results above, flagship accuracy drops to 93.4% and the reliability gap widens to 6.0 points (Table 2; Figure 1). The patterns sharpen: property inheritance remains the hardest inference (82.7%, −1.0 pp, n=160); deictic reference drops to 89.0% (−3.5 pp, n=90); and—notably—indirect reference, easy on the broad dataset (99.0%), drops to 94.4% (−4.6 pp, n=300) for target item. Zero anaphora drops to 94.2% (−3.8 pp, n=90) (Table 3).

Dataset / Method	Acc [min–max]	pass@k	pass^k
Broad (711)			
Standard	97.7 [97.1–98.2]	98.6	96.1
Targeted (720)			
Standard	93.4 [91.9–96.2]	95.7	89.7
Two-step	93.6 [91.5–95.7]	95.5	90.6
Sim. FC	96.6 [94.2–97.7]	98.3	93.4
Native FC†	94.6 [90.2–98.0]	97.4	91.2

Table 2: Avg. flagship model performance across datasets and methods (k=5 runs). Acc shows mean [min–max]. †Excluded due to tool-calling API issues. Full per-model results in Table 14 and Table 15.

		Broad		Targeted	
Component	**Inference**	n	**Acc**	n	**Acc**
target item	—	296	96.8	190	92.1
	deictic ref.	44	92.5	90	89.0
	zero anaphora	67	98.0	90	94.2
	indirect ref.	176	99.0	300	94.4
new item size	—	106	96.9	140	91.2
	inheritance	55	83.7	160	82.7
	directional	—	—	100	98.3
target field value	—	197	99.7	140	98.0
	directional	123	99.0	200	98.5
All explicit		27	98.5	4	97.2
No amendment		100	99.9	50	99.9

Table 3: Per-component avg. flagship accuracy (k=5) on broad (n=711) and targeted (n=720) datasets. — = component is explicit. Full breakdown in Table 11. The targeted dataset oversamples hard inferences.

Amendment type further impacts difficulty. Table 4 reveals that the amendment type further compounds difficulty. Inheritance drops to 79.9% under substitution (e.g., "*Replace my Coca-cola with a Sprite*") but rises to 89.0% under addition (e.g., "*Add another of those fries*").[7] Deictic reference is far harder under substitution (80.7%, e.g., "*Make that a Big Mac*") than modification (99.3%, e.g., "*Make that without pickles*"). Indirect reference—whether synecdoche ("*the burger*", "*my drink*") or lexical paraphrase ("*the coke*")—is hard across *multiple* amendment types: substitution (90.0%), addition with a size change (91.5%), ingredient modification (93.3%), and deletion

[5] We speculate that being explicit about components like amendment offer, which are normally left implicit may itself cause confusion—e.g., "I'm changing my order. Change my cheeseburger to a Big Mac" is over-emphatic compared to simply "Change my cheeseburger to a Big Mac."

[6] At n=3 we simply have a higher proportion of hard inferences than at n=4; cf. Table 12, Appendix C.1.

[7] Interestingly, "*Add another of those fries, but larger*" (which adds directional size inference) is easier, because the directional cue draws attention to the anchor's size.

Inference (comp.)	Amend. type	n	Acc. Std.	2-st.	Sim.
inheritance (size)	substitution	110	79.9	75.8	90.5
deictic ref. (target)	substitution	50	80.7	69.7	87.5
inheritance (size)	add (same size)	50	89.0	94.7	94.4
zero anaphora (target)	substitution	50	89.7	93.9	98.8
indirect ref. (target)	substitution	50	90.0	87.6	95.8
	add (diff. size)	80	91.5	93.9	92.5
	modify (ingr.)	20	93.3	92.4	94.6
	delete	20	93.7	96.9	98.1

Table 4: Hardest inference×amendment type pairs: avg. flagship accuracy on the targeted dataset (k=5, n=720). Std. = Standard, 2-st. = Two-step, Sim. = Simulated FC. Comp. = implicit component (size = new item size, target = target item). All remaining combinations ≥97%; full breakdown in Table 19.

(93.7%). All remaining inference×amendment type combinations exceed 97% (Appendix F).

4.4 What Methods Help?

Sim. FC helps universally; Two-step can hurt. Table 4 shows that Sim. FC improves *every* hard pair: inheritance+substitution (90.5%, +10.6 pp), deictic+substitution (87.5%, +6.8 pp). Overall, Sim. FC averages 96.6% (+3.2 pp), with Claude Sonnet 4.5 reaching 97.7%. Two-step, by contrast, *worsens* the two hardest cases: deictic+substitution (69.7%, −11 pp), inheritance+substitution (75.8%, −4.1 pp). It helps moderate cases, e.g., zero anaphora+substitution (93.9%, +4.2 pp), but averages only 93.6% (+0.2 pp), with inconsistent per-model effects. Native FC averages 94.6% for the 5 flagships with successfully queried APIs; on the same 5 models, Sim. FC reaches 97.1% vs. Standard's 92.9%, confirming that the prompt-based FC approach outperforms native tool-calling.[8] Full per-model results are in Appendix D (Table 15).

Why does Sim. FC help? Two-step prompting *also* provides the order as explicit context—the model generates it first, and does so accurately and reliably[9]—but only shows minimal improvement (+0.2 pp). Sim. FC, by contrast, improves by +3.2 pp. Since both methods explicitly represent the current order JSON, the gain must come from explicitly identifying and parameterizing edit operations (`add`, `delete`, `modify`, or `substitute`). This maps directly onto the task decomposition we describe in Section 1: interpreting an amendment requires establishing whether a change is being made, the type of change, which aspects of the order are affected, and the updated values. Function calling forces the model to make each of these decisions explicitly (e.g., selecting an operation type, identifying the target item, relevant fields, and specifying new values) rather than jumping straight from utterance to final order JSON. Additionally, the function specifications constrain the model's output space—e.g., restricting operations to valid types and scoping each call to a single item edit—which reduces errors such as applying multiple edits or replacing the entire order with a single item.

Reliability gap persists across methods. For flagship models, the gap remains 6.0 points (Standard), 4.9 (Two-step), 4.9 (Sim. FC). The hardest pairs (inheritance+substitution, deictic+substitution) exceed 16 points under Standard; Sim. FC reduces this to 11–12 points but does not eliminate it (Table 19, Appendix F). Easy combinations have gaps near zero—the same inferences that are hard drive unreliability.

Effects of model scale. Mid-tier models (Haiku 4.5, Gemini Flash) perform near flagship levels, but smaller models degrade sharply under Standard (accuracy / reliability gap): GPT-5-nano (83.0% / 16.9), Llama 3.1 8B (67.7% / 32.6) vs. flagships (93.4% / 6.0). The ranking of inference difficulty is stable across scales: the same inferences that are hard for flagships remain hardest for smaller models (Appendix D, Table 16). Sim. FC helps flagships modestly (+3.2 pp / −1.1) but disproportionately benefits smaller models: GPT-5-nano (+9.7 pp / −8.3), Llama 3.1 8B (+13.9 pp / −4.4). Two-step, ineffective for flagships (+0.2 pp / −1.1), helps the smallest model, Llama 3.1 8B, in accuracy (+10.5 pp) but less so in reliability (−0.4). These approaches may be relevant for deployment in resource-constrained settings (e.g., mobile phones), particularly for tasks such as fast-food ordering.

5 Analysis of Errors and Interventions

5.1 Error Analysis

Since each example has a known edit operation and structured ground truth, we classify errors

[8] The function descriptions are similar in both settings, making this gap slightly unexpected. We analyze this in Appendix D)

[9] On the broad dataset, order-only accuracy is 99.9% (Table 3); pass^k = 99.5% (Table 18).

deterministically by comparing predicted and expected orders, and analyze them through the lens of amendment type, implicit components, and inference types. For flagship models under Standard on the targeted dataset, 201 examples (27.9%) have at least one failure (full taxonomy in Appendix G).

Inheritance (new item size). When the new item's size must be inferred from an anchor item, models often omit or hallucinate it—despite the anchor's size being explicit in the order (and sometimes the utterance) and the menu requiring one. The dominant failure is *wrong/missing size on replacement* (61 examples, 30.3% of errors): e.g., "No small Sprite. I'll have a Coca-Cola instead"—all 7 flagships output Coca-Cola without inheriting *small*. Under addition, 35 examples (17.4%) produce *wrong/missing size on new item*—e.g., "Add another order of those fries" yields fries with no size, though the order contains **small** fries.

Target resolution (target item). When the target is referenced implicitly—via synecdoche, lexical paraphrase, deixis, or zero anaphora—34 examples (16.9%) produce *added item missing*: e.g., "Add one more of that drink" requires resolving "drink" to "Sprite"; models omit the item entirely (0% accuracy on this example).

Implicit amendment type. When substitution is phrased similarly to addition, 24 examples (11.9%) produce *both old and new present*: e.g., "Actually I want a large Coca-Cola instead" yields both water and Coca-Cola—cues like "instead" or "actually" are insufficient when the target is unnamed.

Scope misinterpretation. In 30 further examples, models misinterpret the amendment's scope: applying a size change to multiple items (14), replacing the entire order with a single item (11), or dropping an unrelated item (5).

5.2 Clarification

Given the errors identified above, we test a turn-taking setting: the Standard prompt is extended with an optional `ask_clarification` function call. If invoked, the model lists which components it considers underspecified, receives their values in explicit natural language, and then outputs the order based on that. Otherwise, it outputs the order directly.

Dataset. We evaluate on a *clarification subset* (324 examples from the targeted dataset: 145 where models erred, 100 controls, 79 for component coverage) and, separately, on 50 *underspecified examples* with genuinely ambiguous utterances (e.g., "Another order of that. But small this time," where multiple items could be the referent of "that"). The full protocol is in Appendix H.

Model	Acc	ΔStd	Clar%	R|ask	Prec
Clarification subset (324 examples)					
Gemini 2.5 Pro	96.7	+13.5	13.5	54.5	98.9
GPT-5	94.1	+12.0	16.7	58.6	97.0
GPT-4o	90.6	+1.9	18.5	51.2	81.1
Claude Opus 4.6	90.7	+6.8	4.8	62.4	100.0
Claude Sonnet 4.5	89.8	+3.5	12.0	52.1	91.8
Hermes 3 405B	93.8	+2.3	7.2	44.0	87.3
DeepSeek V3.2	85.9	-2.1	4.0	43.5	75.7
Average	91.7	+5.4	11.0	52.3	90.2
Underspecified examples (50)					
Gemini 2.5 Pro	44.4	+24.4	26.4	40.5	100.0
GPT-5	45.2	+14.4	28.4	40.8	94.9
GPT-4o	32.8	+31.2	34.0	43.4	71.5
Claude Opus 4.6	32.0	+2.8	19.2	42.9	100.0
Claude Sonnet 4.5	52.0	+45.2	58.0	41.0	65.9
Hermes 3 405B	29.2	+15.2	14.0	47.1	87.5
DeepSeek V3.2	11.2	+5.6	3.2	42.5	53.3
Average	35.3	+19.8	26.2	42.6	81.9

Table 5: Clarification results for flagship models, averaged over k=5 runs. ΔStd = change vs. Standard (pp); Clar% = clarification rate; R|ask = component recall when asking; Prec = component precision. On underspecified examples, clarification is always warranted.

Results. Clarification improves accuracy by +5.4 pp on the clarification subset and +19.8 pp on underspecified examples (Table 5). When models ask, they identify the correct components (precision >90%) but miss half (recall ∼52%). The larger issue is *whether* they ask at all: only 11% of clarification examples and 26% of underspecified ones. Effects vary substantially by model (Table 5). There is an inherent trade-off: asking about every implicit parameter is impractical, but proceeding without asking leads to the errors above.

We consider three factors to understand the low clarification rate. **First, it is model-specific:** the same prompt yields very different rates across models—on the clarification subset, Opus asks on only 4.8% of examples while GPT-5 asks on 16.7% (Table 5; Appendix H). The variation is even wider on genuinely ambiguous examples: 3.2% (DeepSeek) to 58.0% (Sonnet). This suggests model-specific behavior, most likely from training. **Second, detection is partial:** when models do ask, they identify the correct components with high precision (>90% on the clarification subset,

84% on genuinely ambiguous examples), but recall is low (∼52% and ∼41%, respectively), meaning they miss about half the components they should ask about. The near-zero unnecessary question rate (≤0.8%; Appendix H) on fully explicit examples shows models *can* distinguish ambiguous from unambiguous inputs, but detecting ambiguity is clearly part of the issue. **Third, there is a high threshold to act:** even on genuinely ambiguous examples, models ask only 26% of the time on average.

Laban et al. (2026) find that LLMs make assumptions early in conversation and cannot recover; Rabanser et al. (2026) observe that models do not behave differently on ambiguous vs. unambiguous inputs. Both align with our finding: models that could benefit from asking choose not to. Learning to recognize when information is implicit or ambiguous and then requesting it could improve robustness, though this recognition is itself a pragmatic reasoning challenge.

6 Conclusion

Our task is deliberately simple: one speaker, standard orders from a small fixed menu, a single unambiguous amendment. Models are given the full menu and a structured output format—all the information they need. Yet certain combinations of amendment type (operation), implicit component, and inference type remain challenging (83–89% flagship accuracy, below 80% at the worst) across all models and methods we tested. This is our key finding; and in our view, the fact that these three dimensions define a space in which errors cluster in specific combinations serves to validate the utility of characterizing amendments in this way. We also find that while explicitly providing the set of operations and required parameters via function calling improves performance (+3.2 pp), it does not entirely eliminate these failures. The challenge is pragmatic, not architectural.

More realistic conversational interactions would compound every factor we isolate: consider complex and misspecified orders, multiple orders, multiple amendments, amending amendments, larger or changing menus, or greater ambiguity. Current models already fail in many cases in this highly controlled setting—and fail to recognize when they lack information (asking on only 11% of examples, despite high precision when they do). It's very likely that these failures would interact and amplify in practice. The errors we observe are predictable: each combination of operation, implicit component, and inference produces a characteristic failure mode. This suggests the problem is systematic, not incidental, and that developing robust task-oriented conversational interfaces will require further effort in characterizing how natural conversational behaviors challenge model reasoning in order to develop viable solutions. The work reported here provides one step in that direction.

Limitations

Data. Our data is entirely synthetic and template-generated; no real user interactions were collected. While templates enable us to control for specific inferences, they cannot capture the full range of natural language variation (disfluencies, hedging, culturally specific phrasing). Designing templates and configurations that isolate specific inferences is tedious and may not scale easily to new domains or inference types. The process is also not error-free: several rounds of data review were needed, producing the exclusion lists described in Appendix C.3. Developing better ways to devise synthetic datasets that systematically exercise certain inference types remains an important challenge.

Scope. The evaluation covers a single task and domain (fast-food ordering) with a fixed menu of ∼30 items across 4 categories. Each example contains exactly one amendment; real conversations involve longer and more complex orders, multiple sequential amendments, and corrections to corrections. The inference challenges we profile, while empirically difficult, are still only a subset of what real amendments require—we do not test cases where users treat distinct menu items as variants of one item ("Make that chocolate" for a shake, expecting Vanilla Shake → Chocolate Shake), shorthand upgrades ("Make that a double" for cheeseburger → double cheeseburger), or inferences a helpful interlocutor might make (e.g., suggesting a combo to save money when the items already constitute one, or suggesting an item upgrade to benefit from a combo). Item sizes are simplified to a three-point scale (e.g., small, medium, large), and ingredients can only be removed from items, which means inferences relating modifications are kept straightforward. We also do not vary the menu itself to test whether models adapt to menu scale, different item relationships, and constraints.

Clarification experiment. The clarification subset (324 examples) is smaller than the broad and targeted datasets. The underspecified set contains only 50 examples, limiting statistical power even with k=5 runs, and tests only one type of ambiguity (ambiguous deictic reference to the target item). Other forms of genuine ambiguity were not the focus of this study and so are not covered.

API. DeepInfra API errors during our function-calling experiments prevented reliable evaluation of native tool-calling for Hermes and DeepSeek models; their Native FC results are therefore not included in the main text.

Acknowledgements

We would like to thank Doug Downey and David Demeter for insightful discussions. This work was supported in part by a gift from Adobe Research; by the National Science Foundation under grant IIS-2006851; and by Northwestern University's McCormick School of Engineering through a Walter P. Murphy Fellowship. AI coding tools (Cursor) were used under author supervision for software engineering tasks and limited editorial assistance; all scientific contributions are the authors' own.

References

Anthropic. 2025a. Claude haiku 4.5 system card. https://www.anthropic.com/claude-haiku-4-5-system-card.

Anthropic. 2025b. Claude sonnet 4.5 system card. https://www.anthropic.com/claude-sonnet-4-5-system-card.

Anthropic. 2026. Claude opus 4.6 system card. https://www-cdn.anthropic.com/14e4fb01875d2a69f646fa5e574dea2b1c0ff7b5.pdf.

Vevake Balaraman, Arash Eshghi, Ioannis Konstas, and Ioannis Papaioannou. 2023. No that's not what i meant: Handling third position repair in conversational question answering. *Preprint*, arXiv:2307.16689.

Sarah Bregel. 2024. Bacon ice cream: McDonald's AI drive-thru ordering full of glitches. *Fast Company*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Conference on Empirical Methods in Natural Language Processing*.

Herbert H. Clark. 1996. Common ground. In *Using Language*, chapter 4. Cambridge University Press, Cambridge.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley, editors, *Perspectives on socially shared cognition*, pages 127–149. American Psychological Association.

Yan Cong. 2024. Manner implicatures in large language models. *Scientific Reports*, 14(1):29113.

Tim R. Davidson, Benoit Seguin, Enrico Bacis, Cesar Ilharco, and Hamza Harkous. 2026. Reasoning-driven synthetic data generation and evaluation. *Transactions on Machine Learning Research*. J2C Certification.

DeepSeek-AI. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Google DeepMind. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf.

Wyatte Grantham-Philips. 2023. Mcdonald's, ibm test drive-thru ai, but face unexpected challenges. *AP News*.

Aaron Grattafiori, Abhimanyu Dubey, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

H. P. Grice. 1975. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Volume 3, Speech Acts*, pages 41–58. Academic Press, New York. http://www.ucl.ac.uk/ls/studypacks/Grice-Logic.pdf.

Daniel Jurafsky and James H. Martin. 2021. Chatbots & dialogue systems. Chapter 24, *Speech and Language Processing*, 3rd ed. draft of December 29, 2021.

Anna Kaplan. 2024. Mcdonald's gets rid of AI drive-thru ordering after botched orders. *Today.com*.

Takyoung Kim, Yukyung Lee, Hoonsang Yoon, Pilsung Kang, Junseong Bang, and Misuk Kim. 2022. Oh my mistake!: Toward realistic dialogue state tracking including turnback utterances. In *Proceedings of the Towards Semi-Supervised and Reinforced Task-Oriented Dialog Systems (SereTOD)*, pages 1–12, Abu Dhabi, Beijing (Hybrid). Association for Computational Linguistics.

Lea Krause and Piek T.J.M. Vossen. 2024. The Gricean maxims in NLP - a survey. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 470–485, Tokyo, Japan. Association for Computational Linguistics.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. LLMs get lost in multi-turn conversation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chien-Wei Lin, Vincent Auvray, Daniel Elkind, Arijit Biswas, Maryam Fazel-Zarandi, Nehal Belgamwar, Shubhra Chandra, Matt Zhao, Angeliki Metallinou, Tagyoung Chung, and 1 others. 2020. Dialog simulation with realistic variations for training goal-oriented conversational systems. *arXiv preprint arXiv:2011.08243*.

McDonald's Corporation. 2024. Our menu. Accessed: 2024-06-22.

MiniMax. 2026. MiniMax M2.5: Built for real-world productivity. https://www.minimax.io/news/minimax-m25.

OpenAI. 2024. Gpt-4o system card.

OpenAI. 2025a. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/.

OpenAI. 2025b. OpenAI GPT-5 system card. *Preprint*, arXiv:2601.03267.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. 2026. HAL reliability benchmark: GAIA analysis. https://hal.cs.princeton.edu/reliability/benchmark/gaia/analysis/. Accessed: 2026-05-21.

Harvey Sacks, Emmanuel A. Schegloff, and Gail Jefferson. 1978. chapter 1 - a simplest systematics for the organization of turn taking for conversation. In JIM SCHENKEIN, editor, *Studies in the Organization of Conversational Interaction*, pages 7–55. Academic Press. This chapter is a variant version of "A Simplest Systematics for the Organization of Turn-Taking for Conversation," which was printed in Language, 50, 4 (1974), pp. 696–735. An earlier version of this paper was presented at the conference on "Sociology of Language and Theory of Speech Acts," held at the Centre for Interdisciplinary Research of the University of Bielefeld, Germany. We thank Dr. Anita Pomerantz and Mr. Richard Faumann for pointing out to us a number of errors in the text.

Emanuel A. Schegloff, Gail Jefferson, and Harvey Sacks. 1977. The preference for self-correction in the organization of repair in conversation. *Language*, 53(2):361–382.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *Preprint*, arXiv:2501.17399.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. 2024. Workbench: a benchmark dataset for agents in a realistic workplace setting. In *First Conference on Language Modeling*.

Ryan Teknium, Jeffrey Quesnelle, and Chen Guang. 2024. Hermes 3 technical report. *Preprint*, arXiv:2408.11857.

Xin Tian, Liankai Huang, Yingzhan Lin, Siqi Bao, Huang He, Yunyi Yang, Hua Wu, Fan Wang, and Shuqi Sun. 2021. Amendable generation for dialogue state tracking. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 80–92, Online. Association for Computational Linguistics.

Sarina Xi, Vishisht Rao, Justin Payan, and Nihar B. Shah. 2025. Flaws: A benchmark for error identification and localization in scientific papers. *Preprint*, arXiv:2511.21843.

Shisen Yue, Siyuan Song, Xinyuan Cheng, and Hai Hu. 2024. Do large language models understand conversational implicature- a case study with a Chinese sitcom. In *Proceedings of the 23rd Chinese National Conference on Computational Linguistics (Volume 1: Main Conference)*, pages 1270–1285, Taiyuan, China. Chinese Information Processing Society of China.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

A Menu

We sample items from the following menu (Table 6), inspired by the McDonald's menu (McDonald's Corporation, 2024).

Items are organized into four categories: main, side, drink, and dessert. Where applicable, items use a three-point size scale (small, medium, large); chicken nuggets use piece counts (6, 10, 20). Items without a size entry (most mains, Vanilla Cone, apple slices) have no size attribute. Only mains have removable ingredients. The *Alternative names* column lists other terms that may be used for indirect reference in amendment utterances; items without alternatives (Lemonade, water) cannot be targeted via indirect reference.

Item	Sizes	Removable ingredients	Alternative names
Mains			
McChicken	—	lettuce, mayonnaise	chicken sandwich, sandwich, chicken burger
quarter pounder w/ cheese	—	pickles, onions, ketchup, mustard, cheese	quarter pounder, burger, cheeseburger, sandwich
double quarter pounder w/ cheese	—	pickles, onions, ketchup, mustard, cheese	double quarter pounder, quarter pounder, burger, sandwich, cheeseburger
double QP w/ cheese & bacon	—	pickles, onions, ketchup, mustard, cheese, bacon	double quarter pounder, quarter pounder, burger, cheeseburger, bacon burger, bacon cheeseburger
chicken nuggets	6, 10, 20	—	nuggets
Big Mac	—	pickles, onions, lettuce, special sauce	burger, sandwich
Filet-O-Fish	—	lettuce, tartar sauce	fish sandwich, sandwich
cheeseburger	—	pickles, onions, ketchup, mustard, cheese	burger, sandwich
double cheeseburger	—	pickles, onions, ketchup, mustard, cheese	burger, sandwich, cheeseburger
Sides			
fries	S/M/L	—	french fries
salad	S/M/L	—	green salad
apple slices	—	—	apple
Drinks			
Coca-Cola	S/M/L	—	Coke, soda
Dr. Pepper	S/M/L	—	soda
Lemonade	S/M/L	—	—
Sweet Tea	S/M/L	—	tea
water	S/M/L	—	—
Sprite	S/M/L	—	soda
Desserts			
McFlurry w/ OREO Cookies	—	—	McFlurry, ice cream, oreo ice cream
McFlurry w/ M&M's Candies	—	—	McFlurry, ice cream, m&m's ice cream
Vanilla Cone	—	—	cone, ice cream, vanilla ice cream
Vanilla Shake	S/M/L	—	shake, vanilla ice cream shake
Chocolate Shake	S/M/L	—	shake, chocolate ice cream shake
Strawberry Shake	S/M/L	—	shake, strawberry ice cream shake

Table 6: Full menu. Items are organized by category. Columns show available sizes (S/M/L = small, medium, large), removable ingredients, and alternative names used for indirect reference in amendments.

B Data Generation

B.1 Order Generation

Orders are generated by sampling one item per required category (main, side, drink), with a 50% chance of including a dessert. Items and sizes are sampled uniformly within each category, except for sides, where fries are sampled at 80%. To render a natural-language utterance, we combine a randomly selected intent phrase (e.g., "I'll have", "Can I get", drawn from 17 forms) with item-level determiner templates that introduce surface variation (e.g., "a medium Coca-Cola", "the large fries", "a 10-piece chicken nuggets"). Items are listed in a fixed category order: main, side, drink, and optionally dessert. All variation is purely surface-level and does not affect the inferences required to interpret any subsequent amendment.

JSON output format. Each item in the order is represented as a JSON object with four fields: `name`, `category`, `size`, and `modifications`. The full order is a list of such objects. For example:

```
[{"name": "cheeseburger",
  "category": "main",
  "size": null, "modifications": []},
 {"name": "fries",
  "category": "side",
  "size": "medium", "modifications": []},
 {"name": "Coca-Cola",
  "category": "drink",
  "size": "medium", "modifications": []}]
```

Items without applicable sizes (e.g., burgers) use `null`. Modifications are represented as a list of ingredients to be removed (e.g., `["no pickles", "no onions"]`).

B.2 Amendment Generation

Each example involves exactly one amendment to the order. Amendments are generated from 624 unique structured templates across seven operation subtypes (Table 7).

B.2.1 Template Structure

Each template is a string with placeholders that are filled at generation time with values sampled from the order and menu. Every template carries metadata specifying which components are explicit, implicit, or N/A. Five of the seven components are fixed at authoring time; the remaining two (`target_field_value` and `new_item_size`) depend on the sampled items and are resolved at generation time. For example, `new_item_size` is N/A when substituting for a Cheeseburger (which has no size), but explicit or implicit when substituting for a Coca-Cola, depending on whether the utterance mentions the size (e.g., large). Some templates are *flexible*: the generator produces variants with and without explicit size mentions from the same template string, e.g., "Change my large Coca-Cola to a large Sprite" (size explicit) vs. "Change my Coca-Cola to a Sprite" (size implicit, must be inherited).

Amendment Type	Subtype	Templates
Deletion	—	60
Addition	new item	29
	existing, same size	71
	existing, diff. size	89
Modification	ingredients	105
	size	145
Substitution	—	125
Total		**624**

Table 7: Amendment templates by amendment type (operation) and subtype.

B.2.2 Generation Constraints

Several constraints ensure that every generated example is valid and unambiguous:

Operation validity. The menu, order parameters, and amendment type (and subtype) constrain which values can be sampled to parameterize amendment templates. Deletion requires the item to exist in the order. Substitution samples a *different* item from the *same* category (e.g., one drink for another). Modification–size requires the item to have available sizes; modification–ingredients requires it to have removable ingredients (both size and ingredients are sampled from the menu). Addition–new item excludes items already in the order. Addition–existing item samples from the order; same-size addition requires the sampled item to have a size, while different-size addition additionally excludes the item's current size from the available size values. Each example has metadata, including the component values, which are always updated to reflect the sampled values. For example, when addition samples an item without a size, `new_item_size` is N/A.

Indirect references. Templates with `target_item=implicit__indirect` use alternative item names from the menu—lexical paraphrases, synecdoche, or metonymy (e.g., "Coke," "soda," "drink" for Coca-Cola; "chicken

sandwich," "sandwich," "main" for McChicken; see Table 6). Because each order contains at most one item per category, a category-level reference like "the main" or a shared synonym like "soda" always resolves to exactly one item, making every indirect reference *unambiguous* in context. Some templates additionally exclude alternative names that collide with primary names of other items in the same category, potentially causing ambiguity. For example, given an order with a quarter-pounder with cheese, "remove the cheeseburger from my order" is unambiguous, whereas "I want that cheeseburger without pickles" or "Add one more of that cheeseburger" could be confused with adding a cheeseburger from the menu.

Deictic and zero-anaphora references. Templates with `target_item=implicit__that` use "that" or "those" to refer to a contextually salient item (e.g., "Make that a Big Mac instead," given an order of cheeseburger, medium fries, medium Coca-Cola). In substitution templates, negated forms like "not that" are a special case, as the expression could in principle reject the entire order; we only generate cases where recency and category matching disambiguate: "Not that. I want a small Sprite instead" is unambiguous because Coca-Cola is the most recent item and Sprite is also a drink. By contrast, "Not that. I'll have a Big Mac instead" would be ambiguous—"that" could refer to any item or the whole order—so we do not generate such examples. Templates with `target_item=implicit__not_mentioned` omit the target entirely (zero anaphora) (e.g., "I want a Sprite instead"), rely on similar constraints.

Directional size inference. All sized items use a three-point scale (e.g., small/medium/large; see Table 6). The model must apply a function: given the current size and a direction (+1 or −1 on the scale), compute the target size. Some templates specify a step ("one size larger"), making the target unambiguous from any starting size; the generator also sets the starting size to the middle of the scale so the step remains valid. For bare directional terms ("Make my Sprite larger"), the generator likewise sets the starting size to the middle, so the direction resolves to exactly one target—"larger" from medium can only mean large.

Size inheritance in substitution. When a substitution template leaves the new item's size implicit (`new_item_size=implicit`), the new item inherits the replaced item's size. This requires both items to have a size attribute. The generator forces explicit size for dissimilar pairs within the same category—e.g., Coca-Cola ↔ water (drinks) or fries ↔ salad (sides)—where size inheritance would be unreasonable, while implicit inheritance is allowed for similar pairs (e.g., Coca-Cola ↔ Sprite, both soft drinks with the same size scale): "Change my large Coca-Cola to a **large** water" (size explicit, forced) vs. "Change my large Coca-Cola to a Sprite" (size implicit, inherited).

Ingredient modifications. Because only mains have removable ingredients (Table 6), ingredient modification always unambiguously targets the main item. The generator samples 1–3 ingredients uniformly at random from the item's removable set. We excluded or regenerated cases where removing an ingredient changes the nature of the item and potentially maps it to a different menu item—e.g., removing cheese from a Cheeseburger or bacon from a Double Quarter Pounder with Cheese and Bacon (see Appendix C.3).

Sampling and deduplication. For the broad dataset, each template is sampled a fixed number of times, with templates for underrepresented component values (e.g., zero anaphora for `target_item`, directional up/down for `target_field_value`) boosted to ensure coverage (Table 10). For the targeted dataset, templates matching each profile are shuffled and cycled in round-robin order until the target count is reached. For both datasets, each generated order–utterance pair is checked against existing ones and regenerated if not unique.

Category selection is uniform across valid categories for most operations; the exception is addition–new, which uses weighted sampling (main 35%, side 10%, drink 30%, dessert 25%) to reflect the smaller number of unique sides. For substitution, the new item receives the same size as the replaced item 70% of the time (different size 30%).

For the broad dataset, whether to mention the added or substituted item's size in the utterance is determined with 50% probability; whether to mention the anchor item's size is likewise 50%. Since the anchor's size is always stated in the order utterance, we do not treat its presence or absence in the amendment utterance as a different inference type. Component values (e.g., `new_item_size`) are set to implicit or explicit accordingly. For the targeted dataset, size mention is imposed rather

than sampled, ensuring that inheritance inferences are required where intended.

At most 20% of examples per profile may have an explicitly stated amendment offer, preventing over-representation of less interesting surface forms (e.g., “I’m changing my order”).

B.2.3 Example Templates and Realizations

Table 8 illustrates the generation pipeline: each row shows a concrete utterance (with filled placeholder values in **bold**), the component values the model must infer, and the inference type required. Examples range from fully explicit to implicit across all four operation types.

Amendment Type	Utterance	Implicit components (*correct values*)	Inferences tested
Assumed order: cheeseburger, medium fries, medium Coca-Cola, medium Chocolate Shake.			
Deletion	"I'm changing my order. Please remove the **medium fries**."	— (fully explicit)	—
Deletion	"No wait. No **burger**."	amendment type, target item (*cheeseburger*)	implicit operation type, indirect ref.
Modification	"Change my **medium fries** to **large**."	target field (*size*)	implicit field identification
Modification	"Remove the **pickles** from my **cheeseburger**."	target field (*ingredients*)	implicit field identification
Modification	"Make my **fries** larger."	amendment type, target field (*size*), target field value (*large*)	implicit operation type, implicit field identification, directional size inference
Modification	"Hold the **onions and pickles**."	amendment type, target item (*cheeseburger*), target field (*ingredients*)	implicit operation type, zero anaphora, implicit field identification
Substitution	"Make that a **Vanilla Shake**."	amendment type, target item (*Choc. Shake*), new item size (*medium*)	implicit operation type, deictic ref., anchor property inheritance
Substitution	"No wait. A **Sprite** instead."	amendment type, target item (*Coca-Cola*), new item size (*medium*)	implicit operation type, zero anaphora, anchor property inheritance
Addition	"Please add a **6-piece chicken-nuggets**."	— (fully explicit)	—
Addition	"I want one more **Coca-Cola**. Make that **large**."	amendment type	implicit operation type
Addition	"Give me one more **Coca-Cola**."	new item size (*medium*)	anchor property inheritance
Addition	"I want another of those **fries**."	amendment type, new item size (*medium*)	implicit operation type, anchor property inheritance

Table 8: Amendment examples from the dataset. **Bold** marks values filled from the order and menu at generation time; non-bold text is fixed by the template. The Implicit components column lists which components are implicit; italicized parenthetical values are the correct values the model must infer. Amendment offer is implicit in all rows except the first and is omitted for space.

C Datasets

C.1 Dataset Distribution

Tables 9 and 10 compare the operation-subtype and component-value distributions across all five datasets. Table 11 adds per-value flagship accuracy and the gap relative to explicit baselines. All counts reflect totals excluding instances that have been struck from the dataset for ambiguity or disfluency. The targeted dataset is constructed from difficulty profiles that bias sampling toward harder inferences; the shifts in distribution—e.g., substitution rising from 10.8% to 25.0%, deictic references from 6.2% to 12.5%—reflect this deliberate oversampling.

Implicit-component composition. Table 12 decomposes examples by *which* components are implicit, for n=1 through n=4, on both the broad dataset (711 examples) and the targeted dataset (720 examples).

At every value of n and in both datasets, groups involving harder inferences have lower accuracy, while groups without them approach 100%. The apparent "recovery" at n=4 seen in Figure 1 occurs because n=4 has proportionally fewer hard examples. In short, accuracy is heavily impacted by which components are implicit and the difficulty of the inferences required to resolve them. The number of implicit components is also a factor, but predominantly as a proxy for the distribution of inferences tested at each n. Table 12 provides quantitative evidence for this claim (Section 4).

Shuffled dataset (278 examples). A shuffled variant is generated from the same targeted-dataset profiles at one-third the sampling rate but with randomized item mention order in the order utterance (e.g., "I want large fries, a cheeseburger, . . . " rather than the standard main–side–drink order), testing whether models rely on positional cues. Results are reported in Appendix D.

	Broad (711)	Targeted (720)	Shuffled (289)	Clarification (324)	Underspecified (50)
Addition					
exist. same size	85	80	30	37	30
exist. diff. size	45	130	56	35	20
new	28	10	4	4	—
Deletion	56	30	12	16	—
Modification					
size	190	170	64	65	—
ingredients	130	70	28	30	—
Substitution	77	180	45	107	—
Vanilla (no amendment)	100	50	50	30	—
Total	**711**	**720**	**289**	**324**	**50**

Table 9: Amendment type (operation) and subtype distribution across all five datasets (post-exclusion counts). Broad = baseline evaluation (23 models); Targeted = denser sampling of difficult profiles; Shuffled = randomised item order (same profiles, ≈40% sample); Clarification = subset where models may ask for implicit information; Underspecified = genuinely ambiguous deictic examples.

Component	Value	Broad (711)	Targeted (720)	Shuffled (289)	Clarification (324)	Underspecified (50)
`amendment_offer`	explicit	143	74	21	35	5
	implicit	468	596	218	259	45
	N/A	100	50	50	30	—
`amendment_type`	explicit	269	241	90	118	—
	implicit	342	429	149	176	50
	N/A	100	50	50	30	—
`target_item`	explicit	296	190	73	101	—
	indirect ref	176	300	104	111	—
	deictic	44	90	33	46	50
	zero anaphora	67	90	29	36	—
	N/A	128	50	50	30	—
`target_field`	explicit	124	101	36	45	—
	implicit	196	139	56	50	—
	N/A	391	480	197	229	50
`target_field_value`	explicit	197	140	56	51	—
	dir. up	63	50	18	21	—
	dir. down	60	50	18	23	—
	N/A	391	480	197	229	50
`new_item`	explicit	77	180	45	107	—
	N/A	634	540	244	217	50
`new_item_size`	explicit	90	140	64	75	20
	inheritance	90	160	35	95	30
	dir. up	—	50	18	6	—
	dir. down	—	50	18	7	—
	N/A	531	320	154	141	—

Table 10: Component value distributions across all five datasets (post-exclusion counts). N/A = component does not apply to this operation type (e.g., `new_item_size` is N/A for deletions). For `new_item_size`, "inheritance" = anchor property inheritance; "dir. up/down" = directional size inference (up = larger, down = smaller). See Table 11 for per-value flagship accuracy.

		Broad (711)			Targeted (720)		
Component	**Inference**	n	Flg (%)	Δ	n	Flg (%)	Δ
`amendment_offer`	—	143	96.7	—	74	96.0	—
	implicit	468	97.6	+0.9	596	92.6	-3.4
`amendment_type`	—	269	98.0	—	241	95.5	—
	implicit	342	96.9	-1.1	429	91.5	-4.0
`target_item`	—	296	96.8	—	190	92.1	—
	indirect ref	176	99.0	+2.3	300	94.4	+2.3
	deictic	44	92.5	-4.3	90	89.0	-3.1
	zero anaphora	67	98.0	+1.2	90	94.2	+2.1
`target_field`	—	124	99.0	—	101	98.7	—
	implicit	196	99.7	+0.7	139	98.0	-0.7
`target_field_value`	—	197	99.7	—	140	98.0	—
	dir. up	63	99.3	-0.4	50	98.2	+0.2
	dir. down	60	98.6	-1.1	50	99.1	+1.2
`new_item`	—	77	92.2	—	180	84.3	—
`new_item_size`	—	106	96.9	—	140	91.2	—
	inheritance	55	83.7	-13.1	160	82.7	-8.4
	dir. up	—	—	—	50	98.0	+6.8
	dir. down	—	—	—	50	98.5	+7.4

Table 11: Per-component flagship accuracy (%) and gap vs. explicit baseline (Δ, in pp). — = component is explicit (no inference required); named rows = inference type when implicit. N/A values omitted. Components with small $|\Delta|$ across all inference types do not drive difficulty; those with at least one large negative Δ motivate the targeted profiles. "dir. up/down" = directional size inference (up = larger, down = smaller); "inheritance" = anchor property inheritance. Flg = mean accuracy across 7 flagship models (k=5 runs each).

Implicit components	Distinguishing inferences	Broad		Targeted	
		n	Acc.	n	Acc.
n=1 *implicit components (broad: 100, targeted: 41)*					
new_item_size	inheritance	6	70.0	5	87.4
type	(none)	22	98.6	5	100.0
target_item	indirect ref, deictic	15	99.6	11	97.4
offer	(none)	51	99.8	19	99.7
target_field	implicit	6	100.0	1	100.0
n=1 **avg.**		100	97.7	41	97.6
n=2 *implicit components (broad: 188, targeted: 140)*					
offer, new_item_size	inheritance	7	77.1	21	84.2
type, new_item_size	inheritance	8	86.1	1	88.6
type, target_field	implicit	4	94.3	2	91.4
type, target_item	indirect ref, zero anaphora	11	95.1	7	89.4
offer, type	(none)	64	97.8	23	98.3
offer, target_field_value	dir. down, dir. up	36	99.0	24	98.8
offer, target_item	indirect ref, deictic	22	99.9	36	94.8
target_item, new_item_size	indirect ref, inheritance	6	100.0	12	98.3
offer, target_field	implicit	24	100.0	9	100.0
target_item, target_field	implicit, deictic, indirect ref	6	100.0	5	98.9
n=2 **avg.**		188	97.2	140	94.9
n=3 *implicit components (broad: 189, targeted: 257)*					
offer, type, new_item_size	inheritance	7	69.8	53	81.1
offer, target_item, new_item_size	inheritance, indirect ref, deictic	7	87.8	39	94.2
offer, type, target_item	indirect ref, deictic, zero anaphora	25	90.7	66	84.6
type, target_item, target_field	implicit, zero anaphora, deictic, indirect ref	27	98.8	18	97.0
offer, type, target_field_value	dir. up, dir. down	26	99.2	16	100.0
offer, target_item, target_field_value	indirect ref, dir. up, dir. down	30	99.5	30	99.5
type, target_item, new_item_size	inheritance, indirect ref, zero anaphora	5	100.0	3	96.2
offer, type, target_field	implicit	36	100.0	7	100.0
offer, target_item, target_field	implicit, indirect ref, zero anaphora, deictic	26	100.0	25	93.5
n=3 **avg.**		189	96.9	257	90.3
n=4 *implicit components (broad: 107, targeted: 228)*					
offer, type, target_item, new_item_size	inheritance, indirect ref, deictic	9	83.8	126	89.9
offer, type, target_item, target_field_value	indirect ref, dir. up, dir. down	31	98.1	30	97.0
offer, type, target_item, target_field	implicit, zero anaphora, indirect ref, deictic	67	99.9	72	99.4
n=4 **avg.**		107	98.0	228	93.8

Table 12: Full composition of implicit-component groups for n=1 through n=4, comparing broad (711 examples) and targeted (720 examples) datasets. Acc. = average accuracy of 7 flagship models over k=5 runs.

C.2 Targeted Dataset Profiles

Each template and how it is parameterized leaves specific components implicit—these are *what* the model must infer. Different components admit different *types* of inference for resolving them (Section 3.4). Not all components are equally difficult when implicit. We use the broad-dataset accuracy gap between explicit and implicit values (Table 11) to identify which components warrant detailed analysis, and then test all inference types for those components systematically.

Which components drive difficulty? Three components have at least one inference type that is empirically hard: `target_item` (deictic reference: -4.3 pp), `new_item_size` (anchor property inheritance: -13.1 pp), and `target_field_value` (directional size inference: -0.4 to -1.1 pp). For each, multiple inference types are possible—some hard, some easy. We ensure denser coverage of all of them to confirm which ones drive difficulty and whether difficulty compounds when we combine these with other implicit components.

Why we do not drill down on the other components. The remaining four components could of course be drilled down by inference type as well. However, we felt that their component-level gap was too small to warrant this effort. `amendment_offer` (implicit: $+0.9$ pp) and `amendment_type` (implicit: -1.1 pp) are easy regardless of how they are expressed—e.g., "I want another" (implicit addition) vs. "Please add" (explicit) are both trivially interpretable when downstream components are stated. `target_field` is always implicit for modifications to ingredients (no useful explicit/implicit contrast), and fairly obvious when it is implicit for modifications to size (e.g., "Make my Sprite larger/large." vs. "Change the size of my Sprite to large") and `new_item` is always explicit (it must be stated for the amendment to be interpretable).

Profile construction. A *profile* pairs an amendment subtype with one or two specific inference types for these three components—e.g., "substitution requiring deictic reference resolution and anchor property inheritance." We enumerate profiles by crossing each amendment subtype with the inference types admitted by its relevant components, yielding 33 active profiles (two deictic addition profiles were fully excluded; Section C.3).

Band assignment. Profiles are grouped into three bands by the number of these components left implicit: *Hard* (2 of these components implicit, 30 examples each), *Moderate* (1 implicit, 20 each), and *Control* (all three explicit—baseline, 10 each), plus 50 vanilla orders (no amendment). Bands reflect the number of inferences required, not predicted difficulty. Some single inferences are intrinsically harder than certain two-inference combinations—e.g., M19 deictic reference in substitution (85.4% average flagship accuracy in the targeted dataset, n=20) is harder than H5 indirect reference + directional size up in modification (97.0%, n=30). The Hard band tests whether combining multiple implicit components *compounds* difficulty. The broad dataset covers many of these combinations too sparsely for reliable estimates (e.g., H9 deictic + inheritance: n=4; H8 zero anaphora + inheritance: n=1). Denser sampling in the targeted dataset provides the statistical power to confirm these patterns.

Table 13 lists all profiles with their coverage and flagship accuracy on both datasets.

ID	Amendment Type	Subtype	Inferences	Broad n	Broad Acc	Targeted n	Targeted Acc	Tpl
Hard — *2 of these components implicit, 30 examples each*								
H1	Addition	exist. diff. size	Indirect ref. + Dir. size up	—	—	30	97.5	5
H2	Addition	exist. diff. size	Indirect ref. + Dir. size down	—	—	30	98.0	5
H3	Addition	exist. same size	Indirect ref. + Anchor prop. inheritance	19	99.7	30	98.9	15
H5	Modification	size	Indirect ref. + Dir. size up	31	99.6	30	97.0	8
H6	Modification	size	Indirect ref. + Dir. size down	30	97.9	30	99.5	6
H7	Substitution	—	Indirect ref. + Anchor prop. inheritance	3	86.7	30	83.9	18
H8	Substitution	—	Zero anaphora + Anchor prop. inheritance	1	100.0	30	93.1	11
H9	Substitution	—	Deictic ref. + Anchor prop. inheritance	4	53.6	30	77.5	26
Moderate — *1 of these components implicit, 20 examples each*								
M1	Addition	exist. diff. size	Dir. size up	—	—	20	98.7	5
M2	Addition	exist. diff. size	Dir. size down	—	—	20	99.3	5
M3	Addition	exist. diff. size	Indirect ref.	16	97.1	20	72.7	20
M5	Addition	exist. same size	Anchor prop. inheritance	18	70.0	20	74.1	20
M6	Addition	exist. same size	Indirect ref.	—	—	20	99.1	13
M7	Deletion	—	Indirect ref.	21	100.0	20	93.7	20
M8	Modification	ingredients	Indirect ref.	18	99.8	20	93.3	20
M9	Modification	ingredients	Zero anaphora	51	99.8	20	99.4	18
M10	Modification	ingredients	Deictic ref.	24	100.0	20	99.4	20
M11	Modification	size	Dir. size up	32	98.9	20	100.0	8
M12	Modification	size	Dir. size down	30	99.3	20	98.6	6
M13	Modification	size	Indirect ref.	14	100.0	20	95.3	15
M14	Modification	size	Zero anaphora	12	97.9	20	100.0	7
M15	Modification	size	Deictic ref.	7	100.0	20	99.3	14
M16	Substitution	—	Anchor prop. inheritance	10	87.7	20	57.6	19
M17	Substitution	—	Indirect ref.	8	100.0	20	99.1	18
M18	Substitution	—	Zero anaphora	1	60.0	20	84.6	11
M19	Substitution	—	Deictic ref.	3	75.2	20	85.4	20
Control — *all three components explicit (baseline), 10 examples each*								
C1	Addition	exist. diff. size	all explicit	29	94.1	10	96.3	10
C2	Addition	exist. same size	all explicit	20	100.0	10	100.0	10
C3	Addition	new	all explicit	—	—	10	98.3	10
C4	Deletion	—	all explicit	35	99.9	10	100.0	10
C5	Modification	ingredients	all explicit	37	99.4	10	98.3	10
C6	Modification	size	all explicit	34	100.0	10	100.0	10
C7	Substitution	—	all explicit	18	100.0	10	99.7	10
Vanilla — *plain orders, no amendment*								
vanilla	Vanilla	—	—	100	99.9	50	99.9	—
		Total				**720**		

Table 13: Targeted dataset profiles with broad-dataset comparison. The three components selected for detailed analysis are target item, new item size, and target field value. Each profile pairs an amendment subtype with one or two inferences for these components. n = post-exclusion example count; Acc = mean accuracy (%) across 7 flagship models (k=5 runs each); Tpl = distinct templates (each filled with different menu values for diversity). "Dir. size up/down" = directional size inference (up = larger, down = smaller); "Anchor prop. inheritance" = size inherited from an anchor item in the order. "—" in Broad columns = combination absent from broad dataset (motivating targeted sampling). Two deictic addition profiles (H4, M4; 50 examples) were fully excluded as genuinely ambiguous (Section C.3) and are omitted.

C.3 Data Quality and Exclusions

After generation, we apply post-hoc quality checks to identify examples whose ground truth cannot be reliably inferred from the utterance alone, or where resolution requires unreasonable effort. Such examples are removed via per-dataset exclusion lists.

C.3.1 Broad Dataset (151 excluded, 711 effective)

Of 862 generated examples, 151 are excluded: **puzzle-like directional size resolution** (88)—deictic modify-size templates where all items but one are set to boundary sizes, making only one item able to move in the requested direction (e.g., "Make that larger" when the order contains a cheeseburger (no size), large fries, and a medium Coke—only the Coke can be made larger). The reference is technically unambiguous, but resolving it requires checking every item's size—more puzzle-solving than natural inference; **ambiguous references** (22)—indirect or deictic expressions whose referent cannot be uniquely resolved. In addition, as described in Appendix B.2, negated deictic forms like "not that" in substitutions could in principle reject the entire order. We keep cases where recency and category matching disambiguate (e.g., "Not that. I want a small Sprite instead"—Coca-Cola is the most recent item and Sprite is also a drink; or "I don't want that drink. I want a small Sprite instead," where "drink" provides a category cue); examples where they do not (e.g., "Not that. I'll have a Big Mac instead," where "that" could refer to any item or the whole order) are excluded; **dissimilar size inheritance** (13)—substitutions across dissimilar subcategories (e.g., fries ↔ salad, water ↔ flavoured drinks) where size inheritance is too much to expect; and 28 miscellaneous issues (disfluent templates, reversed amendment order, item-specific edge cases). The most prominent of these item-specific edge cases involved ingredient modification examples where removing an ingredient changes the nature of the item—e.g., removing cheese from a Cheeseburger or bacon from a Double Quarter Pounder with Cheese and Bacon (with Double Quarter Pounder with Cheese also on the menu). Such removals potentially map the item to a different menu entry, overcomplicating the task beyond the intended scope of ingredient modification. These were excluded from the broad dataset; for the targeted dataset, we flagged such examples, then the generator was constrained to avoid such combinations, and affected examples were regenerated.

C.3.2 Targeted Dataset (50 excluded, 720 effective)

Templates whose generated examples were flagged frequently in the manual review of the broad dataset were excluded. Manual review also found 23 unusual or disfluent examples caused by the values sampled from the menu (e.g., "I don't want any cheese on my cheeseburger", "Add chicken nuggets. 10 please."—where "10" could be read as "10 orders of nuggets" rather than 10-piece). Instead of excluding them, we regenerated these examples—and checked them again—to preserve the target distribution. 50 deictic addition examples (30 same-size, 20 different-size) are excluded because recency—resolving "that" to the most recently mentioned item—is the *only* disambiguation signal (e.g., "Add one more of that"). The ground truth is simply "duplicate the last-mentioned item"—no new item category, size constraint, or semantic cue anchors the resolution. These examples are used as the underspecified subset in the clarification experiment (§5.2), where accuracy of 5.7–24.7% under the Standard prompt confirms they are genuinely ambiguous.

C.3.3 Shuffled Dataset (42 excluded, 278 effective)

Of 320 generated examples, 42 are excluded: 18 recency-only deictic additions (same criterion as above); 15 substitutions where an unsized item is replaced by a sized item without stating the size—since the anchor has no size, there is nothing to inherit (e.g., replacing a cheeseburger with chicken nuggets, which require a size, but the cheeseburger has none to inherit from); 2 overlap both categories; and 11 substitution were ambiguous.

D Experiment Leaderboards

Table 14 shows the full 23-model baseline leaderboard on the broad dataset (711 examples, k=5 runs), including single-run temperature-0 accuracy where available.

Table 15 presents per-model performance across all four prompting methods (Standard, Two-step, Sim. FC, Native FC) on the targeted dataset (k=5, n=720). All methods use 15 models. The DeepInfra API used to access Hermes 3 and DeepSeek v3.2 had limited native tool-use support: no support for Hermes 3 and unreliable outputs for DeepSeek v3.2.

Native vs. Simulated FC. The per-model data (Table 15) shows that the gap between Native FC and Sim. FC is inconsistent across providers. Among flagships: Sonnet is stable (97.7% → 98.0%), Opus and GPT-5 show minor drops (∼1.5 pp), while GPT-4o drops 5.9 pp and Gemini Pro drops 4.0 pp—the latter possibly related to schema adaptations (Google required making `size` and `modifications` fields optional; Appendix I.1). Among smaller models the picture is similarly mixed: GPT-5-mini actually improves slightly (97.2% → 97.9%), Gemini Flash and GPT-5-nano are stable, but GPT-4o-mini drops 22.6 pp. The models queried via the DeepInfra API (Llama 70B/8B, DeepSeek, Hermes) show massive drops or total failure—DeepInfra only supports `tool_choice: "auto"` and forced tool calls threw errors (Appendix I.1). The pattern appears to be provider- and generation-specific rather than scale-related: the GPT-5 family handles Native FC well across sizes, while the GPT-4o family struggles. Native FC varies across providers in ways we cannot control, which is why we designed Sim. FC as the primary experimental condition—it uses identical function definitions and prompts across all models.

Temperature 0. Although this is not a creative task, we use default temperature throughout. Temperature-0 results (where available) confirm this choice: most models perform within 1–2 pp of their k=5 average (Table 14). GPT-5 models do not support temperature 0.

Positional bias. The Shuffled variant (278 examples, 9 models including 7 flagships; Appendix C.1) randomizes item mention order in the order utterance, testing whether models rely on positional cues (Table 15). The Standard prompt averages 92.3% on the shuffled dataset vs. 93.4% on the targeted dataset—a minor difference partially attributable to different example compositions (the shuffled set samples one-third as densely from the same profiles). Six of nine models decline slightly; the effect does not change which inferences are hard.

Model scale. Smaller models exhibit the same difficulty patterns—the same hard inference–amendment type pairs dominate errors—but at lower overall accuracy (Section 4). Both Sim. FC and Two-step benefit smaller models disproportionately, likely because the structured decomposition compensates for weaker single-turn reasoning.

Table 16 breaks down accuracy by inference type for each of the 15 models evaluated on the targeted dataset (720 examples, k=5 runs; 50 recency-only exclusions applied), grouped into flagship (7 models, averaged), mid-tier (6 models shown individually: Claude Haiku 4.5, Gemini 2.5 Flash, Gemini 2.5 Flash Lite, GPT-4o-mini, GPT-5-mini, Llama 3.1 70B), and small (GPT-5-nano, Llama 3.1 8B). The ranking of inference difficulty is remarkably stable across scales. Anchor property inheritance is the hardest inference type at every tier: 82.7% flagship → 73–87% mid-tier → 51–52% small. Deictic reference is consistently the second hardest. No inference type that is easy for flagships becomes hard for smaller models—the difficulty ordering is preserved, only the absolute accuracy level shifts downward. This confirms that the pragmatic inference taxonomy captures genuine difficulty drivers, not artifacts of a particular model family or scale.

Table 14: Full leaderboard: Standard method on the broad dataset (711 examples, k=5 runs) with single-run temperature-0 comparison. All metrics averaged over k=5 runs. ⋆ = flagship model (top model per provider). Δ = Temp-0 accuracy − Acc (negative means temp-0 run was less accurate than average default temp over 5 runs). Models sorted by accuracy. — = temperature-0 not supported (GPT-5 family).

			Default temp (k=5)			**Temp-0**	
Rank	**Model**		**Acc**	**pass@k**	**pass^k**	**Acc**	**Δ**
1	hermes-3-llama-3.1-405b	⋆	98.2	99.2	96.3	98.2	+0.0
2	gpt-5-mini		98.2	99.2	96.5	—	—
3	claude-sonnet-4-5	⋆	98.1	98.2	97.9	97.8	−0.3
4	gpt-5	⋆	98.1	98.6	97.2	—	—
5	llama-3.3-70b-instruct-turbo		98.0	98.6	96.6	97.0	−1.0
6	minimax-m2.5		98.0	99.4	95.4	97.7	−0.3
7	deepseek-v3.1-terminus		98.0	98.9	96.5	96.8	−1.1
8	gpt-4o	⋆	97.9	98.7	96.3	97.5	−0.3
9	deepseek-v3.2	⋆	97.7	99.7	92.5	98.6	+0.9
10	llama-3.1-70b-instruct		97.3	99.0	94.1	97.9	+0.7
11	gemini-2.5-pro	⋆	97.2	98.7	95.6	95.6	−1.6
12	minimax-m2.1		97.2	99.3	93.8	96.7	−0.5
13	claude-opus-4-6	⋆	97.1	97.2	97.0	96.5	−0.6
14	gpt-4.1		97.1	97.8	96.1	96.5	−0.5
15	claude-opus-4-5		97.0	97.2	96.8	96.1	−0.9
16	claude-haiku-4-5		97.0	97.3	96.6	96.3	−0.7
17	llama-3.1-70b-instruct-turbo		96.8	98.9	92.8	97.7	+0.8
18	gemini-2.5-flash		96.5	98.0	95.4	96.1	−0.4
19	gpt-4.1-mini		96.3	97.8	94.9	95.4	−0.9
20	gpt-5-nano		96.1	97.6	92.4	—	—
21	gpt-4o-mini		95.1	96.6	92.5	94.7	−0.4
22	gpt-4.1-nano		89.7	93.7	83.3	90.9	+1.1
23	gemini-2.5-flash-lite		89.6	92.0	87.2	89.1	−0.6
	Mean (all 23)		96.6	97.9	94.5		
	Mean (flagship 7)		97.7	98.6	96.1		

Table 15: Per-model performance across methods on the targeted dataset and Standard on the shuffled dataset (k=5 runs). All metrics averaged over k=5 runs; p@k = pass@k, p^k = pass^k. ⋆ = flagship model. Targeted dataset: n=720; Shuffled: n=278 (randomized item mention order, flagship-tier only). †Hermes 3 DeepInfra API had no tool-use support. ‡DeepSeek v3.2 DeepInfra API had unreliable tool-use.

		Standard			**Two-step**			**Sim. FC**			**Native FC**			**Std. Shuffled**		
Model		Acc	p@k	p^k	Acc	p@k	p^k	Acc	p@k	p^k	Acc	p@k	p^k	Acc	p@k	p^k
claude-haiku-4-5		93.1	94.0	91.9	96.3	96.5	96.0	97.5	98.3	96.5	95.7	97.5	93.8	90.3	91.4	89.6
claude-opus-4-6	⋆	92.7	93.2	92.1	95.2	95.4	94.7	97.3	97.5	97.0	95.8	96.1	95.6	91.2	92.1	90.3
claude-sonnet-4-5	⋆	93.0	93.9	92.1	95.7	96.4	95.0	97.7	98.1	96.9	98.0	98.6	97.2	91.4	92.5	90.6
deepseek-v3.2‡	⋆	93.4	98.5	84.2	95.2	97.6	90.7	94.2	98.9	84.2	54.1	92.2	9.2	91.9	96.4	83.5
gemini-2.5-flash		91.7	95.7	85.1	90.7	95.4	84.9	94.4	97.1	91.2	94.7	98.2	90.4	92.0	94.2	88.8
gemini-2.5-flash-lite		87.4	91.4	83.1	80.6	86.8	75.1	81.1	83.6	78.3	81.9	84.2	79.3	—	—	—
gemini-2.5-pro	⋆	92.4	95.7	88.2	91.5	95.0	86.4	97.4	99.0	95.3	93.4	98.8	87.1	92.7	94.2	90.3
gpt-4o	⋆	94.4	96.0	92.5	92.9	94.9	90.8	96.1	97.8	93.6	90.2	95.7	82.4	92.3	93.5	90.3
gpt-4o-mini		89.3	92.5	85.1	89.3	93.2	83.2	84.8	88.9	79.9	62.2	77.9	45.7	—	—	—
gpt-5	⋆	91.9	94.6	87.6	92.0	93.9	88.9	97.2	98.5	95.7	95.9	97.9	93.6	94.4	96.0	92.1
gpt-5-mini		93.8	96.8	89.7	93.1	95.3	90.0	97.2	98.9	95.0	97.9	99.6	96.5	—	—	—
gpt-5-nano		83.0	91.8	74.9	84.4	91.0	75.8	92.7	96.2	87.6	91.8	96.5	81.8	—	—	—
hermes-3-405b†	⋆	96.2	98.2	91.4	92.6	95.6	87.6	96.0	98.5	91.2	0.0	0.0	0.0	94.5	96.4	89.2
llama-3.1-70b-instruct		93.8	97.2	89.9	95.2	97.2	92.8	93.6	97.5	87.6	75.3	94.9	43.3	—	—	—
llama-3.1-8b-instruct		67.7	83.8	51.2	78.2	92.8	60.6	81.6	93.6	65.4	51.1	67.0	30.8	—	—	—
Mean (all)		90.3	94.2	85.3	90.9	94.5	86.2	93.2	96.2	89.0	78.5	86.3	68.4	92.3	94.1	89.4
Flagship (7)		93.4	95.7	89.7	93.6	95.5	90.6	96.6	98.3	93.4	75.3	82.8	66.4	92.6	94.5	89.5
Flagship (5)†‡		92.9	94.7	90.5	93.5	95.1	91.2	97.1	98.2	95.7	94.6	97.4	91.2	92.4	93.7	90.7

			Mid-tier models						Small models	
Inference	n	**Flag.**	**Haiku**	**Flash**	**Fl. Lite**	**4o-mini**	**5-mini**	**Ll. 70B**	**5-nano**	**Ll. 8B**
Inheritance	160	82.7	87.1	73.1	85.1	75.5	80.5	84.9	51.4	52.1
Deictic ref.	90	89.0	92.9	84.4	88.4	87.1	82.0	86.9	56.0	54.2
Zero anaphora	90	94.2	86.4	93.6	95.8	87.1	96.2	93.1	58.9	43.3
Indirect ref.	300	94.4	94.5	94.5	82.8	90.4	95.9	95.0	91.5	68.7
Dir. size	200	98.5	97.2	98.9	79.8	94.1	99.4	99.0	98.3	76.5
Implicit field	139	98.0	97.4	98.1	98.6	96.0	98.6	97.0	97.1	84.7
All explicit	101	99.4	100.0	99.6	98.2	98.4	100.0	99.6	99.6	81.4

Table 16: Accuracy by inference type and model size on the targeted dataset (k=5, n=720). Flag. = avg. of 7 flagship models (Claude Sonnet 4.5, Claude Opus 4.6, Gemini 2.5 Pro, GPT-4o, GPT-5, DeepSeek v3.2, Hermes 3). Column abbreviations—Haiku: Claude Haiku 4.5; Flash: Gemini 2.5 Flash; Fl. Lite: Gemini 2.5 Flash Lite; 4o-mini: GPT-4o-mini; 5-mini: GPT-5-mini; Ll. 70B: Llama 3.1 70B; 5-nano: GPT-5-nano; Ll. 8B: Llama 3.1 8B. The same inference types that are hard for flagships (inheritance, deictic reference) remain the hardest across all model scales and families. Exceptions: Flash Lite struggles with directional size and indirect references; Haiku 4.5 shows a specific weakness on zero anaphora.

E Inference Combination Rankings

Tables 17–18 rank all 66 inference combinations by accuracy. Each example belongs to exactly one row, so the n values sum to 711. By contrast, Table 3 aggregates across all rows sharing a given inference type.

Amendment Type	Inferences	n	Acc All	Acc Flg	pass@k	pass^k
Substitution	Offer + Type + Deictic ref. + Inheritance	2	42.2	47.1	54.3	23.9
Substitution	Offer + Type + Deictic ref.	2	44.4	62.9	56.5	34.8
Substitution	Offer + Deictic ref. + Inheritance	2	49.6	60.0	63.0	32.6
Substitution	Offer + Type + Deictic ref.	2	52.2	64.3	56.5	50.0
Substitution	Offer + Type + Zero anaphora	2	57.4	70.0	63.0	52.2
Addition (same size)	Offer + Inheritance	8	59.8	63.6	75.0	40.2
Addition (same size)	Offer + Type + Inheritance	10	67.8	75.1	80.0	47.8
Substitution	Type + Zero anaphora	1	71.3	60.0	78.3	65.2
Substitution	Offer + Type + Indirect ref. + Inheritance	2	80.0	80.0	93.2	63.7
Substitution	Offer + Type + Inheritance	5	82.4	85.1	86.1	80.0
Addition (diff. size)	Offer + Type	14	87.5	90.2	91.6	79.8
Substitution	Offer + Inheritance	5	90.1	90.3	94.8	81.7
Addition (diff. size)	Type + Indirect ref.	7	91.2	94.3	96.2	82.3
Substitution	Offer + Deictic ref.	1	91.3	100.0	95.7	87.0
Modification	Offer + Type + Indirect ref. + Dir. down	15	93.1	96.2	95.8	90.2
Addition (diff. size)	None (all explicit)	15	94.6	97.7	96.8	91.3
Substitution	Type + Zero anaphora + Inheritance	1	94.8	100.0	95.7	91.3
Substitution	Deictic ref.	4	95.0	100.0	97.8	90.2
Addition (same size)	Offer + Type + Indirect ref. + Inheritance	9	95.3	100.0	97.0	93.9
Addition (diff. size)	Indirect ref.	9	95.9	99.4	98.1	93.2
Modification	Offer + Indirect ref. + Dir. down	15	96.3	99.6	97.4	94.4
Substitution	Offer	6	97.3	99.5	97.8	95.6
Deletion	Offer + Indirect ref.	10	97.7	100.0	97.8	97.4
Addition (same size)	Offer + Type	5	97.7	94.3	99.1	95.7
Deletion	Offer + Type + Indirect ref.	11	97.8	100.0	98.8	96.4
Modification	Offer + Type + Indirect ref. + Dir. up	16	97.9	99.8	99.2	95.9
Modification	Offer + Dir. down	20	98.0	99.3	98.9	95.4
Modification	Offer + Dir. up	16	98.0	98.8	98.6	97.0
Modification	Type + Zero anaphora + Field value	12	98.0	97.9	98.9	96.4
Substitution	Offer + Indirect ref.	3	98.3	100.0	98.5	97.1
Modification	Offer + Indirect ref. + Dir. up	15	98.3	99.4	99.1	95.8
Substitution	Offer + Type + Indirect ref.	5	98.4	100.0	99.1	97.4
Modification	Offer + Type + Dir. down	10	98.5	99.4	99.1	97.4

Table 17: Inference combination rankings (part 1 of 2) on the broad dataset (711 examples, 23 models, k=5 runs), sorted by accuracy. **Inference shorthand**: Offer = amendment offer; Type = amendment type; Inheritance = anchor property inheritance; Dir. up/down = directional size inference (up = larger, down = smaller); Field value = implicit field identification. Horizontal rules separate difficulty tiers. Shading indicates difficulty.

Amendment Type	**Inferences**	n	**Acc All**	**Acc Flg**	**pass@k**	**pass^k**
Addition (same size)	Indirect ref. + Inheritance	10	98.5	99.4	100.0	95.5
Substitution	Offer + Indirect ref.	4	98.7	100.0	98.9	97.8
Modification	Type + Deictic ref. + Field value	2	98.7	100.0	100.0	97.8
Deletion	Offer + Type	19	98.7	99.8	99.3	97.9
Modification	Offer + Type + Dir. up	16	99.0	99.1	99.7	97.0
Modification	Type + Field value	23	99.0	99.0	99.6	98.3
Substitution	Offer + Type	11	99.0	100.0	100.0	97.6
Modification	Offer + Type + Indirect ref. + Field value	13	99.1	99.8	99.7	98.0
Deletion	None (all explicit)	16	99.1	100.0	99.2	98.6
Addition (new)	Offer	5	99.1	100.0	100.0	97.4
Modification	Offer + Type + Indirect ref. + Field value	7	99.3	100.0	99.3	99.3
Addition (new)	Type	7	99.4	100.0	100.0	97.5
Substitution	Offer	10	99.4	100.0	100.0	98.7
Addition (same size)	Type + Indirect ref.	6	99.4	100.0	100.0	98.5
Substitution	Offer + Type	8	99.5	100.0	100.0	97.8
Addition (same size)	Offer + Indirect ref.	6	99.5	99.5	100.0	98.5
Modification	Type + Zero anaphora + Field value	42	99.6	99.8	100.0	98.3
Modification	Offer + Type + Deictic ref. + Field value	18	99.6	100.0	100.0	98.3
Modification	Offer + Type + Field value	17	99.6	100.0	100.0	98.5
Addition (same size)	Offer	11	99.7	99.0	100.0	98.4
Addition (same size)	Type	10	99.7	100.0	100.0	98.7
Addition (new)	Offer + Type	12	99.8	99.8	100.0	99.3
Modification	Field value	17	99.8	100.0	100.0	99.5
Vanilla (no amendment)	None (all explicit)	100	99.9	99.9	100.0	99.5
Modification	Field value	14	99.9	100.0	100.0	99.4
Addition (same size)	None (all explicit)	10	100.0	100.0	100.0	100.0
Addition (new)	None (all explicit)	4	100.0	100.0	100.0	100.0
Modification	Offer + Deictic ref. + Field value	6	100.0	100.0	100.0	100.0
Modification	Offer + Indirect ref. + Field value	5	100.0	100.0	100.0	100.0
Modification	Offer + Zero anaphora + Field value	9	100.0	100.0	100.0	100.0
Modification	Deictic ref. + Field value	5	100.0	100.0	100.0	100.0
Modification	Indirect ref. + Field value	7	100.0	100.0	100.0	100.0
Substitution	Indirect ref. + Inheritance	1	100.0	100.0	100.0	100.0

Table 18: Inference combination rankings (part 2 of 2), continued from Table 17.

F Amendment Type by Inference Breakdown

Table 19 expands the body table (Table 4) to show all amendment type×inference combinations across three methods (Standard, Two-step, Sim. FC) with both flagship accuracy and reliability gap (pass^k − pass@k) for each. Native FC is omitted because native tool-calling was either unsupported or unreliable on the API used to access 2 of the 7 flagship models. The data confirms that substitution is consistently the hardest amendment type across all inference types, and that the reliability gap concentrates on the same hard cases—exceeding 16 points for anchor property inheritance and deictic reference under substitution. Each row includes a representative example utterance.

See Appendix C.3 for the exclusion criteria applied to both datasets.

		Avg. Flagship Accuracy			Reliability Gap (pass^k − pass@k)			
Amendment Type	n	**Standard**	**Two-step**	**Sim. FC**	**Standard**	**Two-step**	**Sim. FC**	**Example utterance**
deictic reference (target item)								
substitution	50	80.7	69.7	87.5	16.0	17.1	12.5	No, wait. Make that a Big Mac.
modify (size)	20	99.3	97.1	97.9	0.0	0.0	0.0	Actually, make that a large Chocolate Shake.
modify (ingr.)	20	99.4	99.1	100.0	2.1	1.4	0.0	No pickles or onions on that.
zero anaphora (target item)								
substitution	50	89.7	93.9	98.8	12.6	9.7	5.5	I want a Dr. Pepper instead.
modify (ingr.)	20	99.4	99.1	100.0	2.9	2.9	0.0	I don't want onions or pickles.
modify (size)	20	100.0	99.9	99.9	0.0	0.7	0.7	Wait, no. Large fries.
indirect reference (target item)								
substitution	50	90.0	87.6	95.8	8.6	7.4	7.4	Let me change the shake. I want a Vanilla Shake instead.
add (diff. size)	80	91.5	93.9	92.5	9.8	6.5	10.0	Add one more of that soda, but make it small.
modify (ingr.)	20	93.3	92.4	94.6	3.6	3.6	2.1	Hold on. No ketchup or onions on that cheeseburger.
delete	20	93.7	96.9	98.1	2.1	2.1	3.6	Uhm...actually remove that burger from my order.
modify (size)	80	97.5	99.8	99.9	2.9	0.7	0.4	Go up a size on those french fries.
add (same size)	50	99.0	99.8	99.6	2.6	0.9	1.7	Wait. I'd like to add one more of the burger.
anchor property inheritance (new item size)								
substitution	110	79.9	75.8	90.5	16.8	16.4	11.7	No wait. Let me change that shake. I want a Vanilla Shake instead.
add (same size)	50	89.0	94.7	94.4	9.1	6.3	8.9	I'd like another of those fries.
directional (target field value)								
add (diff. size)	100	98.3	99.2	97.3	5.7	3.4	8.9	I want one more of that soda. But in a larger size.
modify (size)	100	98.7	99.7	99.9	2.1	0.9	0.3	Go up a size on those fries.
implicit field (target field)								
modify (ingr.)	70	97.5	97.0	98.4	2.7	2.9	0.6	Hold on. No ketchup or onions on that cheeseburger.
modify (size)	70	98.4	99.0	99.3	1.0	0.5	0.2	No wait. I want large fries instead.

Table 19: Amendment type×inference breakdown on the targeted dataset (k=5, 720 examples). Rows are grouped by inference type and the component it applies to (in parentheses). Indented rows show the amendment type and subtype. Shading indicates low accuracy or high reliability gap. Native FC is omitted as it is not supported by all 7 flagship models. Example utterances assume the order: Cheeseburger, medium fries, medium Coca-Cola, small Chocolate Shake.

G Error Analysis

Since each example has a known amendment type and structured JSON output, we deterministically classify errors by comparing predicted orders against expected ones—identifying which item is wrong and what property differs. We analyze flagship models under Standard on the targeted dataset (720 examples, k=5 runs). Of these, 201 examples (27.9%) have at least one flagship failure.

G.1 Error Taxonomy

Table 20 lists all error categories observed in flagship-failed examples. Categories are determined by the *dominant error*—the most frequent error description across all models that failed each example.

Category definitions.

- **Wrong/missing size on replacement**: The model substitutes the item correctly but omits or changes the size that should be inherited from the replaced item (e.g., replacing "small Sprite" with "Coca-Cola" but dropping the size, or making up "large Coca-Cola").
- **Added item missing**: The model fails to add the requested item entirely—typically when the target is referenced implicitly via synecdoche, lexical paraphrase, deixis, or zero anaphora (e.g., "add another of that drink" where "drink" must be resolved from context).
- **Wrong/missing size on new item**: The model adds the correct item but omits the size or assigns the wrong one, even though the menu requires a size and the anchor item's size is explicit in the order (e.g., "another fries" should inherit the existing size *small*); occasionally the model produces a made-up size.
- **Both old and new present**: In a substitution, the model adds the new item but fails to remove the old one, treating the utterance as an addition rather than a replacement—even with cues like "instead" or "actually."
- **Original item size changed**: An item not targeted for a size change has its size altered (e.g., "one more of that soda but make it large"—the model changes the existing Dr. Pepper from small to large instead of adding a second one).
- **Only replacement in order**: The model outputs only the substituted item, losing all other items from the order. This suggests the model interprets the substitution as replacing the entire order (e.g., "... and a large Chocolate Shake. Wait, not that. I'll have a medium Vanilla Shake instead."—the model outputs only the shake, losing three other items).
- **Original item lost**: An unrelated item disappears from the order (e.g., adding fries causes the salad to vanish).
- **Missing modification**: An ingredient modification is not applied to the target item (e.g., "Make that cheeseburger without pickles"—the quarter pounder w cheese does not receive the "no pickles" modification because the indirect reference is not resolved).
- **Target item missing**: The target item is deleted from the output instead of being modified—e.g., "no mustard or bacon on that cheeseburger" causes the model to remove the double quarter pounder entirely rather than applying the modification.
- **Target not deleted**: A deletion is not executed—the target item remains in the order (e.g., "I don't want that burger" but the double quarter pounder stays).
- **Extra modification**: A spurious modification is added that was not requested (e.g., an unrelated item gains "no bacon" when only a size change was requested).

G.2 Inference-to-Error Mapping

The "Inference link" column in Table 20 maps each error category to the inference type and amendment type that produce it. The dominant error categories group naturally by inference:

- **Inheritance**: wrong/missing size on replacement (n=61, 30.3% of errors) under substitution, and wrong/missing size on new item (n=35, 17.4%) under addition.
- **Target resolution**: added item missing (n=34, 16.9%) when indirect or deictic reference fails under addition; missing modification (n=5, 2.5%) and target item missing (n=4, 2.0%) under modification; target not deleted (n=3, 1.5%) under deletion.

Error category	n	%	Inference link
Wrong/missing size on replacement	61	30.3	inheritance (size) + substitution
Wrong/missing size on new item	35	17.4	inheritance (size) + addition
Added item missing	34	16.9	indirect/deictic reference (target) + addition
Both old and new present	24	11.9	implicit amendment type + substitution
Original item size changed	14	7.0	scope misinterpretation
Only replacement in order	11	5.5	scope misinterpretation
Missing modification	5	2.5	indirect reference (target) + modification
Original item lost	5	2.5	scope misinterpretation
Target item missing	4	2.0	indirect reference (target) + modification
Target not deleted	3	1.5	indirect reference (target) + deletion
Extra modification	3	1.5	scope misinterpretation
Old item kept, replacement missing	2	1.0	implicit amendment type + substitution
Total	201	100.0	

Table 20: Error taxonomy for flagship-failed examples under Standard on the targeted dataset (720 examples, k=5 runs, 7 flagship models). Categories are determined by the dominant error across all models that failed each example. Inference link shows the inference type and amendment type that produce the error.

- **Implicit amendment type**: both old and new present (n=24, 11.9%) and old item kept, replacement missing (n=2, 1.0%) under substitution.
- **Scope misinterpretation**: original item size changed (n=14, 7.0%), only replacement in order (n=11, 5.5%), original item lost (n=5, 2.5%), and extra modification (n=3, 1.5%).

Note that some errors (e.g., scope misinterpretation) are not tied to a single inference type and can co-occur with any of the above.

G.3 Examples with Explanations

Below are illustrative examples from each major error category, showing the original order, the amendment utterance, the expected output, and what the model produces instead.

Wrong/missing size on replacement.

Original: "I'd like a cheeseburger, one medium salad, a small Sprite and a medium Strawberry Shake."
Amendment: "No wait. No small Sprite. I'll have a Coca-Cola instead."
Expected: cheeseburger, salad (medium), Coca-Cola (**small**), Strawberry Shake (medium)
Error: Models output Coca-Cola with no size (flagship accuracy: 5.7%). The size "small" must be inherited from the Sprite being replaced.

Added item missing.

Original: "Can I order the cheeseburger, the medium fries, a large Sprite and a Vanilla Cone."
Amendment: "Add one more of that drink to my order. Size medium."
Expected: cheeseburger, fries (medium), Sprite (large), Vanilla Cone, **Sprite (medium)**
Error: All 7 flagships fail to add the second Sprite (flagship accuracy: 0%). The indirect reference "drink" must be resolved to "Sprite" from context.

Both old and new present.

Original: "I'll have 20-piece chicken nuggets, medium fries, a small water and one large Chocolate Shake."
Amendment: "Actually I want a large Coca-Cola instead."
Expected: chicken nuggets (20), fries (medium), **Coca-Cola (large)**, Chocolate Shake (large)
Error: Models keep the water and add the Coca-Cola (flagship accuracy: 62.9%). Despite "instead" signalling substitution, models treat this as an addition because the target (water) is never named.

Wrong/missing size on new item.

Original: "I want one quarter pounder with cheese, the small fries, the large Sweet Tea and one large Strawberry Shake."
Amendment: "Also another fries."
Expected: ..., fries (**small**), ..., fries (**small**)
Error: Models add fries without a size (flagship accuracy: 25.7%). The size "small" must be inherited from the existing fries in the order.

Target not deleted.

Original: "I'm having the double quarter pounder with cheese, the large fries, one medium water and a Vanilla Cone."
Amendment: "Actually I don't want that burger."
Expected: fries (large), water (medium), Vanilla Cone
Error: Models keep the double quarter pounder (flagship accuracy: 31.4%). "That burger" is an indirect reference to "double quarter pounder with cheese."

Only replacement in order.

Original: "I want a double quarter pounder with cheese, the large fries, one small Sprite and one small Strawberry Shake."
Amendment: "Wait, not that. I'll have a medium Vanilla Shake instead."
Expected: double quarter pounder with cheese, fries (large), Sprite (small), **Vanilla Shake**

(medium)
Error: Models output only the Vanilla Shake, losing all other items (flagship accuracy: 28.6%). The model interprets "not that" + new item as replacing the entire order rather than substituting one item.

Original item size changed.

Original: "I'm getting one Filet-O-Fish, apple slices, and a small Dr. Pepper."
Amendment: "Let me have one more of that soda. Make it large."
Expected: Filet-O-Fish, apple slices, Dr. Pepper (**small**), Dr. Pepper (**large**)
Error: Models change the existing Dr. Pepper from small to large instead of adding a second one (flagship accuracy: 45.7%). "Make it large" is applied to both the original item rather than the new addition.

H Clarification Experiment

H.1 Protocol

We extend the Standard prompt with an optional `ask_clarification` function call. The model either produces the final order JSON directly, or invokes the function to ask for clarification about a list of components it considers unclear or underspecified:

```
{"ask_clarification":
  {"missing_slots":
    ["operation", "target_item",
    "new_item", "new_item_size"]}}
```

We then return the ground-truth values for those components in explicit natural language—e.g., "The operation is substitution. The target item is the Sprite. The new item is Coca-Cola. The size of the new item is medium."—and the model then produces the final order. At most one clarification call (with multiple slots) is permitted per example.

Available slots. Five slots correspond to components that may be implicit: `operation` (add/delete/modify/substitute), `target_item` (which item to act on), `target_field` (size or modifications; modify only), `target_field_value` (the resulting value; modify only), and `new_item_size` (size of new/substituted item). Two components are excluded: `new_item` is always explicit in our data; `amendment_offer` is subsumed by `operation` (knowing the operation implies an amendment was offered).

H.2 Datasets

Clarification subset (324 examples). Selected from the targeted dataset: 145 examples where at least one flagship model erred under Standard (k=5), 100 controls where all flagships were correct (stratified by implicit component count), 55 additional modify examples ensuring $n \geq 30$ per slot type, and 30 fully-explicit examples. Six genuinely ambiguous examples are excluded from accuracy comparisons (effective n=324).

Underspecified examples (50). Sourced from the targeted dataset exclusion list (§C.3): genuinely ambiguous utterances where the referent cannot be determined from context alone—e.g., "Another order of that. But small this time." Analysis focuses on clarification behaviour rather than accuracy.

H.3 Metrics

All metrics averaged over k=5 runs. **Acc**: order-level exact match. **Clar%**: fraction of examples where the model asks. **R|ask**: of the truly implicit slots in examples where the model asked, what fraction did it identify? **Prec**: of all slots requested, what fraction were truly implicit? **Unnec.**: false alarm rate on fully-explicit examples.

Nine models were evaluated ($\star$ = flagship in Table 21): GPT-5, GPT-4o, Claude Opus 4.6, Claude Sonnet 4.5, Gemini 2.5 Pro, Hermes 3 405B, DeepSeek v3.2 (flagship); Claude Haiku 4.5, Gemini 2.5 Flash (non-flagship).

H.4 Results

Clarification improves flagship accuracy by +5.4 pp on the clarification subset and +19.8 pp on underspecified examples. Models ask on only 11% of clarification-subset examples (26% on underspecified), but when they do, component precision exceeds 90% and conditional recall is ∼52%. Unnecessary questions–asked on fully explicit examples–are near-zero ($\leq$0.8%). Clarification rate increases with genuine ambiguity (26% vs. 11%). Model variation is substantial: Opus achieves 100% precision but asks rarely (4.8%); GPT-5 balances rate (16.7%) and quality (R|ask 58.6%, Prec 97%); DeepSeek and Haiku rarely ask ($\leq$6%) and show minimal benefit.

		Accuracy (%)			Clarification Quality (%)			
Model		**Clar**	**Std**	**ΔStd**	**Clar%**	**R|ask**	**Prec**	**Unnec.**
Clarification subset (324 examples)								
Gemini 2.5 Pro	⋆	96.7	83.1	+13.5	13.5	54.5	98.9	0.0
GPT-5	⋆	94.1	82.1	+12.0	16.7	58.6	97.0	0.0
Gemini 2.5 Flash		93.6	82.4	+11.2	16.9	51.7	97.4	0.4
Hermes 3 405B	⋆	93.8	91.5	+2.3	7.2	44.0	87.3	0.4
Claude Opus 4.6	⋆	90.7	83.8	+6.9	4.8	62.4	100.0	0.0
GPT-4o	⋆	90.6	88.6	+1.9	18.5	51.2	81.1	0.4
Claude Sonnet 4.5	⋆	89.8	86.3	+3.5	12.0	52.1	91.8	0.0
Claude Haiku 4.5		86.7	88.6	−1.9	5.4	47.9	87.0	1.5
DeepSeek v3.2	⋆	85.9	88.0	−2.1	4.0	43.5	75.7	0.8
Avg. Flagship (7)		91.7	86.2	+5.4	11.0	52.3	90.2	0.2
Avg. All (9)		91.3	86.1	+5.3	11.0	51.8	90.7	0.4
Underspecified examples (50)								
Claude Sonnet 4.5	⋆	52.0	6.8	+45.2	58.0	41.0	65.9	—
Gemini 2.5 Flash		60.0	26.0	+34.0	40.4	36.1	86.5	—
GPT-4o	⋆	32.8	1.6	+31.2	34.0	43.4	71.5	—
Gemini 2.5 Pro	⋆	44.4	20.0	+24.4	26.4	40.5	100.0	—
Hermes 3 405B	⋆	29.2	14.0	+15.2	14.0	47.1	87.5	—
GPT-5	⋆	45.2	30.8	+14.4	28.4	40.8	94.9	—
Claude Haiku 4.5		16.4	6.4	+10.0	22.0	37.3	98.3	—
DeepSeek v3.2	⋆	11.2	5.6	+5.6	3.2	42.5	53.3	—
Claude Opus 4.6	⋆	32.0	29.2	+2.8	19.2	42.9	100.0	—
Avg. Flagship (7)		35.3	15.4	+19.8	26.2	42.6	81.9	—
Avg. All (9)		35.9	15.6	+20.3	27.3	41.3	84.2	—

Table 21: Clarification results, averaged over k=5 runs. Clar = clarification accuracy; Std = Standard accuracy; ΔStd = change vs. Standard (pp); Clar% = clarification rate; R|ask = component recall when asking; Prec = component precision; Unnec. = unnecessary questions, asked on fully explicit examples (not applicable to underspecified set). ⋆ = flagship model. Clarification subset sorted by accuracy; underspecified examples sorted by ΔStd.

I Implementation Details

I.1 Models and APIs

Table 22 lists all models evaluated, grouped by provider, with the resolved API identifier string. OpenAI aliases resolve server-side to dated snapshots. DeepInfra identifiers omit the namespace prefix for brevity (full paths use `org/model` format). All experiments use the provider's default sampling temperature (GPT-5 models do not support temperature 0). Each example is queried k=5 independent times to compute average accuracy, pass@k, and pass^k. Total API cost across all experiments is estimated at $3,000–4,000.

For Standard (including Two-step), Simulated FC, and Clarification, structured JSON output is enforced via prompt instructions combined with each provider's JSON mode. Anthropic is the exception: its JSON mode requires a schema and validates against it, unlike the other providers' unvalidated "respond with JSON" modes. To keep conditions comparable, we use prompt instructions only for Anthropic, which consistently output the correct JSON format anyways. Native FC uses each provider's function-calling mode instead.

OpenAI. JSON: prompt instructions + `response_format: json_object`. Native FC: strict mode, `tool_choice: "required"` (must call a tool).

Anthropic. JSON: prompt instructions only (see above). `max_tokens: 1024`. Native FC: `tool_choice: {"type": "any"}` (must call a tool).

Google. JSON: prompt instructions + `response_mime_type: "application/json"`. Native FC: `function_calling_config(mode="ANY")` (must call a function), with `size` and `modifications` optional (Google rejects empty-string enum values).

DeepInfra. JSON: prompt instructions + OpenAI-compatible `response_format: json_object`. Native FC: `tool_choice: "auto"` (model decides whether to call a tool), unlike the other providers which force a tool call. We tried values like "required" and "any", but those API calls threw errors. This may have contributed to the tool-use issues reported for these models.

Where available, we use provider batch APIs (OpenAI Batch API, Google Gemini Batch API) for cost-efficient bulk inference. Batch APIs were not used for turn-taking settings (Two-step and Clarification) or for Gemini Native FC (unsupported). Examples that produced API errors—rate limits, timeouts, model unavailable, etc.—were automatically re-run until a valid response was obtained, ensuring complete coverage across all model–example pairs.

Model	API identifier
OpenAI (Chat Completions)	
GPT-5	gpt-5-2025-08-07
GPT-5-mini	gpt-5-mini-2025-08-07
GPT-5-nano	gpt-5-nano-2025-08-07
GPT-4o	gpt-4o-2024-08-06
GPT-4o-mini	gpt-4o-mini-2024-07-18
GPT-4.1	gpt-4.1-2025-04-14
GPT-4.1-mini	gpt-4.1-mini-2025-04-14
GPT-4.1-nano	gpt-4.1-nano-2025-04-14
Anthropic (Messages)	
Claude Opus 4.6	claude-opus-4-6
Claude Opus 4.5	claude-opus-4-5
Claude Sonnet 4.5	claude-sonnet-4-5
Claude Haiku 4.5	claude-haiku-4-5
Google (Gemini)	
Gemini 2.5 Pro	gemini-2.5-pro
Gemini 2.5 Flash	gemini-2.5-flash
Gemini 2.5 Flash Lite	gemini-2.5-flash-lite
DeepInfra (OpenAI-compat.)	
Hermes-3 405B	Hermes-3-Llama-3.1-405B
DeepSeek-V3.2	DeepSeek-V3.2
DeepSeek-V3.1-T	DeepSeek-V3.1-Terminus
Llama-3.1 8B	Meta-Llama-3.1-8B-Instruct
Llama-3.1 70B	Meta-Llama-3.1-70B-Instruct
Llama-3.3 70B	Llama-3.3-70B-Instruct-Turbo
MiniMax-M2.5	MiniMax-M2.5
MiniMax-M2.1	MiniMax-M2.1

Table 22: Models evaluated with resolved API identifiers. OpenAI (OpenAI, 2025b,a, 2024); Anthropic (Anthropic, 2026, 2025b,a); Google (Google DeepMind, 2025); DeepInfra (Grattafiori et al., 2024; Teknium et al., 2024; DeepSeek-AI, 2024; MiniMax, 2026).

I.2 Prompt Templates

Every prompt begins with the same preamble and menu reference:

```
You are a McDonald's Drive-Through Assistant.
Your job is to accurately record customer orders
as structured JSON.

<METHOD-SPECIFIC INSTRUCTIONS — see below>

Use the following McDonald's MENU to identify
valid items, their respective categories (roles
in a food order), sizes, and modifications:
<MENU JSON — see Table 6>

<METHOD-SPECIFIC CLOSING — see below>
```

The method-specific portions are shown below.

Standard / Two-step

Standard and Two-step share the same system prompt. In Two-step, the model first receives the order utterance alone (without the amendment) to produce an initial order JSON; the amendment is then sent as a follow-up turn.

```
Output a JSON object with the following structure:
{
  "order": {
    "items": [...]
  }
}

Each item in the "items" array should have:
- "name": the item name (string)
- "category": one of "main", "side", "drink",
  "dessert" (string)
- "size": the size if specified, otherwise ""
  (string, empty string if no size)
- "modifications": list of modifications
  (e.g., ["no pickles", "no onions"]),
  or [] if none (array)

If a customer orders multiple of the same item,
include that item multiple times in the array.

Example:
User: "I want a Big Mac, large fries, a medium
       Coca-Cola, and a small Vanilla Shake."
Assistant:
{
  "order": {
    "items": [
      {"name": "Big Mac",
       "category": "main",
       "size": "", "modifications": []},
      {"name": "fries",
       "category": "side",
       "size": "large", "modifications": []},
      {"name": "Coca-Cola",
       "category": "drink",
       "size": "medium", "modifications": []},
      {"name": "Vanilla Shake",
       "category": "dessert",
       "size": "small", "modifications": []}
    ]
  }
}
```

After the menu, Standard/Two-step appends:

```
Customers may change their order during the
conversation. The final JSON should reflect only
what they want after all changes.

Respond ONLY with the JSON object.
Do not include any other text.
```

Simulated Function Calling (Sim. FC)

For both FC methods, the ground-truth order JSON is injected as an assistant message before the amendment utterance, so the model receives the correct order as context. The model outputs a parameterized edit operation as a JSON object; a deterministic executor applies it to produce the final order, which is compared against the ground truth. The method-specific instructions replace the output format:

```
You will receive a customer order conversation.
Always respond with a JSON object containing an
"operations" list.

Case 1 — No recorded order yet: output an "add"
  operation for each item the customer orders.
Case 2 — A recorded order (assistant message)
  is present and the customer requests a change:
  output only the operation(s) that describe
  the change.

These are the available operations:

1. add — Add a new item to the order.
{"operation": "add",
 "item": {"name": "...",
   "category": "...", "size": "...",
   "modifications": [...]}}

2. delete — Remove an existing item.
{"operation": "delete",
 "target": {"category": "...",
   "name": "..."}}

3. modify — Change size/modifications.
{"operation": "modify",
 "target": {"category": "...",
   "name": "..."},
 "changes": {"size": "...",
   "modifications": [...]}}

4. substitute — Replace with a different
   item in the same category.
{"operation": "substitute",
 "target": {"category": "...",
   "name": "..."},
 "new_item": {"name": "...",
   "category": "...", "size": "...",
   "modifications": [...]}}

Field values:
- "category": "main","side","drink","dessert"
- "size": "small","medium","large","6",
  "10","20", or ""
- "modifications": subset of ["no pickles",
  "no lettuce", ...], or []
```

```
<in-context examples omitted for brevity>
```

After the menu, Sim. FC appends:

```
Respond ONLY with the JSON operation(s).
Do not include any other text.
```

Native Function Calling (Native FC)

Native FC uses the same injected-order setup as Sim. FC (see above) but has no method-specific instructions before the menu. After the menu, it appends:

```
Use the provided tools to process the
customer's request.
- For a new order (no prior assistant message),
  call add_item once per item.
- For an amendment to an existing order
  (assistant message present), call the
  appropriate tool(s): add_item, delete_item,
  modify_item, or substitute_item.
```

The four tool functions are passed via each provider's native tool-calling API. Below is the OpenAI format (strict mode); Anthropic uses `input_schema`, Google uses `functionDeclarations`. Google rejects empty-string enums, so `size` is optional rather than required; Anthropic similarly makes `size` and `modifications` optional.

```
add_item(name, category, size, modifications)
delete_item(category, name)
modify_item(category, name, size, modifications)
substitute_item(category, name, new_name,
                new_size, new_modifications)

Shared enums:
  category: main | side | drink | dessert
  size:     "" | small | medium | large | 6
            | 10 | 20
  modifications: [no pickles, no lettuce, ...]
```

All fields are required in the OpenAI strict-mode schema; Google and Anthropic make `size` and `modifications` optional due to enum handling differences.

Clarification

Clarification uses the Standard prompt (output format + menu) and appends the following clarification instruction:

```
IMPORTANT: If the customer's request is ambiguous
or you cannot determine all the information needed
to update the order, you may ask ONE clarification
question. You will receive a response with the
values for the slots you asked about, and then you can
produce the final order.

To ask for clarification, respond with ONLY a JSON
object in this format:
{"ask_clarification":
  {"missing_slots": [<list of slots>]}}

The available slots you can ask about are:
- "operation": whether the customer wants to add,
  delete, modify, or substitute.
- "target_item": which item the customer wants
to act on.
- "target_field": which attribute (size vs
  modifications) to change (modify only).
- "size": the resulting size of the item being
  modified (modify only).
- "new_item_size": what size a new or replacement
  item should be (add/substitute only).

Rules:
- You may list MULTIPLE missing slots in a single
  call if more than one piece of information is
  ambiguous.
- You may call ask_clarification AT MOST ONCE.
- If the amendment is clear and you can determine
  all needed information, skip clarification and
  respond directly with the final order JSON.
- After receiving the customer's clarification
  response, respond with the final order JSON.

Example:
Order: "I want a cheeseburger, medium fries,
and a small Sprite. Actually, make that bigger."
The target item and resulting size are unclear,
so you respond:
{"ask_clarification":
  {"missing_slots": ["target_item", "size"]}}

The customer then clarifies: "I'm talking about
the fries. Make it large."
You then respond with the complete updated
order JSON.
```