PREFERENCE REASONING UNDER INDETERMINACY IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models evolve into decision-making agents, the ability to reason over preferences becomes fundamental to alignment, coordination, and collective intelligence. Yet, unlike standard benchmarks, real-world preference reasoning is inherently *indeterminate*: information may be incomplete, and valid solutions may not exist. We argue that indeterminacy, rather than correctness alone, is a central challenge for AI reasoning. We formalize this challenge along two axes, (i) *epistemic indeterminacy*, arising from incomplete, partial, or expressive preferences, and (ii) *structural indeterminacy*, arising from the non-existence of solutions under standard social choice concepts. Across a hierarchy of tasks, we show that language models systematically fail to distinguish between determined and undetermined instances, exhibiting miscalibrated reasoning even in verification settings.

1 INTRODUCTION

Large Language Models (LLMs) have evolved from statistical language generators into systems capable of executing increasingly complex reasoning tasks, including logical inference and algorithmic problem solving. This progression signals a shift toward models that operate over structured representations rather than purely linguistic patterns, enabling reasoning over preferences.

Preference reasoning is a foundational component of modern AI systems, spanning across alignment, fine-tuning, and recommender systems. In agentic settings, LLMs are tasked with acting on behalf of users, requiring either *implicit* or *explicit* inference, comparison, and aggregation of potentially conflicting preferences when interacting with environments and other agents. At scale, these mechanisms extend to collective decision-making, where models are used to elicit and aggregate preferences into social judgments (Fish et al., 2023; Conitzer et al., 2024), as in platforms such as Pol.is, Remesh, and deliberative frameworks like the Habermas Machine (Tessler et al., 2024).

Despite these advances, current evaluation paradigms for LLM reasoning rely primarily (and often exclusively) on *closed-world benchmarks* in which a ground-truth solution is often assumed to exist. However, real-world decision-making is inherently more complex, often characterized by undetermined scenarios in which preferences are incomplete or structural constraints preclude the existence of a valid solution under a given objective (aka a solution concept). This motivates a shift in assessing whether a model can distinguish between what is determined (entailed by axioms or existing as a solution) and what is undetermined (not entailed or non-existent). With this lens, a central axis of evaluation extends beyond correctness to whether models can identify the boundary between determinacy and indeterminacy: *Can advanced reasoning models identify whether a preference query is not answerable or whether a solution is infeasible under given specification?*

We study preference reasoning under two distinct forms of *indeterminacy*: (i) **epistemic indeterminacy**, arising from incomplete preference information, and (ii) **structural indeterminacy**, arising from the interaction between preference structure (e.g., ties) and solution concepts, which may render objectives infeasible even when preferences are fully specified.[1] This view is formally aligned with the Open-World Assumption (OWA) and multi-valued semantic frameworks such as Kleene's

[1]We avoid using the term "decidable/undecidable" since decidability is often used to imply computational tractability. In this paper, we are concerned about epistemological determinacy when efficient algorithms exist.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

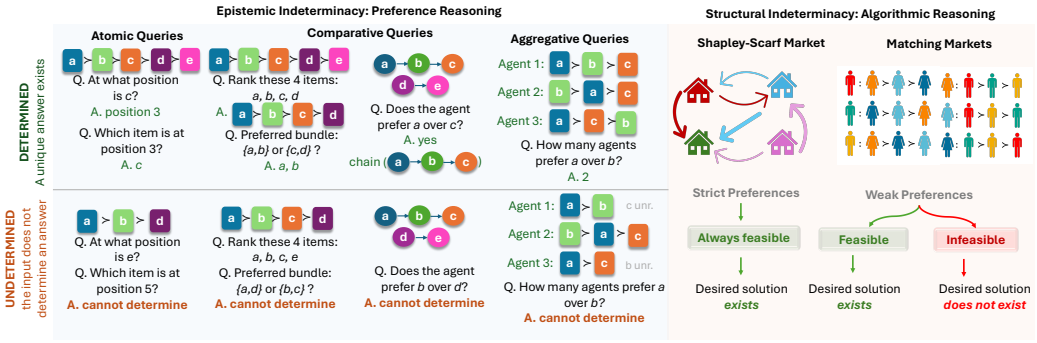


Figure 1: Overview of our taxonomy of evaluating LLMs' reasoning with indeterminacy.

three-valued logic ($K3$), in which propositions may take an explicit "unknown" or "indeterminate" truth value (Kleene, 1952).

We ground our evaluation in classical economic problems over ranked preferences, which provide a principled testbed for reasoning under constraints. These settings require models to interpret incomplete preferences and ties, resolve conflicts across agents, and satisfy global objectives defined by axiomatic solution concepts such as stability and welfare. Crucially, these concepts distinguish between feasible and infeasible outcomes, making this domain ideal for evaluating not only reasoning accuracy but also indeterminacy. Consequently, we transform indeterminacy into an observable and measurable failure mode of LLMs.

1.1 Main Results

We study preference-based reasoning in LLMs and introduce a formal taxonomy of determined vs. undetermined reasoning across increasing levels of complexity: (i) *atomic queries*, requiring retrieval from a single preference (e.g., what is the rank of alternative 'a'?); (ii) *comparative queries*, requiring entailment or refutation of relations (e.g., whether 'a' is preferred to 'b'); (iii) *aggregative queries*, requiring aggregation across multiple preferences (e.g., how many agents prefer 'a' over 'b'?); and (iv) *structural (algorithmic) queries*, requiring the construction of outcomes satisfying social choice solution concepts (e.g., finding a matching solution in the *core*).

We leverage the nuanced interaction between *preference expressivity* (e.g., partial orders, ties, and incomplete lists) and solution concepts from social choice to generate a spectrum of complex reasoning tasks within an axiomatic framework of indeterminacy. Crucially, all tasks—both query resolution and feasibility determination—are computable in polynomial time using well-established combinatorial algorithms, isolating reasoning, rather than computational hardness, as the primary challenge for LLMs. Our main findings are as follows.

1. **Epistemic Indeterminacy**: LLMs perform significantly worse on undetermined questions compared to determined ones, owing to systematic assumptions (e.g. lexicographic ordering of preferences) made when inputs are under-specified. Providing an explicit indeterminacy option helps on a subset of tasks, but models continue to exhibit systematic reasoning errors.
2. **Structural Indeterminacy**: LLMs' performance on tasks requiring algorithmic reasoning degrades rapidly with market size—even in settings where solutions are guaranteed to exist—and deteriorates further in the presence of structural infeasibility, with models failing both to correctly identify infeasible instances and to generate valid solutions when they do exist. Providing an **indeterminacy option** (aka "return null when infeasible") improves infeasibility detection but induces systematic bias, leading models to incorrectly declare non-existence on feasible instances.
3. **Verification of Solution Concepts**: Even in selection tasks where only verification is required (as opposed to generation), LLMs continue to select incorrect options even when valid solutions are present. Although NOTA improves average accuracy, models seldom use this option, rarely abstaining even when no valid option is present, which reflects poor calibration in distinguishing feasible from infeasible cases. In addition, LLMs exhibit a systematic **intention–action misalignment** in preference reasoning: even when they appear to target a specific solution concept, the

outcomes they select frequently do not match that concept, including in selection tasks where only verification is required.

4. **Assisted Reasoning**: We consider assisted reasoning under two settings: **(i) refinement via feedback**, where models are given violations of the target property and attempt to iteratively repair invalid solutions, and **(ii) reasoning with code execution**, where models generate and execute programs to solve instances. While both settings improve performance, the gains are largely driven by brute-force enumeration on small markets and heuristic search on larger instances, neither of which scales to deployment-relevant market sizes.

1.2 Related Work

LLM Reasoning and Abstention. A growing body of work evaluates LLMs as procedural reasoners. Recent benchmarks show that LLMs fail to reliably execute classical algorithms as instance size grows (Hosseini et al., 2025; Shojaee et al., 2025). A separate but related literature studies whether models know when to abstain: frontier models systematically miscalculate their uncertainty (Kapoor et al., 2024; Yona et al., 2024), hallucinate confidently even on questions they could answer correctly (Simhi et al., 2025; Abbasi-Yadkori et al., 2024), and in many settings fail to abstain at all (Kirichenko et al., 2025). Our findings sit at the intersection of these threads. Our findings sit at their intersection, linking procedural failure with abstention failure in preference reasoning.

Beyond the fact that these benchmarks pose questions in natural language and do not consider structured preferences, two differences matter. First, because the ways of resolving an underdetermined query are enumerable in our setting, we identify the specific assumption a model substitutes for the missing information (e.g., ordering bundles by their highest-ranked item) rather than only whether it failed to abstain. Second, and more importantly, our benchmark includes problems for which the requested solution does not exist at all, a case with no counterpart in the abstention setting.

LLMs in Economic Settings. Adjacent literature evaluates LLMs as agents in economic and social-choice contexts. Computational social choice, an extensively studied domain on which we build (Long & Velez, 2021; Amanatidis et al., 2022; Echenique et al., 2023), has begun engaging with LLMs both as fairness-aligned allocators (Hosseini & Khanna, 2025; Cookson et al., 2026) and as solvers of canonical solution concepts (Hosseini et al., 2025; Fish et al., 2025), alongside applications to voting and participatory budgeting (Yang et al., 2024; Thach et al., 2026). A parallel line of work uses LLMs as proxies in preference elicitation pipelines (Handa et al., 2024; Huang et al., 2025; Li et al., 2025), to which our work also contributes by characterizing how reliably LLMs parse structured preferences. Appendix C contains an extended related work.

2 Preference-Based Tasks and Methodology

2.1 Formalizing Problems and Solution Concepts

Problem Domains. We consider three economic problems requiring reasoning over preferences: **house (or object) allocation** (Svensson, 1999), **Shapley-Scarf housing markets** with endowments (Shapley & Scarf, 1974), and **two-sided matching markets** (Gale & Shapley, 1962), each with increasing structural complexity and distinct axiomatic solution requirements. Let A denote a set of agents and B a set of alternatives (objects). Each agent $i \in A$ is endowed with a preference relation $\succeq_i$ over B, which is a weak and potentially partial order. We write $b_1 \succ_i b_2$ if agent i strictly prefers b_1 to b_2, and $b_1 \succeq_i b_2$ if i weakly prefers b_1 to b_2, allowing indifference. In two-sided markets each $b \in B$ likewise holds a preference relation $\succeq_b$ over A. In settings with incomplete preferences, $\phi \succ_i b$ indicates that b is not present in agent i's preference list and is thus unranked or incomparable. A **preference profile** is the collection of all agents' preferences, denoted by $\succeq = (\succeq_{a_1}, \ldots, \succeq_{a_m}, \succeq_{b_1}, \ldots, \succeq_{b_n})$, where $m = |A|$ and $n = |B|$.

In a *house allocation* problem, agents in A are assigned objects in B with no initial endowments; in the *Shapley–Scarf housing market*, each agent $i \in A$ is initially endowed with an object $e_i \in B$, forming an exchange economy; and in *two-sided matching markets*, agents in A and B both have preferences over each other, inducing bilateral matching constraints. A **matching** is a mapping $\mu : A \cup B \to A \cup B$ such that for all $a \in A$, $\mu(a) \in B \cup \{\emptyset\}$ and for all $b \in B$, $\mu(b) \in A \cup \{\emptyset\}$, with each agent and object matched to at most one counterpart and $b = \mu(a)$ if and only if $a = \mu(b)$.

Table 1: Taxonomy of preference reasoning tasks, each exhibiting determined or undetermined instances depending on preference expressivity and feasibility of solutions.

Task Type	Subtype	Example (Determined / Undetermined)
Atomic	Rank retrieval	Determined: rank of a in $\succeq_i$ Undetermined: rank of b under incomplete list
Comparative	Pairwise, RS-extension, ranking	Determined: $a \succ_i b$ holds Undetermined: $a \succ_i b$ under partial information
Aggregative	Preference aggregation (counts)	Determined: $\#\{i : a \succ_i b\}$ Undetermined: count under missing comparisons
Structural	Core, stability (weak/strong/super)	Determined: existence of stable matching Undetermined: existence under ties / incompleteness

Solution Concepts. Preferences and market structure induce standard **solution concepts**: in house allocation, the primary objective is **Pareto optimality**; in Shapley–Scarf markets, the central solution concept is the **core**, where no coalition of agents can reallocate their endowments to make all members strictly better off; and in two-sided matching markets, the standard concept is **stability**, requiring that no blocking pair exists. All three settings admit polynomial-time algorithms for computing canonical solutions and verifying feasibility under standard assumptions.

Stronger Notions under Ties. When preferences admit ties, the canonical solution concepts above split into refinements that differ in their robustness to indifference. In Shapley–Scarf markets, a **weak core** allocation is one that no coalition can strictly improve upon, while the **strict core** requires that no coalition can find a reallocation under which every member is weakly better off and at least one is strictly better off. In two-sided matching, an analogous hierarchy arises: a matching is **weakly stable** if no pair strictly prefers each other to their current partners; **strongly stable** if no pair contains an agent who weakly prefers a partner who strictly prefers them back; and **super stable** if no pair contains agents who weakly prefer each other. The stronger refinements (strict core, strong stability, super stability) may be infeasible for some instances, generating the structural indeterminacy we study in Section 4. Appendix D.1 provides detailed formalisms, and a review of the relevant algorithms.

2.2 Methodology and Experimental Setup

Preference Expressivity and Reasoning Tasks. Preference expressivity gives rise to nuanced query tasks and solution concepts; we consider a spectrum of structures ranging from strict complete orders (SO), strict but possibly incomplete orders (SI), complete orders with ties (TO), and incomplete orders with ties (TI), as well as general partial orders (See Table 2). Queries follow the hierarchy of atomic, comparative, aggregative, and structural (algorithmic) queries introduced in Section 1.1, and Table 1 gives a determined and an undetermined example of each type.

Queries. For preference queries, we distinguish between **determined** queries, whose answers are uniquely implied by the input preferences, and **undetermined** queries, where the available information is insufficient to resolve the query. Formally in logic, the former corresponds to whether the query is *entailed* (or *refuted*) by the preference instance, while the latter implies lack of information. For example, preference $a \succ c$ is determined if it is a necessary consequence of the provided axioms (e.g., $a \succ b$ and $b \succ c$ via transitive closure). Otherwise, it is undetermined. An undetermined query is scored correct only if the response indicates that the answer cannot be determined. A response that commits to an answer counts as incorrect even when it names the assumption it relied on. Appendix G.3 re-scores those responses as correct and shows that the gap is essentially unchanged.

For algorithmic queries, given an instance, a solution concept is **feasible** if there exists a solution satisfying the concept, and **infeasible** otherwise. For example, an instance of a matching market may admit *no* super stable matching, in which case it is infeasible with respect to super stability.

This taxonomy enables us to move beyond "accuracy" and evaluate the Invalid Rate, i.e. the frequency with which a model produces a "determined" response to an "undetermined" query. This measure captures speculative completion, or the *hallucination of certainty* (Simhi et al., 2025; Abbasi-Yadkori et al., 2024), providing a principled diagnostic of a key failure mode in LLM reasoning.

Models. We evaluate four large language models that achieve state-of-the-art results in reasoning and coding benchmarks (see, for example (Rein et al., 2023; Jimenez et al., 2024)): **GPT-5.2** (Singh

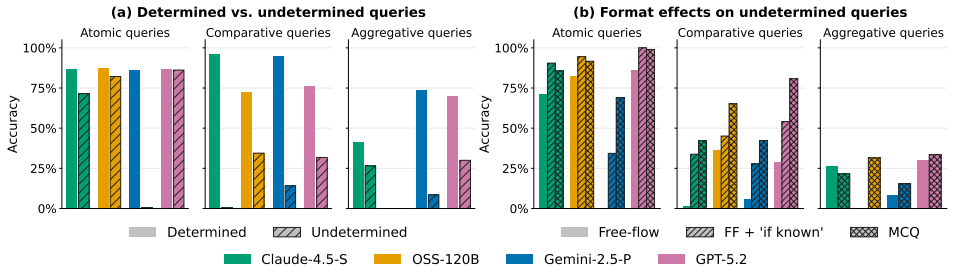


Figure 2: Comparing LLMs' performance (a) on determined vs. undetermined preference queries, and (b) with different prompting formats on undetermined preference queries, split by task type.

et al., 2025), **Gemini-2.5-Pro** (Deepmind, 2025), **Claude-4.5-Sonnet** (Claude-4.5-S) (Anthropic, 2025), and a frontier-class open-source model **OSS-120B** (Agarwal et al., 2025).

Prompt Generation. Each model was queried 30 times on a given question type (See Appendix J for the prompt-templates used for each type of task). Each query is a zero-shot, single-turn setting at the default temperature. In addition, we evaluated the models through two critical variations: i) **multi-shot refinements with feedback**, enabling models to refine and evaluate responses, and ii) **code-execution capability**, enabling multiple iterations and with the ability to write code and verify responses (Section 5). Throughout the paper, we report results at four reference sizes by the number of agents/items n: **Small** (n=10), **Modest** (n=30), **Medium** (n=50), and **Large** ($n \geq 100$).

3 Epistemic Indeterminacy: Preference Reasoning

We begin with epistemic indeterminacy: cases where a question is not answerable based on the information provided in the input preferences. Across the three query families introduced in Section 2.2 (atomic, comparative, and aggregative), we vary preference expressivity (i.e. completeness, ties, partial orders) to construct analogous questions of *determined* and *undetermined* instances on the same profile [2] (see Appendix E.1 for details on how each type of question is designed). Figure 1 illustrates a few examples per query type. Our experiments investigate three criteria, (i) how performance depends on whether a query is determined, (ii) how it scales with input size, and (iii) how it depends on the way the question is framed. [3]

Determined vs. Undetermined Queries. We compare performance on determined and undetermined queries across the question types in Figure 1, considering Medium and Large preferences. To reflect a more realistic setting, queries are issued in a *free-flow* format that does not provide explicit options to choose from. Responses are scored against the ground truth on determined queries, and on whether the model correctly indicates that the question is unanswerable, on undetermined ones (see Appendix J for the exact prompt formats). Figure 2 (a) shows that accuracy on undetermined queries is substantially lower than on the matched determined queries, collapsing to near-zero in several settings. The gap is driven by assumptions the models silently impose on the input, which suppress the underspecification rather than flag it. The gap is smallest on atomic queries, where the model only has to notice that an item is missing from the list, and largest on comparative queries, where nothing is missing and it has to work out that no answer follows from the preferences it was given.

LLMs' Systematic Biases. We illustrate two such patterns, on bundle comparisons and on partial-order pairwise queries. Two bundles are *responsive set* (RS) incomparable when neither can be paired with the other via an injection that maps each item to a weakly preferred counterpart. Intuitively, neither bundle dominates the other item-by-item. Resolving such a comparison requires an extra-axiomatic assumption, such as the lexicographic rule, which compares bundles by their most-preferred item, breaking ties by the next-most-preferred.

[2] By default, we use structured JSON format to express preference profiles as inputs to the models. In Appendix G.5 we demonstrate how our findings replicate when preferences are rendered in natural language.

[3] We report model accuracy, aggregated by problem types in the plots. Statistical analyses and paired comparisons are provided in the corresponding tables in the appendices.

As Figure 3 (a) shows, all four models default to lexicographic ordering on almost every RS-incomparable pair, across multiple preference types. We show in Appendix G.3 that only in a small numbers of responses, this ordering is explicitly flagged as an "assumption", while others default to the rule silently. This assumption also persists even when the lexicographically weaker bundle is much larger (20 items vs. 1).

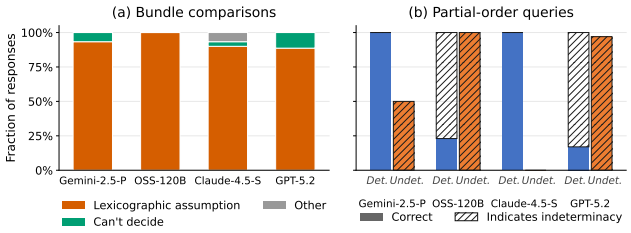


Figure 3: Behavior of LLMs on RS-incomparable bundle comparisons (a), aggregated across different preference types, and partial-order pairwise queries (b).

Partial-order queries elicit a different pattern. To answer whether a is preferred to b, a model must check whether the stated comparisons induce a chain between the two items; if no chain exists in either direction, the pair is incomparable. Figure 3 (b) shows that models split into two failure modes. Claude-4.5-S and Gemini-2.5-Pro commit on undetermined pairs by either rejecting the input as incorrectly specified or using ad-hoc heuristics such as out-degree counts in the DAG induced by the stated comparisons (See Appendix G.4 for details), at the cost of accuracy when no chain exists. OSS-120B and GPT-5.2 do the opposite: they perform an incomplete search and prematurely conclude that two items are incomparable, scoring well on undetermined queries but poorly on determined ones where a chain does exist.

Framing Effects. The free-flow prompt used above does not signal that a query may be undetermined. To test whether the indeterminacy gap reflects a reasoning failure or a framing artefact, we evaluate two prompt variations: *free-flow + "if known"*, where the prompt ends with the qualifier "if known", and *MCQ*, where the model chooses among a set of options that includes "there is not enough information to decide" (prompts are in Appendix J). Figure 2 (b) shows that providing a way to flag indeterminacy improves how often models do so depending on the task, with the largest gains under MCQ. However, we show in Appendix G.4 through the example of partial-order queries, the MCQ format is potentially a double-edged sword. It improves Gemini-2.5-Pro's detection of indeterminacy while biasing Claude-4.5-S into selecting that option even when the query is determined.

Scaling. Beyond the indeterminacy gap, accuracy degrades as the input grows. Figure 6 (Appendix G) shows a clear decline as the number of alternatives or agents moves from 100 to 200. The decline is more pronounced on determined queries: models make essentially no errors on atomic and comparative queries up to size 100, but begin to fail at size 200.

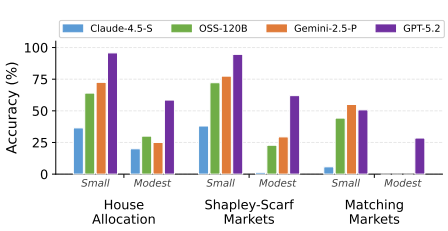
4 Structural Indeterminacy: Algorithmic Reasoning over Preferences

Algorithmic reasoning over preferences extends beyond preference inference, requiring models to (i) interpret structured preference profiles, (ii) execute multi-step procedures, and (iii) satisfy the constraints imposed by a target solution concept. In this setting, a fundamental challenge arises from **structural indeterminacy**: even under fully specified preferences, a solution concept may be infeasible for a given instance. This phenomenon is central to computational social choice, where the interaction between preference domains (e.g., strict versus weak orders) and solution concepts (e.g., core stability or matching stability notions) governs the existence of admissible outcomes.

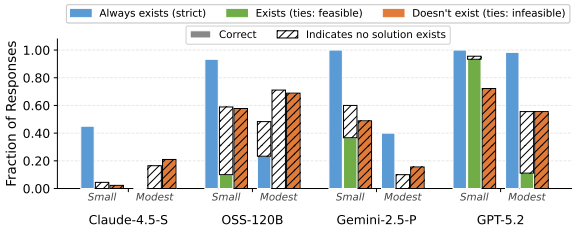
In this section, we systematically evaluate the ability of large language models to reason in such settings. We evaluate how performance scales with market size, whether models identify infeasible instances, and whether an indeterminacy option changes their behavior. We sample instances based on Impartial Culture (Black et al., 1958; Eğecioğlu & Giritligil, 2013) (i.e. uniformly at random), and perform a *rejection sampling* strategy to include an equal number of feasible and infeasible instances (for solution concepts that admit both types). Appendix E.2 has further details on instance creation.

4.1 Scaling, Infeasibility, and Indeterminate Option

Scaling. We begin with a baseline setting in which preferences are given as complete strict linear orders, ensuring that standard solution concepts are guaranteed to exist. We consider a range of



(a) Decrease in accuracy of computing solutions due to an increase in input size, across different domains.



(b) Performance on the generation task by model, problem size, and existence category, pooled across Shapley-Scarf and Matching Markets.

canonical objectives, including Pareto optimality, core outcomes, and stability, as well as social welfare criteria such as egalitarian, utilitarian, and rank-maximal solutions.[4]. As shown in Figure 4a, model performance degrades rapidly as market size increases, even in moderately sized instances. This decline is consistent across solution concepts, indicating limited scalability of LLMs in multi-step algorithmic reasoning tasks. While performance is comparatively stronger for concepts with well-structured canonical algorithms, the overall trend aligns with prior findings on the limitations of LLM reasoning in complex settings (Shojaee et al., 2025; Hosseini et al., 2025). In Appendix H, we demonstrate that models default to canonical algorithms for the domain when the solution concept is unspecified (Appendix H.3), and display similar behavior when tasked with computing weaker solution concepts (Appendix H.2).

Deciding Infeasibility. In Shapley–Scarf and two-sided matching markets, ties in preferences give rise to richer and more demanding solution concepts, namely the strict core, and strong and super stability. Because these concepts impose stricter constraints than the weak core and weak stability, some instances admit no solution that satisfies them, and are therefore infeasible. This allows us to evaluate whether models construct valid solutions when they exist, and whether they identify infeasibility when they do not.

Figure 4b provides an overview of model performance across the two domains. As shown, all models exhibit declining performance both in correctly identifying infeasible instances, where no solution exists, and in generating valid outputs when the corresponding solution concepts are feasible. In small markets ($n = 10$), GPT-5.2 performs well in identifying infeasible instances, likely due to its ability to rely on heuristics and near-exhaustive search (see Appendix K for details). However, even at modest scales (e.g., $n = 30$), its performance deteriorates to the level of other models, as such approaches are no longer computationally feasible.

Indeterminacy Option. Given that certain instances may not admit a solution under specific concepts (e.g., super stability or the strict core), we further examine whether LLMs can correctly reason about such infeasibility when explicitly provided with an abstention option (i.e., declaring non-existence or returning a null solution).

Figure 5 (a) summarizes model accuracy across three solution concepts in both markets, disaggregated by feasible and infeasible instances. While the indeterminacy option improves accuracy in detecting infeasible instances, it introduces a systematic bias toward over-declaring non-existence, even when solutions exist. Consequently, performance on feasible instances deteriorates, as models increasingly abstain rather than produce valid solutions. Appendix K provides an in-depth analysis of the heuristic strategies underlying these behaviors.

4.2 Verification: Selection from a Menu of Options

Thus far, we observe that language models struggle to correctly reason about solution concepts, both in generating feasible outcomes and in identifying infeasibility, even when explicitly provided with an abstention option. This raises the question of whether these failures stem from limitations in algorithmic reasoning (i.e., executing the underlying steps) or from fundamental deficiencies in feasibility reasoning. To separate the two, we follow prior work (e.g., (Hosseini & Khanna, 2025)) in

[4]See Appendix D.1, for detailed formalism of each solution concept and Appendix H (Figure 8) for performance on each.

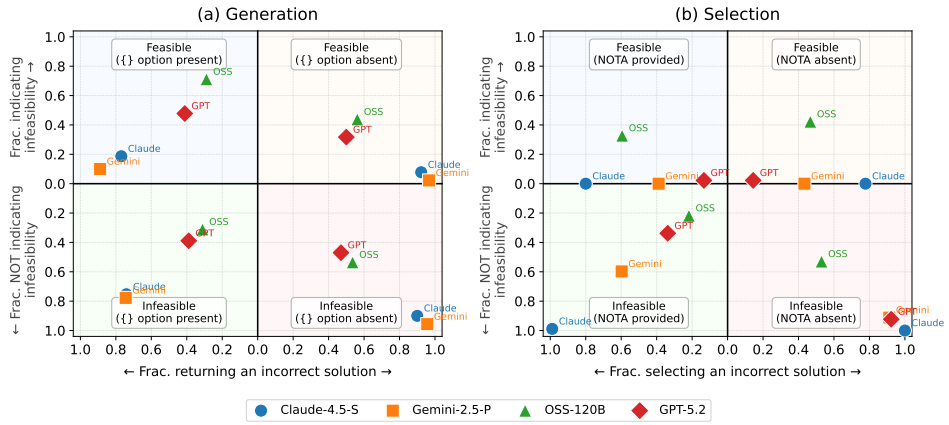


Figure 5: Errors made by models on generation (a) and selection (b) tasks as a function of problem feasibility and availability of an indeterminacy option, averaged over different solution concepts (Modest instance size). An ideal model would cluster near the origin in all four quadrants.

distinguishing between generation and selection tasks. Selection tasks are easier, as they require only *verification* of candidate solutions rather than the synthesis of a solution via multi-step procedures.

We consider the same market settings with weak preferences. For each instance, the model is presented with a set of pre-computed candidate solutions (five to seven in number) and is required to select a valid one. Instances are generated via rejection sampling (Appendix E.2), and each candidate is either the solution computed for a distinct solution concept (with some options satisfying no property). In certain variants, a "none of the above" (NOTA) option is included to permit abstention; in others, it is omitted to enforce selection.

Figure 5 (b) shows that models frequently select incorrect candidates even when a valid solution is present. For infeasible instances, the removal of the NOTA option significantly increases the likelihood with which they select an incorrect option instead of indicating that no option applies. Moreover, while GPT-5.2 substantially outperforms other models when a valid solution is present, its performance degrades markedly on infeasible instances, particularly as problem size increases or when the NOTA option is absent.

Intention-Action Misalignment. Given these failures in selection tasks, a key question is what objective LLMs are implicitly optimizing when selecting among candidate solutions. To investigate this, we construct instances across different preference types—strict complete, complete with ties, and incomplete with ties—in each domain, such that the candidate options each satisfy distinct (often non-overlapping) solution concepts. Our objective is to characterize the implicit criteria guiding model behavior and to assess whether these criteria are consistently realized in the selected outcomes.

Figure 13 (Appendix H.5) shows the fraction of responses in which models select solutions satisfying various properties. To ensure comprehensive coverage, the candidate set includes outcomes aligned with standard welfarist criteria, such as utilitarian and egalitarian objectives. We further infer model intentions using LLM-based judges, enabling a comparison between the property a model appears to target and the property actually satisfied by its selected solution.[5] This analysis reveals a systematic intention-action misalignment: models frequently imply adherence to a particular solution concept, yet select outcomes that do not match the criteria they appear to optimize. This intention-action gap is especially pronounced in settings with ties or incomplete preferences, where structural constraints may render certain solution concepts infeasible.

5 Performance Improvements with Assisted Reasoning

We consider assisted reasoning under two settings: (i) refinement via feedback and (ii) reasoning with code execution, to examine whether the models recover their performance gap on indeterminacy.

[5]We analyze reasoning traces to identify the "intended" concepts using LLM-as-a-judge (see Appendix O).

Both are presented below. In Appendix L, we demonstrate how prompt-level mitigation such as providing few-shot examples or explicit instructions to not make assumption (both general and problem specific), fail to reliably calibrate the models.

Refinements with Feedback. To test whether models fail because they cannot recognize an incorrect solution or because they cannot repair it, we evaluate a multi-round setup in which each response receives structured feedback identifying the violation (a blocking pair in matching markets, a blocking coalition in Shapley–Scarf markets), followed by up to two retries. We run this with OSS-120B and GPT-5.2, each at the smallest size at which it fails in the one-shot setting (Small and Modest, respectively; Appendix N). Feedback improves performance substantially on Shapley–Scarf markets but barely at all on matching markets (Figure 14 in Appendix N). On Shapley–Scarf instances the models typically return a valid but incorrect allocation, which the feedback then corrects in many cases. On matching instances both models declare infeasibility from the first round, and once a model returns no solution there is nothing for the feedback to act on.

Reasoning with Code Execution. In the previous sections we observed how LLMs fail to recognize when questions are undetermined, and are miscalibrated in identifying when a given solution concept is achievable for a given input. To understand whether these limitations are present only when LLMs reason in-context, we evaluate their performance when allowed to write code to solve the same problems, given their impressive ability to write code (Jimenez et al., 2024; Jain et al., 2024).

For preference reasoning, solving determined questions with code execution completely eliminates errors related to scale. However, on undetermined queries such as (RS-incomparable) bundle comparisons and the ranking task, the answers are *always* incorrect. The assumptions models make now become part of the implemented algorithmic steps. The contrast with the structural setting is informative. There, models often write an explicit feasibility check and return no solution when the search finds none (Appendix K), but on undetermined preference queries they write no such check.

For algorithmic reasoning, allowing LLMs to write code to solve both feasible and infeasible problems significantly improves their performance, although not by implementing the "correct" approach (see Table 29 in Appendix M for exact accuracy levels). In fact, their approach changes with the input size. For Small instances, the models resort to brute-force enumeration of all solutions. The strategy changes for Modest and for Medium instances, where they could use slightly more efficient heuristics to solve the problem (See Appendix M for detailed descriptions of the approaches used).

However, as becomes clear with Medium instances, these approaches are not scalable. For Shapley-Scarf Markets, the models perform well on feasible instances, but often time-out on infeasible ones[6]. Similarly, the strategy used for Matching Markets misidentifies feasible instances as infeasible, in the few responses that do not time-out. Hence, while code improves performance on smaller markets, it does not enable models to recognize and use approaches relevant for deployment-level sizes.

Dissociating the two axes. The two forms of indeterminacy respond to different interventions, which is what separates them from a single shared calibration failure. Code execution improves structural reasoning but not epistemic queries, where the model compiles its assumption into the program it writes. Prompt-level intervention does the reverse. It raises indeterminacy detection on preference queries (Figure 2(b)) but does not improve feasibility reasoning due to structural (in)determinacy. This suggest that the escape hatch instead moves the error into over-abstention (Figure 5(a)).

6 Concluding Remarks

We argue that indeterminacy is a fundamental dimension of reasoning in preference-based AI systems. As language models increasingly mediate alignment, coordination, and collective decisions, current closed-world training paradigms systematically bias models toward hallucinating determinacy, feasibility, and preference information where none is entailed—a broader manifestation of *verisimilitude* in AI, where outputs appear plausible despite lacking logical grounding. Our results suggest that robust AI will require open-world reasoning frameworks (e.g. multi-valued semantics), richer notions of non-entailment and impossibility, and benchmarks that evaluate not only correctness, but the ability to recognize when no justified answer exists.

[6]If the model does not return a response in 20 minutes, we treat it as a time-out.

AI USE STATEMENT

All research questions, experimental designs, and methodology in this work are our own. We used generative AI tools for implementing methods, limited to writing and debugging the code for benchmark generation, inference, analysis, and plotting, and for supporting qualitative analysis and interpretation of results, namely synthesizing findings from the reasoning traces analyzed in Appendix K and Appendix O. We have not used generative AI tools for designing the methodology or experiments, generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims or proofs, proposing or refining hypotheses, or translation. Additionally, we used generative AI tools for editing the paper to improve readability. We have reviewed all AI-assisted work: the main correctness checkers were validated against an independent brute-force solver, and the LLM judges against two additional judges and a human annotator (Appendix O.3). We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We do not foresee ethical risks arising from this work. The study involves no human subjects and no personal data. All preference profiles are synthetic and sampled under Impartial Culture, and all models are queried through their public interfaces. We discuss the implications of our findings for deployed systems that reason over preferences in Appendix B.

REPRODUCIBILITY STATEMENT

All details needed to reproduce our results are provided in the paper. Appendix E.1 and Appendix E.2 describe how instances are generated, Appendix J gives the exact prompt templates, Appendix F lists the models and inference settings, Appendix O describes the judge setup, and Appendix I describes the statistical analysis. The exact replication procedure, including instance generation, model querying, and scoring, is outlined in the code provided with the submission.

REFERENCES

Yasin Abbasi-Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvari. To believe or not to believe your LLM: Iterative prompting for estimating epistemic uncertainty. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=k6iyUfwdI9.

Atila Abdulkadiroğlu and Tayfun Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260, 1999.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A. Voudouris. Fair division of indivisible goods: A survey. In Luc De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 5385–5393. ijcai.org, 2022. doi: 10.24963/IJCAI.2022/756. URL https://doi.org/10.24963/ijcai.2022/756.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Anthropic. Introducing claude sonnet 4.5, Sep 2025. URL https://www.anthropic.com/news/claude-sonnet-4-5.

David Eric Austin, Anton Korikov, Armin Toroghi, and Scott Sanner. Bayesian optimization with llm-based acquisition functions for natural language preference elicitation. In Tommaso Di Noia, Pasquale Lops, Thorsten Joachims, Katrien Verbert, Pablo Castells, Zhenhua Dong, and Ben London (eds.), *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024, Bari, Italy, October 14-18, 2024*, pp. 74–83. ACM, 2024. doi: 10.1145/3640457.3688142. URL https://doi.org/10.1145/3640457.3688142.

Haris Aziz, Péter Biró, and Makoto Yokoo. Matching market design with constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):12308–12316, Jun. 2022. doi: 10.1609/aaai.v36i11.21495. URL https://ojs.aaai.org/index.php/AAAI/article/view/21495.

Haris Aziz, Xin Huang, Nicholas Mattei, and Erel Segal-Halevi. Computing welfare-maximizing fair allocations of indivisible goods. *European Journal of Operational Research*, 307(2):773–784, 2023. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2022.10.013. URL https://www.sciencedirect.com/science/article/pii/S0377221722007822.

Haris Aziz, Isaiah Iliffe, Bo Li, Angus Ritossa, Ankang Sun, and Mashbat Suzuki. Envy-free house allocation under uncertain preferences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(9):9477–9484, Mar. 2024. doi: 10.1609/aaai.v38i9.28802. URL https://ojs.aaai.org/index.php/AAAI/article/view/28802.

Sree Bhattacharyya, Samarth Khanna, Leona Chen, Lucas Craig, Tharun Dilliraj, and James Z. Wang. Beyond confidence: Rethinking self-assessments for performance prediction in llms, 2026. URL https://arxiv.org/abs/2605.07806.

Duncan Black et al. *The theory of committees and elections*. Springer, 1958.

Graham Brightwell. Models of random partial orders. *Surveys in combinatorics*, 5383, 1993.

Jianhui Chen, Yuzhang Luo, and Liangming Pan. Mechanistic data attribution: Tracing the training origins of interpretable LLM units. *CoRR*, abs/2601.21996, 2026. doi: 10.48550/ARXIV.2601.21996. URL https://doi.org/10.48550/arXiv.2601.21996.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Baker Grosse, and Eric P. Xing. What is your data worth to GPT? LLM-scale data valuation with influence functions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=zPKeJAEo27.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H Holliday, Bob M Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailey Schoelkopf, et al. Social choice should guide AI alignment in dealing with diverse human feedback. *arXiv preprint arXiv:2404.10271*, 2024.

Benjamin Cookson, Soroush Ebadian, and Nisarg Shah. Fairness perceptions of large language models. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 35393–35401. AAAI Press, 2026. doi: 10.1609/AAAI.V40I42.40848. URL https://doi.org/10.1609/aaai.v40i42.40848.

Google Deepmind. Gemini pro, Mar 2025. URL https://deepmind.google/technologies/gemini/pro/.

John P Dickerson, Hadi Hosseini, Samarth Khanna, and Leona Pierce. Who gets the kidney? human-ai alignment, indecision, and moral values. In *Proceedings of the 2026 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '26, pp. 8128–8147, New York, NY, USA, 2026.

Association for Computing Machinery. ISBN 9798400725968. doi: 10.1145/3805689.3806437. URL https://doi.org/10.1145/3805689.3806437.

F. Echenique, N. Immorlica, V.V. Vazirani, and A.E. Roth. *Online and Matching-Based Market Design*. Cambridge University Press, 2023. ISBN 9781108831994. URL https://books.google.com/books?id=1ea-EAAAQBAJ.

Ömer Eğecioğlu and Ayça E Giritligil. The impartial, anonymous, and neutral culture model: a probability model for sampling public preference structures. *The Journal of Mathematical Sociology*, 37(4):203–222, 2013.

P Erdős and A Rényi. On random graphs i. *Publ. math. debrecen*, 6(290-297):18, 1959.

Sara Fish, Paul Gölz, David Parkes, Ariel Procaccia, Gili Rusak, Itai Shapira, and Manuel Wuthrich. Generative social choice. *Journal of the ACM*, 2023.

Sara Fish, Julia Shephard, Minkai Li, Ran I Shorrer, and Yannai A Gonczarowski. Econevals: Benchmarks and litmus tests for economic decision-making by llm agents. *arXiv preprint arXiv:2503.18825*, 2025.

David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

Kunal Handa, Yarin Gal, Ellie Pavlick, Noah D. Goodman, Jacob Andreas, Alex Tamkin, and Belinda Z. Li. Bayesian preference elicitation with language models. *CoRR*, abs/2403.05534, 2024. doi: 10.48550/ARXIV.2403.05534. URL https://doi.org/10.48550/arXiv.2403.05534.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Hadi Hosseini and Samarth Khanna. Distributive fairness in large language models: Evaluating alignment with human values. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=5pQFE4yIZ5.

Hadi Hosseini, Medha Kumar, and Sanjukta Roy. The degree of fairness in efficient house allocation. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz (eds.), *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, Frontiers in Artificial Intelligence and Applications, pp. 3636–3643. IOS Press, 2024. doi: 10.3233/FAIA240920. URL https://doi.org/10.3233/FAIA240920.

Hadi Hosseini, Samarth Khanna, and Ronak Singh. Matching markets meet LLMs: Algorithmic reasoning with ranked preferences. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1zKElu2MuQ.

Wenyue Hua, Ollie Liu, Lingyao Li, Alfonso Amayuelas, Julie Chen, Lucas Jiang, Mingyu Jin, Lizhou Fan, Fei Sun, William Wang, Xintong Wang, and Yongfeng Zhang. Game-theoretic LLM: agent workflow for negotiation games. *CoRR*, abs/2411.05990, 2024. doi: 10.48550/ARXIV.2411.05990. URL https://doi.org/10.48550/arXiv.2411.05990.

David Huang, Francisco J. Marmolejo Cossío, Edwin Lock, and David C. Parkes. Accelerated preference elicitation with llm-based proxies. *CoRR*, abs/2501.14625, 2025. doi: 10.48550/ARXIV.2501.14625. URL https://doi.org/10.48550/arXiv.2501.14625.

Robert W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48(3):261–272, 1994. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(92)00179-P. URL https://www.sciencedirect.com/science/article/pii/0166218X9200179P.

Robert W. Irving, Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E. Paluch. Rank-maximal matchings. *ACM Trans. Algorithms*, 2(4):602–610, October 2006. ISSN 1549-6325. doi: 10.1145/1198513.1198520. URL https://doi.org/10.1145/1198513.1198520.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Minje Jang, Sunghyun Kim, Changho Suh, and Sewoong Oh. Top-k ranking from pairwise comparisons: When spectral ranking is optimal. *CoRR*, abs/1603.04153, 2016. URL http://arxiv.org/abs/1603.04153.

Jingru Jia, Zehua Yuan, Junhao Pan, Paul McNamara, and Deming Chen. Large language model strategic reasoning evaluation through behavioral game theory. *CoRR*, abs/2502.20432, 2025. doi: 10.48550/ARXIV.2502.20432. URL https://doi.org/10.48550/arXiv.2502.20432.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine M. Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew Gordon Wilson. Large language models must be taught to know what they don't know. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=QzvWyggrYB.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel Bell. Abstentionbench: Reasoning LLMs fail on unanswerable questions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=OkHC30LLpO.

SC Kleene. *Introduction to metamathematics*. North-Holland, 1952.

Belinda Z. Li, Alex Tamkin, Noah D. Goodman, and Jacob Andreas. Eliciting human preferences with language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=LvDwwAgMEW.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Shengxin Liu, Xinhang Lu, Mashbat Suzuki, and Toby Walsh. Mixed fair division: A survey. *J. Artif. Intell. Res.*, 80:1373–1406, 2024. doi: 10.1613/JAIR.1.15800. URL https://doi.org/10.1613/jair.1.15800.

Xinghua Long and Rodrigo A Velez. Balanced house allocation. *arXiv preprint arXiv:2109.01992*, 2021.

Jinpeng Ma. Strategy-proofness and the strict core in a market with indivisibilities. *Int. J. Game Theory*, 23(1):75–83, March 1994. ISSN 0020-7276. doi: 10.1007/BF01242849. URL https://doi.org/10.1007/BF01242849.

Ali Montazeralghaem, Guy Tennenholtz, Craig Boutilier, and Ofer Meshi. Asking clarifying questions for preference elicitation with large language models. *CoRR*, abs/2510.12015, 2025. doi: 10.48550/ARXIV.2510.12015. URL https://doi.org/10.48550/arXiv.2510.12015.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Thomas Quint and Jun Wako. On houseswapping, the strict core, segmentation, and linear programming. *Mathematics of Operations Research*, 29(4):861–877, 2004. doi: 10.1287/moor.1040.0106. URL https://doi.org/10.1287/moor.1040.0106.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

Manuel Ríos, Rubén Francisco Manrique, Nicanor Quijano, and Luis Felipe Giraldo. The illusion of rationality: Tacit bias and strategic dominance in frontier LLM negotiation games. *CoRR*, abs/2512.09254, 2025. doi: 10.48550/ARXIV.2512.09254. URL https://doi.org/10.48550/arXiv.2512.09254.

Kang Rong, Qianfeng Tang, and Yongchao Zhang. The core of school choice problems. *Economic Theory*, 77(3):783–800, 2024.

Alvin E Roth. The economics of matching: Stability and incentives. *Mathematics of operations research*, 7(4):617–628, 1982.

Alvin E. Roth and Andrew Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, 1977. ISSN 0304-4068. doi: https://doi.org/10.1016/0304-4068(77)90004-0. URL https://www.sciencedirect.com/science/article/pii/0304406877900040.

Ildikó Schlotter and Lydia Mirabel Mendoza-Cadena. The strong core of housing markets with partial order preferences. In Sanmay Das, Ann Nowé, and Yevgeniy Vorobeychik (eds.), *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, pp. 1867–1875. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025. doi: 10.5555/3709347.3743823. URL https://dl.acm.org/doi/10.5555/3709347.3743823.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*, 2023.

AMARTYA SEN. *Collective Choice and Social Welfare: An Expanded Edition*. Harvard University Press, 2017. ISBN 9780674971608. URL http://www.jstor.org/stable/j.ctv2sp3dqx.

Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1 (1):23–37, 1974.

Zhengliang Shi, Ruotian Ma, Jen-tse Huang, Xinbei Ma, Xingyu Chen, Mengru Wang, Qu Yang, Yue Wang, Fanghua Ye, Ziyang Chen, Shanyi Wang, Cixing Li, Wenxuan Wang, Zhaopeng Tu, Xiaolong Li, Zhaochun Ren, and Linus. Social welfare function leaderboard: When LLM agents allocate social welfare. *CoRR*, abs/2510.01164, 2025. doi: 10.48550/ARXIV.2510.01164. URL https://doi.org/10.48550/arXiv.2510.01164.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2506.06941.

Adi Simhi, Itay Itzhak, Fazl Barez, Gabriel Stanovsky, and Yonatan Belinkov. Trust me, i'm wrong: Llms hallucinate with certainty despite knowing the answer. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 14665–14688. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.792/.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Lars-Gunnar Svensson. Strategy-proof allocation of indivisible goods. *Social Choice and Welfare*, 16 (4):557–567, 1999. ISSN 01761714, 1432217X. URL http://www.jstor.org/stable/41106323.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Michael Henry Tessler, Michiel A. Bakker, Daniel Jarrett, Hannah Sheahan, Martin J. Chadwick, Raphael Koster, Georgina Evans, Lucy Campbell-Gillingham, Tantum Collins, David C. Parkes, Matthew Botvinick, and Christopher Summerfield. Ai can help humans find common ground in democratic deliberation. *Science*, 386(6719):eadq2852, 2024. doi: 10.1126/science.adq2852. URL https://www.science.org/doi/abs/10.1126/science.adq2852.

Nguyen Thach, Xingchen Sha, and Hau Chan. Large language models for designing participatory budgeting rules. *CoRR*, abs/2602.09349, 2026. doi: 10.48550/ARXIV.2602.09349. URL https://doi.org/10.48550/arXiv.2602.09349.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, 2023.

Caroline Wang, Daniel Kasenberg, Kim Stachenfeld, and Pablo Samuel Castro. Discovering differences in strategic behavior between humans and llms. *CoRR*, abs/2602.10324, 2026. doi: 10.48550/ARXIV.2602.10324. URL https://doi.org/10.48550/arXiv.2602.10324.

Khurram Yamin, Jingjing Tang, Santiago Cortes-Gomez, Amit Sharma, Eric Horvitz, and Bryan Wilder. Do llms act like rational agents? measuring belief coherence in probabilistic decision making. *CoRR*, abs/2602.06286, 2026. doi: 10.48550/ARXIV.2602.06286. URL https://doi.org/10.48550/arXiv.2602.06286.

Joshua C. Yang, Damian Dailisan, Marcin Korecki, Carina I. Hausladen, and Dirk Helbing. LLM voting: Human choices and AI collective decision-making. In Sanmay Das, Brian Patrick Green, Kush Varshney, Marianna Ganapini, and Andrea Renda (eds.), *Proceedings of the Seventh AAAI/ACM Conference on AI, Ethics, and Society (AIES-24) - Full Archival Papers, October 21-23, 2024, San Jose, California, USA - Volume 1*, pp. 1696–1708. AAAI Press, 2024. doi: 10.1609/AIES.V7I1.31758. URL https://doi.org/10.1609/aies.v7i1.31758.

Gal Yona, Roee Aharoni, and Mor Geva. Can large language models faithfully express their intrinsic uncertainty in words? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7752–7764, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.443. URL https://aclanthology.org/2024.emnlp-main.443/.

Oğuzhan Çelebi and Joel P Flynn. Priority design in centralized matching markets. *The Review of Economic Studies*, 89(3):1245–1277, 05 2022. ISSN 0034-6527. doi: 10.1093/restud/rdab053. URL https://doi.org/10.1093/restud/rdab053.

Technical Appendices and Supplementary Material

A Limitations and future directions

Our evaluation is bounded in three ways that are worth stating. First, we test four models, spanning two closed families and one open-weights family. They are the current frontier, but the panel is small, so we report per-model results throughout rather than a single pooled number. Second, we sample instances under Impartial Culture, which lets us build matched determined and undetermined instances on the same profile and control feasibility exactly. The sampler decides which instances turn out feasible, not the reasoning a task demands, so we do not expect structured distributions such as Mallows or single-peaked preferences to change the picture, though we have not tested this. Third, two of our analyses infer what a model intended from its reasoning trace, and so depend on a judge model. Appendix O.3 reports that judge's agreement with two additional judges and with a human annotator, and flags the one cell where they disagree. Every other result is scored by deterministic checkers, which we validate against a brute-force solver in the same appendix.

These bounds aside, our evaluation establishes what current reasoning models do when confronted with indeterminate preference queries, and leaves open why they do it. The systematic assumptions we identify (e.g. lexicographic tie-breaking on bundles, premature termination on partial-order chains) are robust empirical patterns, but their underlying mechanism is opaque to the input-output methodology used here. Recent advances in mechanistic interpretability (Lindsey et al., 2025; Ameisen et al., 2025; Templeton et al., 2024) suggest that these failures could be traced to specific circuits or features within the model, illuminating whether models silently override their own indeterminacy detection or fail to recognize it altogether.

The role of prompt framing is likewise only partly mapped. We vary it along three coarse axes (free-flow, free-flow + "if known", MCQ) but do not characterize the threshold at which a prompt becomes informative enough to elicit reliable abstention. Finer-grained probing along the lines of Sclar et al. (2023) and Kirichenko et al. (2025) could map this transition more precisely.

A final question is which aspects of training data and post-training procedures determine a model's calibration to indeterminacy. Recent advances in scalable data attribution (Choe et al., 2026) and frameworks that bridge interpretability and data influence (Chen et al., 2026), combined with analyses of calibration shifts during alignment (Kadavath et al., 2022; Tian et al., 2023), could distinguish whether targeted fine-tuning would address the underlying calibration issue or merely mask the surface symptoms.

B Broader Impacts

The findings in this paper bear directly on two deployment contexts where LLMs are increasingly used to reason over preferences. First, agentic systems acting on behalf of users — making purchases, scheduling commitments, mediating between conflicting goals — must distinguish between situations in which the user's stated preferences determine an action and situations in which they do not. Our results show that current frontier models systematically fail at this distinction, which means an agentic system built on these models may, by default, take actions that the user has not actually authorized, while presenting them as reflecting the user's preferences. Second, platforms that aggregate preferences for collective decision-making — participatory budgeting tools, deliberation platforms, large-scale opinion-mapping systems — depend on the assumption that the aggregation faithfully reflects the inputs. When the input is structurally indeterminate (incomplete preferences, infeasibility under ties, etc.), a model that confidently produces an aggregate is generating an artificial consensus that does not exist in the data. Our work surfaces these failure modes in a controlled evaluation, with the goal of making them measurable and addressable before such systems are deployed in higher-stakes settings.

C Extended Related Work

LLM Reasoning Benchmarks. Existing benchmarks evaluate LLMs on mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021), scientific reasoning (Phan et al., 2025; Rein et al.,

2023), and code generation (Jain et al., 2024; Jimenez et al., 2024). These assess whether a model reaches the correct answer, but provide limited insights into the ability of LLMs to execute algorithms correctly. Two recent papers show that LLMs fail to execute well-known algorithms as instance size grows: Hosseini et al. (2025) trace this collapse in the commonly used Deferred Acceptance algorithm to compounding step-level errors, and Shojaee et al. (2025) find the same pattern on combinatorial puzzles (e.g. the missionaries and cannibals problem, tower of Hanoi, etc.). Both papers characterize algorithmic failure without distinguishing among algorithms — that is, they do not ask whether some algorithms are harder than others or why. We build on this by asking a finer question: do LLMs fail uniformly across algorithms, or is performance related to properties of the algorithm itself? Our findings suggest the latter, with training-data familiarity playing a larger role than computational complexity.

Rationality and Strategic Reasoning. A complementary literature asks whether LLMs behave as rational agents in strategic settings. Recent reasoning models come closer to equilibrium play than earlier ones (Jia et al., 2025) and identify strategic patterns faster than humans (Wang et al., 2026), yet LLMs still exhibit anchoring effects and model-specific biases that prevent convergence to optimal solutions (Ríos et al., 2025), and their stated probabilities over outcomes frequently violate basic Bayesian consistency conditions (Yamin et al., 2026). These papers evaluate LLMs as strategic actors optimizing over outcomes. Our focus is orthogonal: we evaluate LLMs as procedure executors applying a specified algorithm to a given input. This distinction is consequential because optimal strategy computation in many settings itself requires running sophisticated algorithms (for instance, computing the core of a cooperative game or finding a stable matching) so failures in procedure execution place a ceiling on strategic rationality as well (Nisan et al., 2007). Hua et al. (2024) show that LLMs can achieve Pareto-optimal and envy-free outcomes in negotiation settings; our results suggest this may reflect approximate pattern-matching rather than principled computation, since the same notions are computed unreliably when formally defined and applied to structured inputs.

LLMs for Social Choice. Computational social choice studies how individual preferences can be aggregated into collective decisions, with active subliteratures on house allocation (Abdulkadiroğlu & Sönmez, 1999; Hosseini et al., 2024; Schlotter & Mendoza-Cadena, 2025; Aziz et al., 2024), resource allocation (Amanatidis et al., 2022; Liu et al., 2024), and matching markets (Echenique et al., 2023; Çelebi & Flynn, 2022; Rong et al., 2024; Aziz et al., 2022). As LLMs are increasingly proposed as components of decision-making pipelines in these domains, recent work has begun to ask both whether LLMs can compute the outcomes that social-choice theory prescribes, and whether their default behaviour aligns with how humans think about fairness.

On the fairness alignment side, Hosseini & Khanna (2025) and Cookson et al. (2026) both find that when LLMs are asked to allocate resources fairly, their choices differ from human judgments in the same way: they tend to maximise overall welfare rather than spread benefits equally, and this pattern is robust to changes in how the prompt is phrased. Hosseini & Khanna (2025) additionally find that what the LLM says it will do does not always match the allocation it actually produces. We recover both findings on a broader range of solution concepts and preference structures (Section H.5), suggesting that they extend beyond resource allocation to allocation and matching tasks more generally. On the solving side, Hosseini et al. (2025) evaluate LLMs as one-shot solvers on stable-matching instances, while Fish et al. (2025) embed an LLM in an iterative loop, giving it feedback in the form of blocking pairs and measuring how its solution improves over rounds. Both papers analyse the kinds of failures the model makes; we contribute to this line of work by studying how LLMs behave when a question is underspecified or when no valid solution exists. Other recent work has applied LLMs to voting (Yang et al., 2024), where the choice an LLM makes among candidates depends on the order in which the options are listed and on which voting rule is used, and to participatory budgeting (Thach et al., 2026), where LLMs are used to design heuristics for selecting which projects to fund.

LLMs for Preference Elicitation. LLMs are increasingly deployed as proxies for human preferences in elicitation pipelines, using techniques ranging from Bayesian query selection (Handa et al., 2024; Austin et al., 2024) to LLM-simulated preference profiles (Huang et al., 2025; Montazeralghaem et al., 2025) and direct comparative elicitation (Li et al., 2025). These systems implicitly assume that LLMs can reason reliably about structured preferences. Our benchmark directly tests this assumption and finds it to be optimistic, particularly under incomplete information and multi-agent aggregation.

Abstention and Uncertainty. Kirichenko et al. (2025) show that abstention, i.e. the ability to refrain from answering when unsure, is unsolved across frontier models and that reasoning fine-tuning can

degrade it. Systematic miscalibration between models' intrinsic uncertainty and how they express it is well documented (Yona et al., 2024; Kapoor et al., 2024), and models can hallucinate with high confidence even on questions they have the capacity to answer correctly (Simhi et al., 2025; Abbasi-Yadkori et al., 2024; Bhattacharyya et al., 2026). A related failure appears in moral decision-making, where models do not reproduce the indecision humans express on hard choices, even though that indecision signals a need for further deliberation (Dickerson et al., 2026). Our results contribute a more structured instance of this failure: when preference queries are formally underdetermined, models do not abstain but instead make implicit, predictable assumptions, such as lexicographic tie-breaking or alphabetical completion of partial lists. Crucially, providing an explicit uncertainty option eliminates this behavior in some settings but not others, suggesting the failure is not purely one of output formatting but of recognizing underdetermination itself.

D Extended Preliminaries

Table 2: Preference expressivity, axioms, and properties. In each case, we consider complete orders and incomplete orders where the preference is truncated (missing the tail).

Preference Expressivity	**Description**	**Determining Property**
Complete Strict Linear Order	Asymmetric, transitive, and total.	Always determined.
Complete Order with Ties	Allows ties ($a \sim b$); total.	Determined (indifference is explicit).
Incomplete Order with Ties	Allows ties and missing pairs.	Undetermined if incomparable ($a \parallel b$).
Partial Order / Pairwise Sets	Transitive and asymmetric; missing pairs.	Determined only if in the transitive closure.

D.1 Solution Concepts

We define the solution concepts evaluated in this work, grouped by the domain and preference structure in which they apply.

Pareto optimality. An outcome μ is *Pareto-optimal* (PO) if there exists no feasible outcome μ' such that $\mu'(i) \succeq_i \mu(i)$ for all $i \in N$ and $\mu'(j) \succ_j \mu(j)$ for some $j \in N$. Under incomplete preferences, we additionally consider *maximum-cardinality Pareto-optimal* (MCPO) allocations: among all allocations that match the maximum number of agents to ranked alternatives, those that are Pareto-optimal.

Core (Shapley–Scarf housing market). The core and TTC are notions of the endowment economy, so we state them for the Shapley–Scarf market rather than for house allocation without endowments. Let $S \subseteq N$ be a coalition. Under **strict preferences**, S *blocks* an allocation μ if there exists a reassignment μ_S of $\{e(i) : i \in S\}$ among members of S such that $\mu_S(i) \succ_i \mu(i)$ for all $i \in S$. An allocation is in the *core* if no coalition blocks it. The core always exists under strict preferences and contains the output of Top Trading Cycles (TTC) (Shapley & Scarf, 1974; Roth & Postlewaite, 1977); it can also contain other allocations.

Under **preferences with ties**, the definition of blocking depends on how ties are treated, giving rise to two notions:

- **Weak core**: S blocks μ if there exists a reassignment μ_S such that $\mu_S(i) \succ_i \mu(i)$ for all $i \in S$ (every member strictly improves). An allocation is in the weak core if no such coalition exists. The weak core always exists and contains the TTC output under any consistent tie-breaking rule.
- **Strict core**: S blocks μ if there exists a reassignment μ_S such that $\mu_S(i) \succeq_i \mu(i)$ for all $i \in S$ and $\mu_S(j) \succ_j \mu(j)$ for some $j \in S$ (every member weakly improves, at least one strictly improves). An allocation is in the strict core if no such coalition exists. The strict core may be empty, but can be computed in polynomial time if it exists (Quint & Wako, 2004).

Under strict preferences the blocking condition given to the models (every member of the coalition strictly improves) is that of the weak core, and we refer to it as the *core* in that setting. The two notions do not coincide under strict preferences: the strict core is a singleton, namely the TTC allocation (Roth & Postlewaite, 1977), whereas the weak core can contain further allocations. Our checker accepts any allocation that satisfies the blocking condition given in the prompt.

Welfare objectives. For an allocation or matching μ:

$$\mathrm{UW}(\mu) = \sum_{i \in N} -1 * \mathrm{rank}_i(\mu(i)), \qquad \text{(Utilitarian Welfare, maximized)}$$

$$\mathrm{EW}(\mu) = \min_{i \in N} -\mathrm{rank}_i(\mu(i)). \qquad \text{(Egalitarian Welfare, maximized)}$$

A *utilitarian welfare-maximizing* (UW) outcome maximizes $\mathrm{UW}(\mu)$ (Aziz et al., 2023); an *egalitarian welfare-maximizing* (EW) outcome maximizes $\mathrm{EW}(\mu)$ (SEN, 2017).

Rank-maximality. For an outcome μ, define its *rank vector* $r(\mu) = (r_1, r_2, \ldots, r_m)$, where r_k is the number of agents assigned an alternative of rank k. An outcome is *rank-maximal* (RM) if its rank vector is lexicographically maximal over all feasible outcomes (Irving et al., 2006).

Example 1 (Rank vector)**.** *With three agents assigned to rank-1 alternatives, none to rank-2, and one to rank-3, the rank vector is* $(3, 0, 1)$*. Rank-maximality prefers* $(3, 0, 1)$ *to* $(2, 2, 0)$*: as many rank-1 assignments as possible first, then as many rank-2, and so on.*

Stability (two-sided matching). Let (m, w) be an unmatched pair. Under **strict preferences**, (m, w) *blocks* a matching μ if $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$, i.e. both strictly prefer each other to their current partners. A matching is *stable* if no blocking pair exists. Stable matchings always exist and can be computed by Deferred Acceptance (DA) (Gale & Shapley, 1962).

Under **preferences with ties**, the meaning of "blocking" is no longer unique, yielding three stability concepts of increasing strength:

- **Weak stability**: (m, w) blocks μ if both strictly prefer each other to their assigned partners ($w \succ_m \mu(m)$ and $m \succ_w \mu(w)$). This coincides with the strict-preference definition; weakly stable matchings always exist.
- **Strong stability**: (m, w) blocks μ if both weakly prefer each other to their partners and at least one strictly prefers ($w \succeq_m \mu(m)$, $m \succeq_w \mu(w)$, and at least one inequality is strict). Strongly stable matchings may not exist, but can be found in polynomial time when they do (Irving, 1994).
- **Super stability**: (m, w) blocks μ if both weakly prefer each other to their partners ($w \succeq_m \mu(m)$ and $m \succeq_w \mu(w)$). Super stable matchings may not exist, but can be found in polynomial time when they do (Irving, 1994).

The three notions are nested: every super stable matching is strongly stable, and every strongly stable matching is weakly stable. The converse does not hold in general. Note that, the definitions of a stable matching when preferences are strict, and that of a super stable matching when preferences have ties, are equivalent (similar to the relationship between strict core and core in the house allocation setting).

Canonical algorithms. Each domain admits a canonical algorithm, i.e. one that is not only computationally efficient but is uniquely characterized by a combination of desirable axiomatic properties, making it the natural first choice for practitioners and the standard reference in the literature.

- **Top Trading Cycles (TTC)** for the Shapley–Scarf housing market computes the unique allocation in the core (Shapley & Scarf, 1974) and is the only mechanism that is individually rational, Pareto-optimal, and strategy-proof (Ma, 1994; Roth, 1982).
- **Serial Dictatorship (SD)** for object allocation is the unique mechanism that is Pareto-optimal, strategy-proof, and non-bossy (Svensson, 1999).
- **Deferred Acceptance (DA)** for stable matching computes a stable matching that is weakly preferred by the proposing side to all other stable matchings, and is the unique stable and strategy-proof mechanism (for that side) (Gale & Shapley, 1962; Roth, 1982).

E Dataset Creation Details

E.1 Preference Reasoning Tasks

E.1.1 Instance Generation

Strict and complete preferences. Preferences are sampled under the impartial culture model: each agent's ranking is drawn uniformly at random from all permutations of A. Profile sizes range from 30 to 200 items; results are reported at sizes 50, 100, and 200.

Strict and incomplete preferences. Starting from a strict and complete preference list, every agent's ranking in a profile is truncated to the same length, drawn once per profile uniformly from $[\lfloor n/2 \rfloor, n-3]$, where n is the number of items. Items beyond the truncation point are unranked.

Complete preferences with ties. Starting from a strict and complete preference list, ties are introduced by randomly selecting contiguous ranges of items and merging them into indifference classes. Each generated profile contains two tie groups, with at least one item ranked outside any tie. The starting and ending position of each tie are selected randomly from possible values. Once these position are selected, we insert brackets into these positions and indicate that the items within the brackets are tied.

Partial order preferences. Partial-order preferences are generated as Erdős–Rényi DAGs (Erdős & Rényi, 1959) (a common approach in prior work (Brightwell, 1993; Jang et al., 2016)) oriented consistently with a latent total order. We use up to 100 nodes and edge counts of 50, 80, or 150 (denser graphs yielding fewer incomparable pairs).

E.1.2 Benchmark Design

The preference reasoning benchmark consists of task families, each targeting a different aspect of preference comprehension.

Atomic queries. Retrieval regarding single item within the preference profile:

- **Position query**: at what position does a given item appear in the agent's preference?
- **Item query**: which item does the agent rank at a given position k?

Comparative queries. Relative preference of multiple items by an agent:

- **Item comparison**: between 2 given items, which is preferred by the agent?
- **Item ranking**: given a set of four items, rank them according to the agent's preference.

Under strict and complete preferences, each question has a unique determined answer. Under strict and incomplete preferences, some questions become undetermined (e.g. the queried item may be absent, or k may exceed the length of the list), and the correct response is to indicate that the answer cannot be determined. Under complete preferences with ties, we additionally include a **top-k query**: does a given item belong to the agent's top-k alternatives? When k falls within a tie group, this question is undetermined.

For all question types, models are prompted to return their answer in a specified format. In the free-flow format, we do not provide explicit options. For undetermined questions, two free-flow variants are used: one that includes "if known", and one that does not. We additionally create MCQ versions for all problems and explore the impact of prompt format on model responses.

Bundle comparisons. A special case of comparative task. Given a single agent's preference ranking and two bundles $B, B' \subseteq A$, the model is asked which bundle the agent prefers. Pairs are drawn from three preference types (strict and complete, strict and incomplete, complete with ties), and are constructed to include both RS-comparable cases (where a definite answer exists) and RS-incomparable cases (where the correct response is that the comparison is undetermined).

Aggregative queries. Given a preference profile over multiple agents, the model is asked: "how many agents prefer item a over item b?" This requires scanning all agents' lists and tallying correctly. We evaluate accuracy (fraction of exact correct counts) and mean absolute error on incorrect responses.

Table 3: Preference reasoning benchmark composition. Profile size refers to the number of items in the preference list. In formats, "ff" refers to free-flow prompt, "ff_if_known" refers to free-flow prompt, added with "if known" phrase, and MCQ refers to multiple-choice question.

Task family	Pref. Type	Question types	Profile sizes	Formats	Prompts
Atomic	Strict + complete	Item query, position query,	50, 100, 200	ff, MCQ	360
	Strict + incomplete	Item query (undetermined), position query (undetermined),	50, 100, 200	ff, ff_if_known, MCQ	540
Comparative	Strict + complete	item comparison, item ranking	50, 100, 200	ff, MCQ	360
	Strict + incomplete	item comparison (undetermined), item ranking	50, 100, 200	ff, ff_if_known, MCQ	720
Bundle comparison	Strict + complete	RS-comparable, RS-incomparable, larger bundle, unequal bundle sizes	50, 100	ff, ff_if_known, MCQ	300
	Strict + incomplete	RS-comparable, RS-incomparable	50	ff_if_known	60
	Ties + complete	RS-comparable, RS-incomparable	50	ff_if_known	60
Top-k query	Ties + complete	Determined, Undetermined, break-down queries	50	ff, ff_if_known	120
Aggregative	Strict + complete	Agent count query	50, 100	ff, MCQ	120
	Strict + incomplete	Agent count query	50, 100	ff, MCQ	120
	Ties + complete	Agent count query	50, 100	ff	60
Partial order	Pairwise (Format 1)	Comparable pair, incomparable pair	100 nodes + {50, 80, 150} edges	ff, ff_if_known, MCQ	420
	Query-answer (Format 2)	Comparable pair, incomparable pair	30 nodes + {30, 50} edges	ff, ff_if_known, MCQ	120
Total					**3360**

Partial order queries. Given a preference specified as a set of pairwise comparisons, the model is asked whether item a is preferred to item b. We test both comparable pairs (connected by a chain of stated comparisons) and incomparable pairs (no such chain exists).

We additionally evaluate a second representation (**Format 2**) in which partial order information is conveyed as query-answer pairs: each data point specifies a subset of items and the most preferred item within it. Although logically equivalent to pairwise comparisons, this representation requires an additional inference step to recover the underlying structure.

E.1.3 Prompt Counts

The complete breakdown of the types of questions asked for each task, preference-type, and size, are provided in Table 3. Total count of 3360 prompts includes different prompting format. The unique number of prompts, not multiplying by format, is 1740.

E.2 Algorithmic Reasoning Tasks

E.2.1 Instance Generation

Preference sampling. Strict and complete preferences are sampled under the impartial culture model, with each agent's ranking drawn uniformly at random from all permutations of A. Complete preferences with ties are derived by post-processing: adjacent items in a strict ranking are merged into a tie-group independently with probability p (p=0.2 for house allocation; p=0.1 for stable matching). Strict and incomplete preferences are obtained by truncating each agent's ranking to a length drawn uniformly from $[\max(1, \lfloor 0.15n \rfloor), \lfloor 0.51n \rfloor]$. An agent is never assigned an alternative it has not ranked, and is left unmatched instead, following the standard convention that unranked alternatives are unacceptable. This is what keeps each solution concept well defined, so that an infeasible instance is infeasible because of the instance rather than because of how the truncation is read.

Rejection sampling with distinctness constraints. Instances are selected via rejection sampling from the stream of randomly generated profiles. Every accepted instance must satisfy a *distinctness constraint*: the optimal outcome for each solution concept evaluated in that setting must be distinct from every other, and each welfare-based outcome (utilitarian, egalitarian, rank-maximal) must be strictly better under its own objective than the canonical outcome and the other welfare-based outcomes. Pareto optimality and the nested stability notions are the exceptions, since under strict preferences the core, utilitarian and rank-maximal allocations are all Pareto-optimal, and a super-stable matching is also strongly and weakly stable. Apart from these nested notions, this ensures that correct model responses can be attributed to the target notion.

E.2.2 Benchmark Design

For each combination of domain, preference type, and size $n \in \{10, 30\}$, we collect 30 instances satisfying the distinctness constraint. Each instance gives rise to two types of tasks: **generation**, where the model computes a solution from scratch given the preference profile, and **selection**, where it chooses among a set of presented candidates. Generation prompts include a definition of the target notion and, by default, an instruction that the model may return $\{\}$ (i.e. an empty solution) if it determines no valid solution exists. Selection prompts present one candidate per solution concept evaluated in that setting, together with one or two *contaminated* candidates obtained by randomly swapping one or two assignments in the canonical algorithm's output, with options in uniformly random order. All models are evaluated on the same instances in a zero-shot, single-turn setting at their default temperature.

The benchmark is structured around two components reflecting a natural partition of the solution concepts.

Strict preferences. Under strict preferences, all solution concepts of interest are guaranteed to exist for every instance. For each applicable solution concept, one generation prompt is produced per instance; a further *unspecified* generation prompt is included in which no notion or definition is provided and the model is simply asked to compute "a matching" or "an allocation." This condition tests whether models default to the canonical algorithm in the absence of an explicit instruction. One selection task is also produced per instance, asking the model to select the option it "prefers the most" from the candidate set without naming a target notion; this tests whether models' default preferences align with canonically characterised solutions.

We evaluate house allocation, object allocation, and stable matching under strict and complete preferences as the primary settings. We additionally evaluate house allocation under strict and incomplete preferences to examine whether incompleteness affects performance; since this involves a single domain with a narrower set of notions, we treat it as a secondary setting.

Preferences with ties. We evaluate house allocation and stable matching under complete preferences with ties.[7] For certain notions such as weak core, weak stability, and the welfare and rank-based notions, the algorithms required are shared with the strict setting (e.g. TTC for weak core, DA for weak stability, Hungarian algorithm for utilitarian welfare, etc.). This enables a direct comparison of

[7] Given the similarity between house allocation and object allocation, we do not separately evaluate object allocation with ties.

Table 4: Benchmark composition for standard tasks (solution concepts guaranteed to exist). Each setting uses 30 instances. Generation includes one prompt per listed notion (each with the {} instruction by default) plus one *unspecified* prompt; selection asks the model to choose its most preferred candidate without specifying a target notion. n_{gen} and n_{sel} denote prompts per instance; total $= 30 \times (n_{\text{gen}} + n_{\text{sel}})$. Harder notions under preferences with ties (strict core; strong and super stability) are excluded from this table and evaluated separately. † Treated as a secondary setting. ‡ PO for house allocation with ties is verified via strong Pareto-optimality under weak preferences.

Domain	Pref. type	Notions (generation)	n_{gen}	n_{sel}	Prompts
House allocation	Strict + complete	Core, PO, UW, EW, RM	6	1	210
	Strict + incomplete†	Core, RM, MCPO	4	1	150
	Ties + complete	Weak core, PO‡, UW, EW, RM	6	1	210
Object allocation	Strict + complete	PO, UW, EW, RM	5	1	180
Stable matching	Strict + complete	Stable, UW, EW, RM	5	1	180
	Ties + complete	Weakly stable, UW, EW, RM	5	1	180
Subtotal per size n					**1,110**

performance across preference types. Table 4 summarises all settings and prompt counts for these standard tasks.

Ties additionally involve three solution concepts not guaranteed to exist for every instance: the strict core (house allocation), and strong and super stability (stable matching). These are excluded from the standard task structure above and are instead evaluated under a dedicated experimental design. For each notion, we collect 30 *feasible* instances (where a valid solution exists, drawn from the standard pool) and 30 *infeasible* instances (where no solution satisfying the notion exists). Eleven of the infeasible strong-stability instances (two at $n = 10$ and nine at $n = 30$) were later found to admit a strongly stable matching and are excluded from all reported results.[8] Each instance receives four prompt variants forming a 2×2 design:

- **Generation with and without escape hatch.** By default the model is instructed it may return {} if no solution exists; in the ablation this instruction is removed. The latter tests whether models generate overconfident solutions when no valid one exists, or express infeasibility through other means (e.g. returning `None`, populating the output dictionary with `"None"` values, or issuing a natural-language disclaimer).
- **Selection with and without NOTA.** We vary whether a "None of the Above" (NOTA) option is included in the candidate set. The NOTA condition tests whether models correctly identify infeasibility when given an explicit escape; the no-NOTA condition tests whether they select a spurious solution under pressure.

This yields 2 (feasibility) $\times$ 2 (task type) $\times$ 2 (escape hatch / NOTA) $\times$ 3 (notions) $\times$ 30 (instances) $=$ 720 additional prompts per size.

Total. The benchmark comprises **1,830 prompts per size** n, consisting of 1,110 from the standard settings in Table 4 and 720 from the harder-notions experiment, for a total of **3,660 prompts** across both sizes $n \in \{10, 30\}$. Each prompt is answered independently by all four models, yielding **14,640 model responses** in total.

F Model and Inference Details

We evaluate four frontier models, listed in Table 5 with their exact identifiers, hosting providers, and per-call inference settings. All four are accessed through their providers' hosted APIs at decoding time, with no fine-tuning. Temperature is set to 1.0 for GPT-5.2, Claude-4.5-Sonnet and OSS-120B, and Gemini-2.5-Pro runs at its provider default; we do not override top-p, top-k, or frequency /

[8] Infeasible instances for super stability are constructed to admit a strongly stable solution, to test whether models conflate the two concepts.

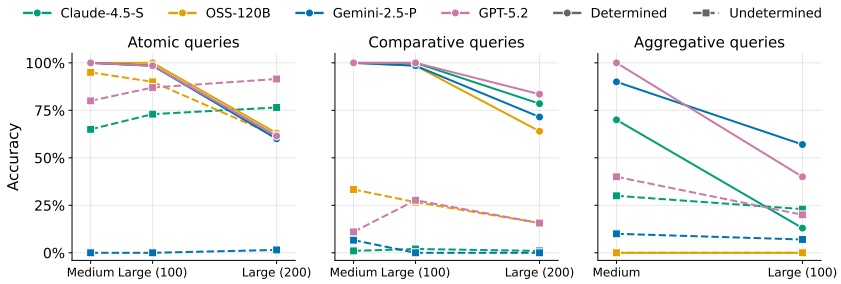


Figure 6: The models' accuracy degrades with increasing problem size in both determined and undetermined queries.

presence penalties. The full inference scripts (auth-key handling, retry / resume logic, code-assisted tool wiring) are released alongside this paper.

Temperature. We ran a temperature-0 condition for Claude-4.5-Sonnet and OSS-120B. For these two we re-ran, at temperature 0, 40 hard-notion generation instances at $n = 10$ that were generated separately from the main benchmark (16 strict core, 16 strongly stable and 8 super stable). Ten of the 40 are labelled feasible and 30 infeasible; one instance labelled infeasible was later found to admit a strongly stable matching.

The results are essentially unchanged. Overall accuracy on this subset is 0.00 for Claude-4.5-Sonnet at temperature 1.0 and 0.03 at temperature 0, and 0.03 at both temperatures for OSS-120B, and the rate at which each model correctly identifies an infeasible instance does not move between the two temperatures. The undetermined preference queries behave the same way: on 60 RS-incomparable bundle comparisons, both models abstain on none of them at either temperature, so the lexicographic default is deterministic as well. Temperature is not what drives these failures, and they are not sampling noise.

Table 5: Inference settings for the four evaluated models. "Max tokens" lists the per-call output budget on the generation task and (where it differs) the selection task. "Reasoning effort" is the provider-side knob exposed by the OpenAI Responses API.

Label	Provider / API	Model identifier	Max tokens	Timeout	Reasoning effort
GPT-5.2	OpenAI (Responses API)	`gpt-5.2`	30k (gen) 40k (sel)	20 min	medium (concise summary)
Claude Sonnet 4.5	Anthropic (Messages API)	`claude-sonnet-4-5-20250929`	30k (gen) 15k (sel)	20 min	—
Gemini 2.5 Pro	Google (`google.genai`)	`gemini-2.5-pro`	provider default	providerdefault	—
GPT-OSS 120B	Groq (chat-completions)	`openai/gpt-oss-120b`	20k (gen) 15k (sel)	providerdefault	—

G Additional Results: Preference Reasoning

This appendix provides the per-model, per-condition results underlying the findings about preference reasoning. In each subsequent section, we present the accuracy of models on each task, and break down the assumptions they make leading to determined responses in undetermined problems.

G.1 Additional Figures

In Figure 6, we show that as the preference profile size increases, the accuracy of models decrease significantly on determined queries. For atomic and comparative queries, models remain almost perfect accuracy up to size 100, but begins to fail at size 200. For aggregative query, the threshold is much lower as model errors increase significantly at size 100, presumably related to the increase of both number of agents and length of preference list leading to quadratic increase in the profile. The

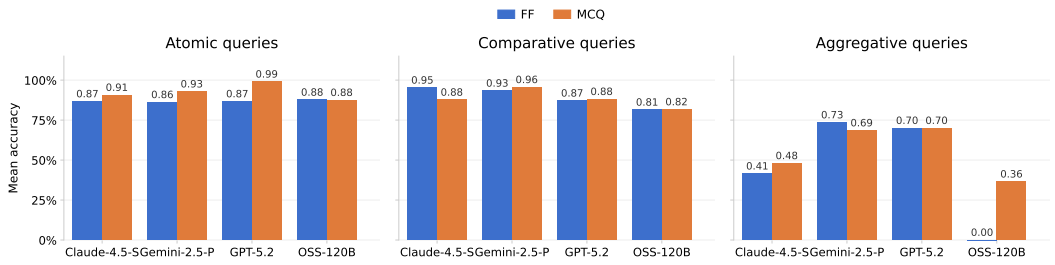


Figure 7: Comparing LLMs' performance with different prompting formats on determined preference queries, split by task type. The most notable effect of format is OSS-120B on aggregative tasks. While completely failing to perform accurate aggregation in free-flow format, it selects the correct count in MCQ for a certain portion, yet still significantly trailing other 3 models

effect of sizes on undetermined problems is less pronounced and model variations exist, reflecting their diverse behavior pattern, either affirming incomparability with greater confidence or becoming more prone towards assumptions as input complexity increases. A breakdown of model performance on different tasks is provided in subsequent sections.

G.2 Detailed results of reasoning over total-order preference profiles

Atomic and comparative queries Table 6 reports atomic and comparative queries on strict and complete preferences, which have a determined answer.

Accuracy on item/position queries are high up to n=100, but significantly decrease as preference profile hit n=200. Rate of decline varies across 4 models.

Accuracy on pairwise comparison and ranking also decreases. Pairwise comparison is less affected than atomic query (presumably due to relative rather than precise location required), while the accuracy is lower in the ranking task, due to the accumulation of pairwise comparisons needed to provide the ranking of 4 items.

Table 6: Accuracy on determined atomic and comparative queries, under free-form prompt.

	Item query			**Position query**			**Comparison**			**Ranking**		
Model	50	100	200	50	100	200	50	100	200	50	100	200
Gemini-2.5-P	1.00	0.967	0.50	1.00	1.00	0.70	1.00	1.00	0.80	1.00	0.967	0.633
OSS-120B	1.00	1.00	0.68	1.00	1.00	0.59	1.00	1.00	0.83	1.00	0.967	0.45
Claude-4.5-S	1.00	0.967	0.50	1.00	1.00	0.73	1.00	1.00	0.90	1.00	1.00	0.80
GPT-5.2	1.00	0.967	0.60	1.00	1.00	0.633	1.00	1.00	1.00	1.00	1.00	0.667

Table 7 reports accuracy on undetermined atomic and comparative queries based on strict and incomplete preferences.

On atomic queries, Gemini most often fails to acknowledge indeterminacy. It invokes assumptions that lead to a definite output, including assumed typographical error in query, and manual completion of the incomplete preference list, by filling the missing items in lexical order behind the listed ones.

On comparative queries, when both items to be compared are not listed, models make error by choose one definite answer, using assumption such as lexical ordering. This accumulates in the ranking task, where all models most often return a full ranking of 4 items, even when only a partial ranking can be determined, as 2 of the items are not listed in the preference profile.

Aggregative queries Table 8 reports aggregation accuracy across the three preference types. The parenthetical numbers under incomplete profiles give the fraction of *incorrect* responses that match the count where agents ranking the preferred item in question but not the other one are included (an implicit and unwarranted disambiguation). OSS-120B fails entirely across all conditions; the other three models are reasonable at $n = 50$ on strict-and-complete profiles but degrade sharply at $n = 100$

Table 7: Accuracy on undetermined atomic and comparison tasks, with "ff + if known" prompts. Correct response is: for item query, "unknown" (queried item absent from profile); for position query, "unknown" (queried position out of bounds); for comparison, "unknown" (both items not listed); for ranking, return only the listed items. A score of 0 indicates the model never gives the correct response.

	Item query			**Position query**			**Comparison**			**Ranking**		
Model	50	100	200	50	100	200	50	100	200	50	100	200
Gemini-2.5-P	0.50	0.30	0.00	0.63	0.40	0.23	0.63	0.50	0.37	0.17	0.20	0.00
OSS-120B	1.00	0.93	0.87	1.00	1.00	1.00	1.00	0.57	0.22	0.00	0.00	0.00
Claude-4.5-S	0.87	0.80	0.93	1.00	1.00	0.83	0.70	0.90	0.47	0.00	0.00	0.00
GPT-5.2	1.00	1.00	1.00	1.00	1.00	1.00	0.43	1.00	1.00	0.00	0.00	0.00

and on tie-containing profiles. The mean absolute errors of the incorrect responses are shown in Table 9. Errors for incomplete preferences are not calculated, since the errors are not only caused by counting, but also induced by assumption on preference over item not being listed.

Table 8: Accuracy on the aggregative task. For incomplete preferences, values in parentheses give the fraction of *incorrect* responses matching the assumption made above.

	Strict + complete		**Strict + incomplete**		**Complete + ties**	
Model	Size 50	Size 100	Size 50	Size 100	Size 50	Size 100
Gemini-2.5-P	0.90	0.57	0.10 (0.89)	0.17 (0.52)	0.70	0.50
OSS-120B	0.00	0.00	0.00 (—)	0.00 (—)	0.00	0.00
Claude-4.5-S	0.70	0.13	0.30 (0.43)	0.23 (0.17)	0.30	0.10
GPT-5.2	1.00	0.40	0.40 (0.83)	0.40 (0.33)	0.80	0.33

Table 9: Mean absolute error of incorrect responses for the aggregation task.

	Strict + complete		**Complete + ties**	
Model	Size 50	Size 100	Size 50	Size 100
Gemini-2.5-P	1.00	4.54	1.08	3.93
OSS-120B	4.60	27.7	9.60	32.5
Claude-4.5-S	1.80	2.62	2.65	3.22
GPT-5.2	0.00	2.83	1.13	2.69

OSS-120B on aggregation: a note on catastrophic failure. OSS-120B returns 0.00 accuracy on aggregation across all preference types and both sizes. Inspection of responses shows that the model returns counts that differ from the ground truth by large margins, rather than failing in a structured way. We treat this as a per-model anomaly and exclude OSS-120B from finding-level claims about aggregation that depend on the structured-error pattern.

G.3 Special undetermined queries: detailed bundle comparison and top-k results

Bundle comparison, all preference types and sizes. Table 10 gives the full per-model breakdown of the lexicographic-ordering finding. The values in parentheses give the fraction of RS-incomparable responses where the model applied lexicographic ordering by highest-ranked element. Across all twelve (model $\times$ preference type) cells, the lexicographic share is 80% or higher; the only deviation from pure lexicographic behaviour is Claude-4.5-S, which occasionally uses the sum of item ranks instead, which can yield the opposite conclusion.

Our results score an undetermined response correct only when it abstains. A model that commits to an answer while naming the assumption behind it is arguably doing something different from one that commits silently, so it is worth asking how much of the gap survives if the former is

credited too. We sorted every RS-incomparable response into four kinds: an explicit abstention, a completion whose lexicographic rule is stated as an assumption, a silent lexicographic completion, and a non-lexicographic answer.

Table 11 reports the strict scoring (abstention only) against the lenient scoring (abstention or a flagged assumption). The lenient scoring raises accuracy by at most 0.17, and every model stays far below its accuracy of roughly 1.00 on the matched determined comparisons. Flagged assumptions are rare, at most 0.17 of responses, while silent lexicographic completion accounts for 0.70 to 1.00. It is this silent completion, rather than flagging that goes uncredited, that the epistemic results measure.

Additionally, we test the bundle comparison tasks on two more schemes. Firstly, we increase the size of bundles to be compared. We randomly generate one bundle, and substitute several items within it with both more preferred and less preferred items, this creates a pair of equal-sized bundles that are RS-incomparable. Secondly, we create bundle pairs with unequal sizes. We select a single item that ranked relatively highly to form a one-item bundle, and then select multiple items less preferred to it to form the second bundle, with varying size up to half of the preference list (over 20 items). In both cases, models keep the lexicographic bias when asked to select the preferred bundle. Results are in Table 12.

Table 10: Bundle comparison accuracy ($n = 50$). For RS-comparable pairs (a definite answer exists), accuracy is reported. For RS-incomparable pairs (no determinate answer exists), the main value is the fraction of responses that correctly select "can't decide," with the fraction applying lexicographic ordering by highest-ranked element in parentheses.

	Strict + complete		**Strict + incomplete**		**Complete with ties**	
Model	RS-comp.	RS-incomp.	RS-comp.	RS-incomp.	RS-comp.	RS-incomp.
Gemini-2.5-P	1.00	0.00 (1.00)	1.00	0.00 (1.00)	1.00	0.20 (0.80)
OSS-120B	1.00	0.00 (1.00)	0.85	0.00 (1.00)	1.00	0.00 (1.00)
Claude-4.5-S	1.00	0.00 (0.90)	1.00	0.00 (0.90)	1.00	0.10 (0.90)
GPT-5.2	1.00	0.17 (0.83)	1.00	0.07 (0.93)	1.00	0.10 (0.90)

Table 11: Bundle comparison scoring for RS-incomparable pairs. In the strict scoring, only "can't decide" is accepted. In the lenient scoring, returning a lexicographically preferred bundle while explicitly stating the rule as an "assumption" is allowed

scoring	**Gemini-2.5-P**	**OSS-120B**	**Claude-4.5-S**	**GPT-5.2**
Abstention only (strict)	0.10	0.00	0.03	0.17
Plus flagged assumption (lenient)	0.23	0.00	0.20	0.20

Table 12: Additional results on RS-incomparable bundles. The main value is the fraction of responses that correctly indicate the comparison is undetermined ("can't decide"), with the fraction using lexicographic ordering in parentheses.

model	increased bundle size	one vs many
Gemini-2.5-P	0.00 (1.00)	0.00 (1.00)
OSS-120B	0.00 (1.00)	0.00 (1.00)
Claude-4.5-S	0.00 (0.80)	0.00 (1.00)
GPT-5.2	0.00 (1.00)	0.00 (1.00)

Top-k queries (complete preference with ties): full prompt-variant results. A unique problem for preference with ties involves indeterminacy due to tie-breaking. We ask whether an item belongs to the top-k choices of the agent. The answer is determined, when exactly k choices can be uniquely selected from the preference list, or when the queried item is not within a tie. Indeterminacy arises when the "kth" position lies within a tie, and the queried item is part of the tie, since there is no definitive way of identifying the "top-k" items to judge whether the agent belongs to them. Table 13 shows the way a top-k membership question is asked changes both the answer the model gives and the listing behaviour underneath it. Prompt 1 asks this directly with "yes / no / uncertain" as the

available answers. Prompt 2 instead asks the model to list the agent's top-k items without mentioning A; correctness is then about whether the response breaks the tie to return only k items, or returns the entire tie group. Prompt 3 combines the two: first asks for the top-k list, then asks whether A belongs, with only "yes / no" available.

Table 13: Top-k selection task under preferences with ties. k is chosen so that the kth position falls within a tie group, and item A is a member of that tie. Prompt 1 directly asks whether A is in the top-k; Prompt 2 asks for the top-k list only; Prompt 3 asks for the list and then asks whether A belongs.

	Prompt 1		**Prompt 2**		**Prompt 3**	
Model	uncertain	yes/no	keeps tie	breaks tie	yes	no
Gemini-2.5-P	0.00	0.90/0.10	0.00	1.00	1.00	0.00
OSS-120B	0.30	0.70/0.00	0.10	0.90	0.90	0.10
Claude-4.5-S	0.40	0.60/0.00	0.50	0.50	1.00	0.00
GPT-5.2	0.63	0.37/0.00	0.60	0.40	0.90	0.10

Two contrasts in the table are worth surfacing. First, the Prompt 1 "uncertain" column is the only place the data set offers a calibrated abstention rate when the response options include "uncertain": Gemini never picks it (consistent with its over-commitment in other tasks), whereas GPT-5.2 picks it 63% of the time. Removing the uncertain option in Prompt 3 does not redistribute these uncertainties evenly between yes and no; almost all are absorbed into yes. Second, the Prompt 2 "keeps tie" column shows a wide spread across models (0% for Gemini, 60% for GPT) when no specific item is mentioned, which then collapses to 90–100% across the board in Prompt 3 once a specific item is named. Both contrasts point to the same underlying observation: surface phrasing reshapes the model's response on a fixed underlying ambiguity. Whether this should be read as a calibration failure (the model has no stable answer to give) or as a context-sensitivity feature (the model adapts to the apparent intent of the prompt) depends on the deployment context; we report the data without taking a position.

G.4 Partial-order pairwise comparison: detailed results

Partial-order queries on comparable pairs. Table 14 reports the per-model breakdown of accuracy on comparable partial-order pairs along both complexity dimensions: edge count varied at fixed chain length 9, and chain length varied at fixed 150 edges. Gemini-2.5-P and Claude-4.5-S handle long chains in dense graphs reliably; OSS-120B and GPT-5.2 degrade in both dimensions, sometimes failing to find chains they had earlier identified at lower density.

Table 14: Accuracy on comparable item pairs in partial orders ($n = 100$ items). Left block: chain length fixed at 9, edge count varied. Right block: edge count fixed at 150, chain length varied.

	Chain length = 9			**Edges = 150**		
Model	50 edges	80 edges	150 edges	chain of 7	chain of 5	chain of 3
Gemini-2.5-P	1.00	1.00	1.00	1.00	1.00	1.00
OSS-120B	1.00	1.00	0.23	0.50	0.73	1.00
Claude-4.5-S	1.00	1.00	1.00	1.00	1.00	1.00
GPT-5.2	1.00	0.93	0.17	0.47	0.80	1.00

Incomparable pairs: free-flow vs. MCQ across graph densities. Table 15 gives the per-density performance of undetermined partial order queries. The dominant pattern: GPT-5.2 and OSS-120B acknowledge incomparability reliably even under free-flow; Gemini-2.5-P declines as density grows; Claude-4.5-S almost never acknowledges incomparability under free-flow, collapsing to 0% at 150 edges. Switching to MCQ recovers near-perfect accuracy across all four models.

When Claude-4.5-S and Gemini-2.5-P fail to acknowledge incomparability under free-flow, inspection of the reasoning traces reveals consistent patterns: assuming the user has made a typographical error in specifying the input and constructing an artificial chain to make the items comparable; selecting the

Table 15: Accuracy on incomparable item pairs in partial orders ($n = 100$ items), across graph densities and prompt formats.

	Free-flow			MCQ		
Model	50 edges	80 edges	150 edges	50 edges	80 edges	150 edges
Gemini-2.5-P	0.80	0.63	0.50	1.00	1.00	1.00
OSS-120B	1.00	1.00	1.00	1.00	1.00	1.00
Claude-4.5-S	0.13	0.07	0.00	1.00	1.00	1.00
GPT-5.2	1.00	0.97	0.97	1.00	1.00	1.00

item that appears anywhere in the input when the other does not; and inferring preference from the relative number of items each one dominates or is dominated by. These are structured assumptions of the same kind documented for bundle comparisons and aggregation. A breakdown of the assumptions is listed in Table 16.

Table 16: Distribution of assumptions that models make to give a definite answer to the unanswerable partial order comparison task under free-flow prompt.

Model	Gemini-2.5-P	Claude-4.5-S
Determining via relative preference	0.30	0.87
Assuming typo in query or conditions	0.30	0.00
Acknowledging lack of chain yet still committing a choice	0.40	0.13

Comparable pairs under MCQ: the cost of an explicit "unknown" option. Table 17 reports the effect of MCQ prompt on determined partial order query, evaluated at chain length 9 with 150 edges over 100 nodes. On these inputs a chain in the input determines the answer, so the correct response is to identify the preferred item rather than abstain. Adding the MCQ "not enough information to decide" option leaves Gemini unaffected (1.00 in both formats) but degrades the other three models. Claude shows the largest drop, from 100% under free-flow to 33% under MCQ; OSS and GPT decline further from already-weak free-flow baselines. Together with Table 15, the result is that the MCQ option moves models in opposite directions on the two question types: where Claude was over-committing (incomparable pairs), it is pulled back toward abstention; where it was answering correctly (comparable pairs), it is pulled away from a definite answer.

Discussion on the effect of prompt formats The determined partial order query is the only preference reasoning task where we observe the model falsely indicating indeterminacy for a determined task, potentially due to the complexity of partial order reasoning. In reasoning with total order profiles, models almost always commit to a determined response, either correct or incorrect, even tending to commit when the problem is logically undetermined. As we implement alternative prompt formats such as MCQ to them, the performance does not change significantly for most determined tasks (Figure 7), while free-flow with "if-known" prompt and MCQ lead the models to acknowledge indeterminacy more often overall in the undetermined tasks, as we demonstrated in the main text (Figure 2).

Table 17: Accuracy on comparable item pairs in partial-order preferences ($n = 100$ items, 150 edges, target chain length 9), under free-flow and MCQ formats.

Model	**Free-flow**	**MCQ**
Gemini-2.5-P	1.00	1.00
OSS-120B	0.23	0.10
Claude-4.5-S	1.00	0.33
GPT-5.2	0.17	0.10

Alternative partial-order representation (Format 2). Table 18 reports results on a query-answer representation of partial orders, where each input data point specifies a subset of items and the most preferred item within that subset (rather than a pairwise comparison). This representation is informationally equivalent to a partial order but requires an inference step to recover the underlying

pairwise structure. The same qualitative pattern as Format 1 holds: free-flow performance varies sharply across models, with Claude-4.5-S collapsing on incomparable pairs; MCQ recovers near-perfect accuracy across all models. The replication across representations supports the MCQ-recovers-abstention pattern not being an artifact of the input format.

Table 18: Accuracy on incomparable item pairs in partial-order preferences using the query-answer representation ($n = 30$ items), across graph densities and formats.

	Free-flow		MCQ	
Model	30 edges	50 edges	30 edges	50 edges
Gemini-2.5-P	0.50	0.20	1.00	1.00
OSS-120B	1.00	1.00	1.00	1.00
Claude-4.5-S	0.20	0.00	1.00	1.00
GPT-5.2	1.00	1.00	1.00	1.00

G.5 Alternative preference presentation: natural language rendering

To verify whether the model biases on undetermined queries are an artifact of the symbolic lists we use to represent preference profiles by default, we re-ran the same queries under three levels of natural-language rendering of the profile. To make the task as easy as possible, each prompt was also cut down to the queried agent's list alone, which is the only preference information the question needs.

- Level 1 is the original symbolic list, for example ["C9", "C33", ...] given for agent A38.
- Level 2 states the same ranking in words, for example "A38 most prefers C9, then C33, and so on".
- Level 3 renders the items as real-world objects and narrates a person choosing a meal, for example "They like pizza the most, then sushi, then ...", with the question rewritten as "Which bundle does the person prefer between {sushi, tiramisu} and {burgers, jambalaya}?".

Alongside the bundle task we include an undetermined atomic query, asking about preference of an item absent from the preference list.

We generated 60 undetermined bundle comparisons with the construction rule of Appendix E.1, together with 60 matched determined comparisons in which one bundle clearly dominates, and ran both on GPT-5.2 and OSS-120B.

As shown in Table 19, correct abstention on the bundle task is at or near 0.00 at all three levels for both models, while the matched determined comparisons are answered correctly in every case. The near-zero abstention is therefore a genuine failure to recognize incomparability, not low ability on the task, and natural-language rendering does not recover the correct "can't decide".

What the models do recognize depends on the source of the indeterminacy. They almost never flag that two bundles of ranked items are formally incomparable, but they flag an unknown answer far more often when the queried item was never ranked. The natural-language levels do not change either behaviour, so the lexicographic default is not an artifact of how the list is written.

H Additional Results: Algorithmic Reasoning over Preferences

H.1 Performance Differs Across Solution Concepts

As shown in Figure 8, all models achieve higher and more consistent accuracy on notions computed by canonical algorithms (core/TTC, PO/SD, stability/DA) than on non-canonical ones (UW, EW, RM). At $n = 30$, most models continue to perform well on canonical notions, while accuracy on UW and RM falls sharply, approaching zero for all models except GPT-5.2. EW occupies an intermediate position, with GPT-5.2 and Gemini-2.5-P maintaining reasonable accuracy on house allocation but all models failing in stable matching. The model hierarchy, unlike in Section 3, is consistent: GPT-5.2

Table 19: Accuracy on undetermined problems, with different levels of natural language representation of preferences. Level 1: original symbolic list ("A38": ["C9", ...]). Level 2: same items, natural sentences. Level 3: items replaced by real dishes, narrated as a person choosing a meal. Each cell is the fraction of 60 responses that correctly indicate indeterminacy. The last row is the matched determined control, where one bundle dominates and a definite answer exists.

Task	Model	Level 1	Level 2	Level 3
RS-incomparable bundle	GPT-5.2	0.00	0.00	0.00
RS-incomparable bundle	OSS-120B	0.00	0.00	0.02
unranked item	GPT-5.2	0.48	0.50	0.42
unranked item	OSS-120B	0.50	0.53	0.33
determined bundle (control)	both	1.00	1.00	1.00

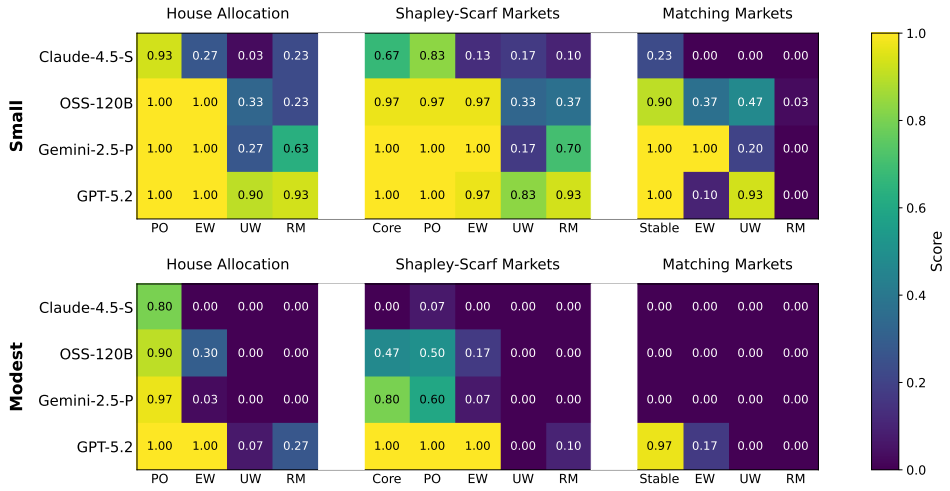


Figure 8: Performance of LLMs in terms of achieving specific notions across all three domains under strict and complete preferences, for instance sizes Small and Modest. Each cell is the fraction of instances where the model's output satisfies the target notion.

performs best overall, Gemini-2.5-P and OSS-120B show mixed relative performance across domains and notions, and Claude-4.5-S performs worst.

H.2 Robustness to Preference Structure

Performance is robust to incompleteness.

For notions evaluated under both strict and incomplete preferences, performance does not decrease when preference lists are truncated (Figure 9). In some cases it is marginally better, likely because shorter lists reduce the number of reasoning steps required. Models also handle MCPO allocations (a

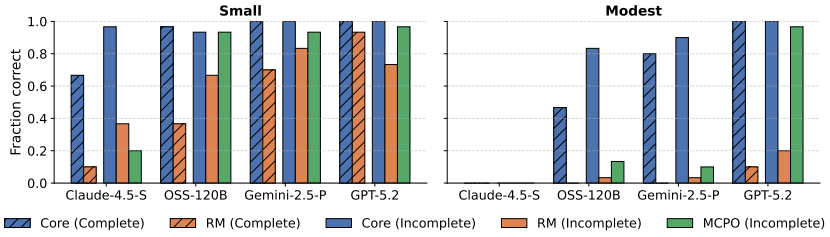


Figure 9: Performance on house allocation under strict and complete vs. strict and incomplete preferences.

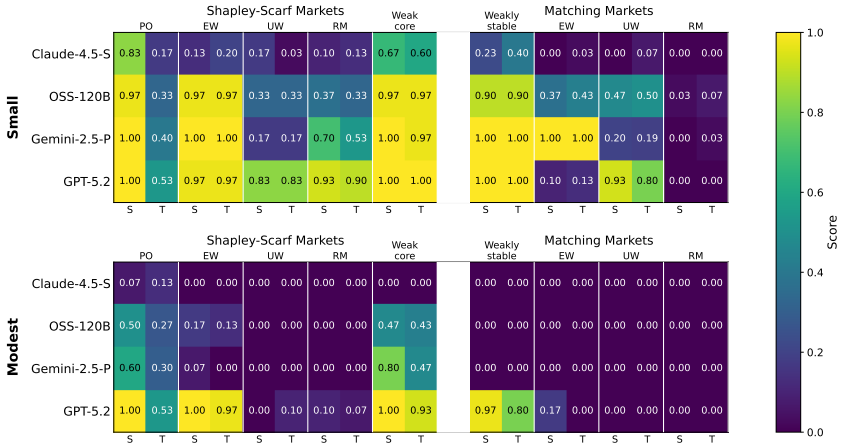


Figure 10: Difference in LLMs' performance in computing various types of solutions with strict preferences (S) and preferences having ties (T).

notion specific to the incomplete setting) reasonably well, though with a sharper size-induced decline than for the core.[9]

Performance is robust to ties, when the same algorithm works.

The introduction of ties likewise does not significantly affect performance on notions whose underlying algorithm is unchanged relative to the strict setting, such as the weak core, UW, EW, and RM. The one case where performance does drop is Pareto-optimality in house allocation: TTC guarantees a PO outcome under strict preferences but not under ties, so models that correctly apply TTC continue to succeed on the core yet fail to achieve PO in the same instances.

H.3 Generation without a Specific Objective

When asked to compute a solution without specifying a target notion, all models default to the canonical algorithm for the relevant domain across all preference types (Figure 11). The gap between hatched and solid bars reveals that models often correctly identify the canonical algorithm yet fail to execute it, particularly at $n = 30$.[10] An exception arises in object allocation at $n = 10$, where some models attempt to compute a solution satisfying stronger notions such as UW rather than simply applying SD; this tendency largely disappears at $n = 30$, consistent with models falling back to the simpler SD procedure as more demanding computations exceed their reasoning budget.

H.4 Failures in Infeasibility Detection

Models rely on heuristic search, which fails at scale.

Models largely fail to compute strict core, strong stability, and super stability even at $n = 10$ (Table 20), with performance declining further at $n = 30$. Where models do succeed at $n = 10$, particularly GPT-5.2 and Gemini-2.5-P, inspection of reasoning traces reveals that they rely on heuristic search rather than correct algorithms: they enumerate candidate solutions, verify blocking conditions case by case, or use variants of the correct algorithm that still require heuristic completion (see Appendix K). This strategy can succeed for small instances but fails systematically at larger ones, where the search space grows beyond the model's effective reasoning capacity.

Models cannot reliably detect infeasibility.

[9] When a max-cardinality matching is infeasible, models consistently assign unranked items to agents rather than leaving them unmatched, revealing an implicit assumption that receiving any item is preferable to receiving none.

[10] We identify the algorithm the model attempts to use, using the method described in Appendix O.

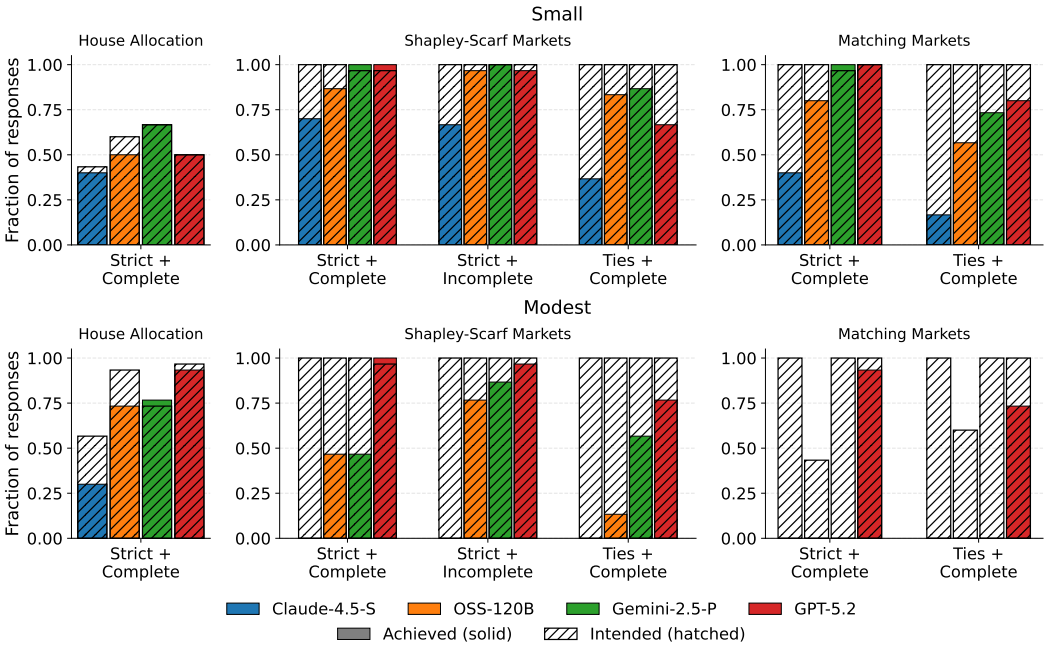


Figure 11: Fraction of responses where LLMs *intend* to use the canonical algorithm (hatched) and where they *achieve* the corresponding solution (solid), across domains and preference types, for $n = 10$ and $n = 30$.

Table 20: Hard notions: accuracy by model, notion, instance size, and treatment. Each cell shows the fraction of correct responses. For **E** rows, the value in brackets is the fraction where the model returned an invalid solution—i.e., a response that is neither a timeout nor a valid one-to-one allocation. For **NE** rows with the $\{\}$ instruction, a response is correct only if the model returned $\{\}$. For **NE** rows without the instruction, a response is correct if it is neither a timeout nor a valid one-to-one allocation. A superscript * indicates that at least one response timed out and was counted as incorrect; timeout counts are given in the note below. Each cell has 30 instances, except strongly stable **NE** rows, which have 28 ($n = 10$) and 21 ($n = 30$) instances after removing instances found to admit a strongly stable matching.

Notion	n		Claude-4.5-S with {}	Claude-4.5-S without {}	OSS-120B with {}	OSS-120B without {}	Gemini-2.5-P with {}	Gemini-2.5-P without {}	GPT-5.2 with {}	GPT-5.2 without {}
Strict Core	10	**E**	0%	10%	3%	3%	17%	17%	100%	77%
		NE	0%	0%	0%	0%	3%	0%	80%	20%
	30	**E**	0% [7%]	0%	0% [17%]	0% [3%]	0% [3%]	3%	10%	7%
		NE	0%	0%	13%	0%	0%	0%	0%	0%
Strongly Stable	10	**E**	0%	7% [3%]	23% [67%]	37% [20%]	70% [10%]	90%	97% [3%]	90%*
		NE	0%	0%	86%	61%	71%	0%	68%*	79%*
	30	**E**	0% [7%]	0% [13%]	0% [97%]	0% [27%]	0% [10%]	0% [7%]	23% [40%]*	40% [17%]*
		NE	0%	19%	95%	19%	10%	5%	62%*	33%*
Super Stable	10	**E**	0% [13%]	0%	3% [80%]	10% [67%]	23% [60%]	33%	83% [3%]*	80%*
		NE	7%	0%	87%	87%	73%	0%	67%*	50%*
	30	**E**	0% [53%]	0% [10%]	0% [100%]	0% [100%]	0% [20%]	0%	0% [93%]	3% [70%]*
		NE	57%	10%	97%	100%	40%	0%	97%	83%*

Timeouts. *GPT-5.2 with {}:* Strongly Stable n=10, **NE**: 8; Strongly Stable n=30, **E**: 3, **NE**: 4; Super Stable n=10, **E**: 4, **NE**: 9. *GPT-5.2 without {}:* Strongly Stable n=10, **E**: 2, **NE**: 6; Strongly Stable n=30, **E**: 5, **NE**: 12; Super Stable n=10, **E**: 6, **NE**: 15; Super Stable n=30, **E**: 3, **NE**: 2. All timeouts are counted as incorrect.

The pattern on infeasible instances reveals two distinct failure modes. The first is a *default-to-empty bias*: models that achieve high NE accuracy often do so by returning $\{\}$ indiscriminately, as evidenced by the simultaneously high invalid-solution rates (bracket values) on feasible instances. The apparent NE accuracy for these models reflects a bias toward claiming infeasibility rather than

genuine detection. This bias is also sensitive to instance difficulty: some models that return $\{\}$ rarely at $n = 10$ return it far more often at $n = 30$, when the problem is large enough that the model appears to abandon computation.

The second failure mode is *prompt-contingent infeasibility detection*: removing the $\{\}$ escape hatch causes NE accuracy to collapse for most models, confirming that their detection depends on the prompt providing an explicit mechanism to express it rather than on genuine reasoning. GPT-5.2 is more robust to this change, though even its NE accuracy degrades at $n = 30$.

The tendency to claim infeasibility also varies by notion in a consistent ordering across all models: nearly no model returns $\{\}$ for the strict core, while high rates are common for strong and super stability. At $n = 10$, GPT-5.2 largely escapes this pattern because it can genuinely solve these problems; at $n = 30$, where it can no longer do so, its NE rates follow the same difficulty-ordered pattern as the other models. This suggests that the propensity to declare infeasibility is driven by perceived problem difficulty rather than correct reasoning about whether a solution exists.

Overall, infeasibility detection that disappears without an explicit escape hatch, NE accuracy dominated by a default-to-empty bias, and a notion ordering tracking difficulty rather than logic all indicate that models lack reliable knowledge of these harder notions and adapt their outputs to whatever the prompt makes available.

Models cannot reliably identify infeasibility when selecting.

We additionally ask models to identify a specific notion among a set of candidates, a task that should be easier than generation since it only requires solution *verification* rather than executing an algorithm to arrive at a solution. As shown in Figure 12, even this task proves difficult: models frequently select incorrect candidates when a correct option exists. When NOTA is available, models use it in poorly calibrated ways. Notably, when no correct option is present, models tend to select whichever candidate satisfies a weaker version of the target notion, such as weak core instead of strict core or a weakly stable matching instead of a strongly stable one. This connects directly to the pattern observed above: the bias toward weaker notions in preference selection translates into a specific type of error when the target notion does not exist. While GPT-5.2 significantly outperforms other LLMs when the intended solution exists, its performance drops significantly when the solution does not exist - especially on larger instances, or when the NOTA option is not provided.

H.5 Selecting from Options

Models reveal systematic preference biases when selecting from candidates. When presented with a candidate set and asked for their most preferred solution, models reveal preferences that diverge from their generation behaviour (Figure 13). In allocation domains, models select utilitarian welfare-maximising outcomes far more often than canonical solutions, contrary to their default behaviour in generation. This recovers and extends the utilitarian bias while generating allocations of resources or tasks observed in previous work (Hosseini & Khanna, 2025; Shi et al., 2025; Cookson et al., 2026), showing that it also surfaces in selection and with a wider set of solution concepts. In stable matching, models favor weakly stable solutions even when stronger options are present.

Intention-action Gap. Figure 13 shows that models often end up selecting a solution labelled with a property different from the one intended.[11] This divide is higher under incomplete and tied preferences than under strict and complete preferences. In the incomplete setting, models tend to prioritize max-cardinality outcomes regardless of the notion they express an intent to target.[12] Under preferences with ties, models frequently fail to distinguish between weaker and stronger versions of the same notion, for instance selecting a weakly stable matching when a strongly stable one is among the options.[13]

[11] The notion a model "intended" to satisfy is identified using an LLM judge (Gemini-2.5-Flash); the judge prompt is provided in Appendix O.

[12] This is the one cell where the judges disagree, so we report it as judge-sensitive rather than pooling it (Appendix O.3).

[13] From an analysis of the reasoning traces, it is clear that models use the absence of (weak) blocking pairs as the primary criterion to evaluate matchings, and compare all matchings that are at least weakly stable in terms of other metrics such as welfare or Pareto-improvements.

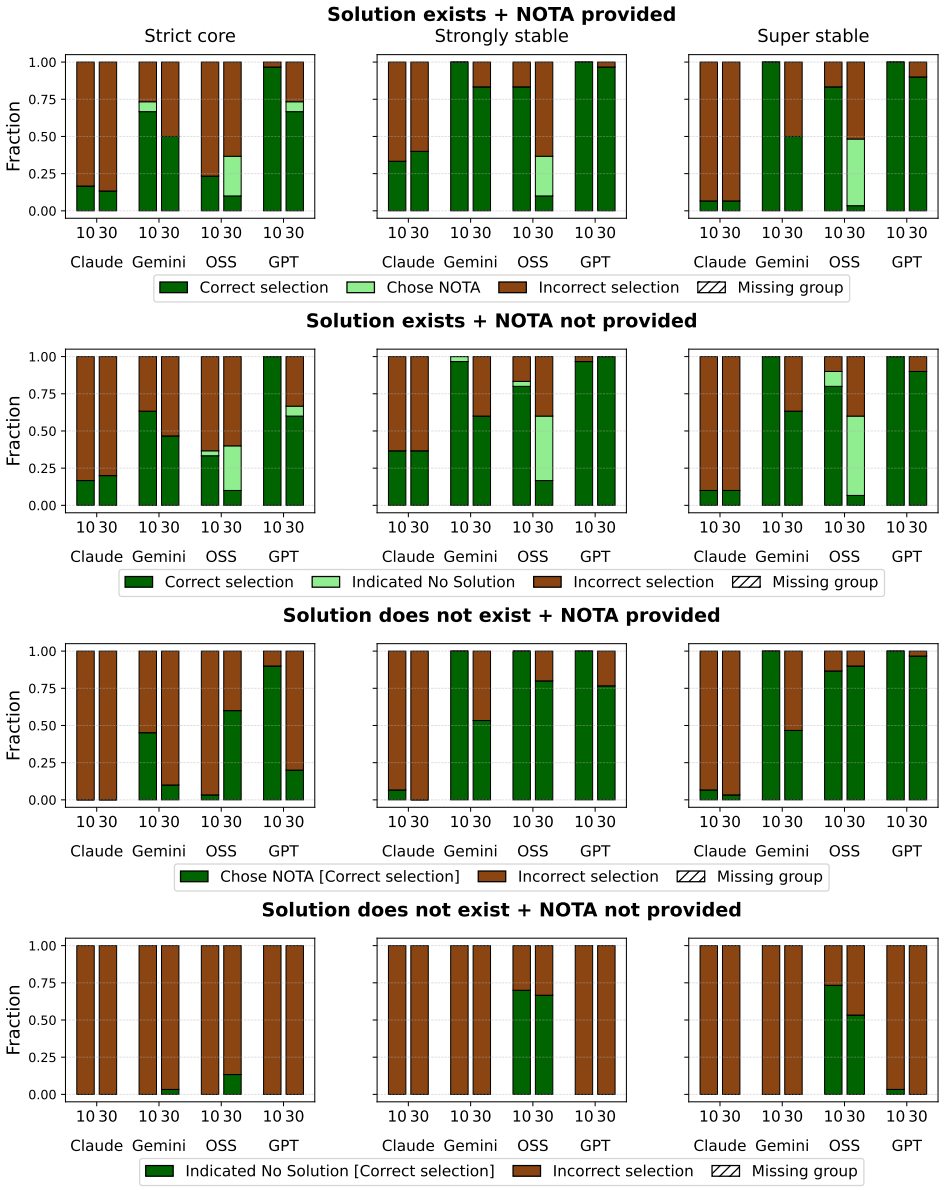


Figure 12: Selection outcomes for harder notions (strict core, strong and super stability), separated by whether a solution exists (E/NE) and whether a "None of the Above" (NOTA) option was included, at $n = 10$ and $n = 30$. Responses that timed out or contained no text are excluded.

I Statistical analysis of quantitative results

We compute 95% confidence intervals for each individual experiment, and paired tests for comparisons between problems, models and formats. Intervals are Wilson intervals for a binomial proportion. Paired comparisons on the same instances use McNemar's test, and comparisons across instance sizes use Fisher's exact test. Every analysis in this appendix runs on the responses already collected, so no additional model queries were needed. Because each prompt is scored from a single draw, these intervals are the statement of uncertainty over instances; Appendix F separately shows the results are unchanged at temperature 0 for the two models that expose it.

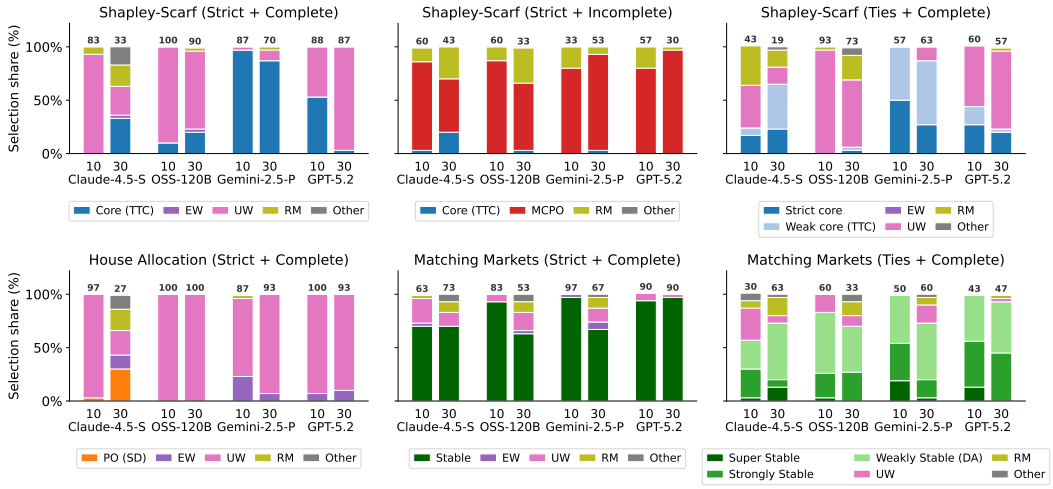


Figure 13: Share of selections for each solution concept across models, domains, and preference types, at $n = 10$ and $n = 30$. The number on top of each bar represents the percentage of responses where the *intended* notion matches the label of the selected option.

I.1 Indeterminacy gap of preference reasoning tasks

For bundle comparison and ranking tasks, the determined-vs-undetermined gap (Figure 2(a)) is robust, with determined queries having accuracy near 1 and undetermined queries at near 0. The confidence intervals do not overlap for every task and model. These are the cases where recognizing indeterminacy requires reasoning. On bundle comparison (Table 10), determined accuracy is 1.00 in eleven of the twelve (model $\times$ preference type) cells and 0.85 in the twelfth, while undetermined accuracy never exceeds 0.20.

The gap does not have this shape everywhere, and where it does not is informative. On atomic and aggregative queries the intervals overlap and the gap is model-dependent, since an unranked item is missing by inspection and recognizing it needs no reasoning step. On partial-order queries the pattern is mixed and in two cases reversed. At 150 edges (Tables 14 and 15), Claude-4.5-S scores 1.00 on determined pairs against 0.00 on undetermined ones and Gemini-2.5-P 1.00 against 0.50, whereas GPT-5.2 scores 0.17 against 0.97 and OSS-120B 0.23 against 1.00, because producing a determinate answer on a determined input is itself hard at this graph density (Appendix G.4). We therefore state the indeterminacy gap for the comparative tasks where it is clean, and report the other families per cell.

I.2 Model differences on hard notions of algorithmic reasoning

For the hard notion problems in Table 20, the model differences are significant, with all six pairwise comparisons having $p \leq 0.004$ on the McNemar's test. GPT-5.2 is the only model with non-trivial accuracy on the matched feasible instances. Table 21 reports the per-pair values.

Table 21: Hard-notion generation: pairwise McNemar tests over 349 paired instances. Per-model accuracies are given in Table 20.

Pair	p
GPT-5.2 vs. Gemini-2.5-P	< 0.001
GPT-5.2 vs. Claude-4.5-S	< 0.001
GPT-5.2 vs. OSS-120B	< 0.001
Gemini-2.5-P vs. Claude-4.5-S	< 0.001
Gemini-2.5-P vs. OSS-120B	< 0.001
Claude-4.5-S vs. OSS-120B	0.004

I.3 Scaling of input size on generation tasks

For the comparison of accuracy on small and modest market sizes in Figure 4a(a), a majority of comparisons are significant under Fisher's exact test, in 31 of 52 cells. The significance holds for models that achieve at least some level of performance on the small market, with room to decline on the modest market size.

I.4 Format effect on undetermined preference queries

For the comparison of different formats on undetermined preference queries in Figure 2, the comparison is significant on approximately half of the cases, and the effect is strongly task-dependent (Table 22), which is why we report it per cell rather than as a single average.

Table 22: Format effect (free-flow vs. MCQ) on undetermined preference queries: cells with non-overlapping 95% intervals.

Model	Significant cells
Gemini-2.5-P	11 of 20
GPT-5.2	10 of 20
Claude-4.5-S	10 of 19
OSS-120B	8 of 20

I.5 Influence of "NOTA" option on hard selection problems

The effect of the provided option in Figure 12 differs across models. When a solution exists, OSS-120B selects NOTA on 32% of instances, while the other models select it on at most 1%. On infeasible instances, all models under-select it (Table 23). Offering the option therefore induces spurious abstention for OSS-120B, and for the other models the error is a failure to abstain when abstention is correct.

Table 23: Selection with a "none of the above" option: fraction of instances on which the model selects NOTA. Responses that timed out or contained no text are excluded.

Model	Feasible (NOTA is wrong)	Infeasible (NOTA is correct)
GPT-5.2	0.01	0.80
Gemini-2.5-P	0.01	0.61
OSS-120B	0.32	0.65
Claude-4.5-S	0.01	0.03

J Prompt Templates

J.1 Preference reasoning tasks.

All preference-reasoning prompts share a common template:

```
You are an intelligent agent who is an expert in algorithms.
Consider the following instance of the object allocation
problem, where [n] alternatives have to be allocated to [n]
agents.  Given below are the preferences agents have over
the alternatives.
<preferences>
[preferences_json]
</preferences>
[task_string]
[format_string]
```

Task strings and format strings for each query type are listed in Table 24. In addition to the base free-flow prompt, for undetermined queries, additional wording is appended to permit an "unknown" or "can't decide" response. We refer to this as free-flow + "if known". Additionally, for each problem we design MCQ (multiple-choice problem) prompts for each task, with the option being potential answers, plus an option of "There is insufficient information to decide."

Table 24: Preference reasoning prompts: task and formatting instructions. The base free-flow prompt is shown, while the wordings in *italics* are added for free-flow + "if known" format.

Task string	**Format string**
What position is [choice] at in [agent]'s preference list?	Return<answer>X</answer>, where X is the number indicating the position, *if known*.
What is [agent]'s [k]th most preferred choice?	Return<answer>X</answer>, where X is the name of the choice, *if known*.
Which alternative does [agent] prefer between [choice 1] and [choice 2]?	Return<answer>X</answer>, where X is the name of the choice, *if known*.
Rank [4 choices] according to [agent]'s preference.	Return the solution in the following format: <answer>ranking</answer>, with *only ranked* choice names in descending order connected by '>' sign.
How many agents prefer [choice 1] over [choice 2]?	Return <answer>X</answer>, where X is the number of agents.
Which bundle does [agent] prefer between {[choice 1], [choice 2]} and {[choice 3], [choice 4]}?	Return the solution in the following format: <answer>{A, B}</answer>, where {A, B} is the preferred bundle. *If there is no preference, return <answer>Can't decide</answer>*.
If [agent] selects its top-[k] alternatives, does [choice] belong to this list?	Return <answer>solution</answer>with solution being yes, no *or uncertain*.
What are the top-[k] choices of [agent]?	Return <answer>solution</answer>with choice names in a list format.
Select the top-[k] choices of [agent]. Does [choice] belong to this list?	Return the solution in the following format: <answer>solution</answer>.
(Partial order) Between [choice 1] and [choice 2], which is preferred, *if known*?	Return the answer in a single word in the following format: <answer>word</answer>.

J.2 Algorithmic reasoning tasks.

Generation tasks.

```
You are an intelligent agent who is an expert in algorithms.
Consider the following instance of the [domain] problem,
where [n] alternatives have to be allocated to [n] agents.
Given below are the preferences agents have over the
alternatives.

<preferences>
[preferences_json]
</preferences>

<endowment>  (Shapley-Scarf market only)
[endowment_json]
</endowment>

Your task is to compute a [notion] allocation for the given
preferences.

[notion_definition]

Return the solution in the following format:
<answer>{"A1":  "assigned alternative", ...}</answer>
```

```
If there is no [notion] solution, return
<answer>{}</answer>.
```

Selection tasks.

```
You are an intelligent agent who is an expert in algorithms.
Consider the following instance of the [domain] problem,
where [n] alternatives have to be allocated to [n] agents.
Given below are the preferences agents have over the
alternatives.
<preferences>
[preferences_json]
</preferences>
<endowment>  (Shapley-Scarf market only)
[endowment_json]
</endowment>
Your task is to select the [notion] allocation among the
given options:
A: [allocation_json]
B: [allocation_json]
...
[K]: None of the above  (NOTA condition only)
[notion_definition]
Return only the letter of your chosen option inside <answer>
</answer> tags,
e.g., <answer>A</answer>.
```

Preference encoding. Preferences are passed to the model as JSON, with one entry per agent listing alternatives in decreasing order of preference. In Shapley-Scarf market tasks, the preferences are one-sided (agents over objects) and the endowment block lists each agent's initial holding.

```
<preferences>
{
"A1":  ["O3", "O1", "O2"],
"A2":  ["O1", "O2", "O3"],
"A3":  ["O2", "O3", "O1"]
}
</preferences>
<endowment>
{"A1":  "O1", "A2":  "O2", "A3":  "O3"}
</endowment>
```

In matching market tasks, preferences are two-sided: each side has its own preference dictionary over the other side, and no endowment block is included.

```
<preferences>
{
"A":  {
"A1":  ["B2", "B1", "B3"],
"A2":  ["B1", "B3", "B2"],
"A3":  ["B3", "B2", "B1"]
},
"B":  {
"B1":  ["A2", "A1", "A3"],
"B2":  ["A1", "A3", "A2"],
"B3":  ["A3", "A2", "A1"]
}
}
</preferences>
```

When preferences include ties, items at the same indifference level are grouped in nested lists, e.g. `["O1", ["O2", "O3"], "O4"]` represents $O_1 \succ \{O_2 \sim O_3\} \succ O_4$. Incomplete preferences omit the unranked alternatives entirely.

Reasoning trace elicitation. GPT-5.2 and OSS-120B do not expose their internal reasoning traces by default, unlike Gemini-2.5-Pro and Claude-4.5-Sonnet, which return a visible chain-of-thought alongside the final answer. To enable an analysis of the strategies being used by these models to solve the given problems, we append the following instruction to the prompt for these two models:

```
Briefly explain your approach within <scratchpad>
</scratchpad> tags after providing the answer in the above
format.
```

This instruction is appended after the answer-format specification after the conditional {} instruction if included, and otherwise after the answer format instruction.

Notion Definitions Used in Prompts Tables 25 and 26 list the exact definitions included verbatim in generation prompts for each solution concept. Definitions were held constant across all models and instance sizes.

Table 25: Definitions provided to models for Shapley-Scarf market and house allocation tasks. The prompts label this notion CORE under strict preferences and WEAK CORE under ties. Both labels carry the same blocking condition, because the two notions coincide, so we merge the rows here. The refinement that genuinely differs is the strict core, which blocks when every member of a coalition weakly improves and at least one improves strictly.

Notion	Definition
Pareto-optimal	An allocation is Pareto-optimal if there is no other feasible allocation that makes at least one agent strictly better off without making any other agent worse off.
MCPO	A max-cardinality Pareto-optimal (MCPO) allocation matches as many agents as possible to alternatives they rank, and among all such maximum-size matchings it is Pareto-optimal (no Pareto-improving reassignment exists).
Core (weak core)	An allocation is in the core if there is no coalition of agents that can reshuffle their initially endowed houses among themselves so that every coalition member is strictly better off than in the allocation.
Strict core	An allocation is in the strict core if there is no blocking coalition that can reshuffle endowed houses so that every member weakly improves and at least one member strictly improves.
UW-maximizing	A utilitarian welfare-maximizing allocation minimizes the total sum of agents' assigned ranks (equivalently, maximizes total ordinal utility).
EW-maximizing	An egalitarian welfare-maximizing allocation minimizes the worst (largest) rank any agent receives (i.e., it optimizes the welfare of the worst-off agent).
Rank-maximal	A rank-maximal allocation maximizes the number of agents receiving a 1st-choice alternative; subject to that, it maximizes the number receiving a 2nd-choice; and so on (lexicographic maximization of the rank-count vector).

K Reasoning Strategies on Feasible and Infeasible Tasks (GPT-5.2)

We restrict the qualitative analysis in this appendix to GPT-5.2. Among the four frontier models we evaluated (Claude Sonnet 4.5, Gemini 2.5 Pro, GPT-OSS 120B, and GPT-5.2), GPT-5.2 is the only model to achieve non-trivial correctness on the three solution concepts that admit both feasible and infeasible instances: the strict core of a Shapley–Scarf housing market, strongly stable matching, and super stable matching. On the feasible instances of Small size, GPT-5.2's correctness rates are 100%, 97% and 83% respectively, against runner-up rates of 17%, 70% and 23% for Gemini 2.5 Pro and at most 23% for the other two models. Other models do return correct non-existence claims on the infeasible instances, but they do so mainly by abstaining broadly, since their accuracy on the matched

Table 26: Definitions provided to models for matching market tasks.

Notion	Definition
Weakly stable	A matching is weakly stable if there is no blocking pair where both agents strictly prefer each other to their current partners; ties do not create blocking unless both sides strictly gain.
Strongly stable	A matching is strongly stable if there is no blocking pair where both agents weakly prefer each other to their current partners and at least one agent strictly prefers the other.
Super stable	A matching is super stable if there is no blocking pair where both agents weakly prefer each other to their current partners; even indifference on both sides can block.
UW-maximizing	A utilitarian welfare-maximizing matching minimizes the sum of (ordinal) partner ranks.
EW-maximizing	An egalitarian welfare-maximizing matching minimizes the maximum (worst) assigned partner rank.
Rank-maximal	A rank-maximal matching lexicographically maximizes the rank-count vector: as many agents as possible get a 1st-choice partner; subject to that, as many as possible get a 2nd-choice; and so on.

feasible instances is close to zero. GPT-5.2 is the only model that gets both sides right. We therefore focus on what GPT-5.2 actually does and how those reasoning strategies behave as the problem size grows from $n = 10$ (Small) to $n = 30$ (Modest).

The analysis below is built on GPT-5.2's own self-reports of its reasoning. Every GPT-5.2 prompt in our benchmark closes with the instruction *"Briefly explain your approach within* `<scratchpad>` `</scratchpad>` *tags after providing the answer in the above format"*, and the scratchpads it produces are the input to the qualitative coding here. To process them at scale, we combine two views: a deterministic verifier that checks each model output (it replays Top Trading Cycles with first-in-list tie-breaking, checks strong and super stability by searching for blocking pairs, and checks the strict core by a polynomial-time blocking-cycle search on the weakly-preferred-endowment digraph), and an LLM-judge (DeepSeek V4 Pro) that extracts six or seven verbatim features per scratchpad, depending on the cell: the named algorithm, any explicit trading cycle of length ≥ 3, any pre-algorithm forced-pair derivations, the stated tie-break rule, the verification step (re-checks vs. property-citation vs. none), and the final claim (allocation/matching produced vs. no-solution claim). The exact judge setup, including a representative prompt, is given in Section K.4.

The data underlying every percentage in this appendix is 698 GPT-5.2 trajectories on the generation task, 120 per $(\textit{notion}, \textit{feasibility})$ cell except the infeasible strongly-stable cell, which has 98, split between Small and Modest sizes. Within each $(\text{cell}, \text{size})$ pair the dataset contains 30 instances (28 at Small and 21 at Modest for infeasible strong stability, after removing the instances that admit a strongly stable matching; see Appendix E.2), with two trajectories per instance drawn from two prompt variants. The first variant, used for the cross-model evaluation elsewhere in the paper, explicitly instructs the model that "if no [strict core / strongly stable / super stable] solution exists, return `<answer>{}</answer>`"; we will refer to it as the *empty-hint* prompt. The second variant omits this instruction and is otherwise identical; we will refer to it as the *no-hint* prompt. The two variants produce strongly divergent behaviour on infeasible cells. Under the no-hint prompt GPT-5.2 never returns $\{\}$ on any of the 169 infeasible-cell trajectories issued under that variant; instead, in 65 of them it maps every agent to null, and in all but one of these its scratchpad argues that no solution exists. We therefore split per-cell statistics by prompt variant whenever the difference matters, and count an all-null answer as a no-solution claim. Some requests timed out; we report these separately, since a timeout is not a behaviour of the model's reasoning.

K.1 Strict core in Shapley–Scarf housing markets

Feasible instances. On feasible strict-core instances, GPT-5.2 invariably runs Top Trading Cycles (TTC). What varies is the tie-break rule, and on this dataset the rule chosen determines correctness

almost entirely. The successful pattern is what we call the *constructive chain heuristic*: at each indifference, the model picks the tied alternative that allows a long trading cycle to close, rather than the lexicographically first tied alternative. The choice is occasionally verbalised, as in this Small-size trace:

> *Approach: 1) Compute a core candidate via Top Trading Cycles (TTC) adapted to ties by selecting one "top" house per agent when indifferent. Running TTC on the endowment graph and choosing A_7's top pointer as C_4 (rather than keeping C_{10}) yields the allocation above. 2) Verify the strict-core condition by searching for a blocking coalition: any blocking reshuffle decomposes into trading cycles, so it suffices to check whether there is a directed cycle of agents $i_1 \to i_2 \to \cdots \to i_k \to i_1$ in which each i_t weakly prefers the endowment of i_{t+1} to its assigned house and at least one strictly prefers it. Building this digraph for the allocation above yields no such cycle, hence no blocking coalition exists.*

We label a trajectory as constructive-chain if the model's allocation differs from canonical first-in-list TTC *and* at least one agent whose own endowment lies in its top indifference class receives a different house from that class (the verifier produces the canonical TTC output for each instance). At Small size, 30 of 60 trajectories (50%) are constructive-chain, and 29 of those 30 (97%) land in the strict core. A residual non-mechanical bucket ("arbitrarily but consistently" chosen ties, in the model's own words) accounts for another 26 trajectories of which 24 (92%) are also in the core. The empty-hint clause provides a small lift: $30/30$ correct under the empty-hint prompt versus $23/30$ (77%) under the no-hint prompt.

At Modest size the picture inverts. The constructive-chain stratum collapses to 2 of 60 (3%), neither correct; 30 of 60 trajectories (50%) match canonical first-in-list TTC exactly, none of which is in the strict core; the residual non-mechanical bucket falls to 5 correct of 28 (18%). GPT-5.2 abandons the constructive heuristic in favour of a mechanical rule that the TTC theorem does not justify under weak preferences, and that empirically never recovers the strict core on these instances.

Infeasible instances. On infeasible instances, the only correct strategy is to argue that no allocation can satisfy the strict-core condition. GPT-5.2 does this via what we call *iterative mandatory-pair enumeration*: a sequence of forced assignments, each justified by exhibiting a coalition that would block any allocation violating it, terminating in a contradiction. A representative trace:

> *Approach: show non-existence by exhibiting a blocking coalition for every possible allocation. If an allocation x gives A_8 anything other than C_2, then coalition $\{A_8, A_5, A_9, A_{10}\}$ blocks x via $A_8 \leftarrow C_2$ (top choice, strict improvement), $A_5 \leftarrow C_5$, $A_9 \leftarrow C_1$, $A_{10} \leftarrow C_7$ (each weakly improves). So any strict-core allocation must assign C_2 to A_8. If an allocation gives A_4 anything other than C_2, coalition $\{A_4, A_5, A_9\}$ blocks similarly, so any strict-core allocation must also assign C_2 to A_4. But C_2 is a single house, so the strict core is empty.*

Whether the model produces this argument at all is controlled almost entirely by the prompt variant. At Small size under the empty-hint prompt, 24 of 30 trajectories (80%) explicitly claim no allocation exists, and all 24 are correct; under the no-hint prompt, 5 of 30 (17%) map every agent to null and argue in the scratchpad that no strict-core allocation exists. The mandatory-pair argument is in the model's repertoire, but it surfaces far more often when the prompt explicitly mentions the empty-answer option. At Modest size, the empty-hint prompt no longer suffices: 0 of 30 trajectories under either variant claim non-existence. The forced-pair chains required at $n = 30$ are longer and the blocking coalitions larger, and even with explicit permission to return $\{\}$, the model commits to producing an allocation rather than working through the chain.

K.2 Strongly stable matchings

Feasible instances. The strongly-stable task is set on a bipartite instance with two sides of agents, denoted A and B. On feasible instances, GPT-5.2 pairs deferred acceptance with hand-coded local repair: it first identifies one or two forced pairs from the preference structure (for instance, an (A_i, B_j) where each side is the unique top choice of the other), runs A-proposing deferred acceptance on a

tie-refinement to populate the matching, scans the result for strong-blocking pairs created by the indifferences that the tie-refinement collapsed, and applies short local swaps until none remain. A representative trace:

> *Approach (strong stability with ties): I first constructed a complete matching via an A-proposing deferred-acceptance run (breaking ties arbitrarily just to get a candidate), then explicitly checked the strong-stability condition. The initial DA matching was not strongly stable: B_5 ranks A_4 as its unique top choice, while A_4 ranks B_5 in its 2nd tier (tied with B_6). If A_4 is not matched to B_5 and not to its strictly-better option B_8, then (A_4, B_5) is a strong blocking pair. This forces $A_4 - B_5$. After enforcing $A_4 - B_5$, I iteratively repaired the matching to remove remaining strong blocking pairs caused by indifferences. The final matching is strongly stable.*

At Small size, 45 of 60 trajectories (75%) produce a strongly stable matching and 11 more produce a super-stable matching (which strictly implies strong stability), for a combined $56/60$ (93%). Pre-algorithm forced-pair reasoning is the most reliable distinguishing feature of correct trajectories.

The approach itself does not change at Modest size, but its execution quality does. Strong stability is met by $15/60$ (25%) and super-stability by $4/60$, for a combined $19/60$ (32%); 25 trajectories (42%) produce no matching (12 return $\{\}$ under the empty-hint prompt, 4 map every agent to null under the no-hint prompt, and 8 requests timed out), and 13 produce a matching that is not even weakly stable. The local repair loop is the bottleneck: as the matching grows, every swap to fix one strong-blocking pair tends to introduce another. Notably, the empty-hint clause *hurts* on this cell at Modest size: $7/30$ (23%) correct under the empty-hint prompt versus $12/30$ (40%) under the no-hint prompt. The most plausible reading is that the explicit $\{\}$ option, when offered on a feasible instance the model is struggling to solve at scale, occasionally tempts it to retreat to the empty answer rather than persist with the local-repair loop.

Infeasible instances. On infeasible strongly-stable instances, GPT-5.2 produces the matching-theoretic analogue of the mandatory-pair impossibility chain, again starting from a forced pair (typically a mutual top tie) and deriving subsequent forced assignments that culminate in an unavoidable strong-blocking pair:

> *Approach (proof of non-existence by forced blocking pairs). A_5 has a top tie $\{B_4, B_6\}$. If A_5 is matched to B_6, then (A_5, B_4) blocks: A_5 weakly prefers B_4 to B_6 (indifferent), and B_4 ranks A_5 in its top tie (with A_2), so B_4 weakly prefers A_5 to any partner. To avoid this, B_4 must be matched to A_2. But then (A_2, B_2) blocks: A_2 strictly prefers B_2 to B_4, and B_2 ranks A_2 as its top choice. Therefore A_5 cannot be matched to B_6. ... Each branch leads to an unavoidable strong-blocking pair, so no strongly stable matching exists.*

As with the strict-core infeasible cell, the rate at which the model produces this argument depends on the prompt variant. Under the empty-hint prompt: $19/28$ (68%) at Small and $13/21$ (62%) at Modest return $\{\}$ with such an argument. Under the no-hint prompt, the model maps every agent to null in $18/28$ (64%) at Small and $6/21$ (29%) at Modest, almost always with the same kind of argument in the scratchpad. Timeouts account for most of the remaining trajectories (8 and 4 under the empty-hint prompt, 6 and 12 under the no-hint prompt). All 32 explicit non-existence claims under the empty-hint prompt are correct. The size behaviour of the empty-hint trajectories is also notable: the rate is essentially size-invariant ($68\% \to 62\%$), in contrast to the feasible cell of this same notion (where Modest correctness drops from 93% to 32%).

K.3 Super stable matchings

Feasible instances. On feasible super-stable instances, GPT-5.2 names the Irving–Manlove super-stable matching algorithm (often as "Irving SUPER" or "the standard super-stable SMTI reduction") and at Small size applies it convincingly:

> *Approach: I applied the standard super-stable SMTI reduction procedure (Irving–Manlove style): repeatedly let free A-agents propose to everyone in their current*

> *first (possibly-tied) tier; each B-agent keeps the best (possibly-tied) proposers seen so far and deletes from their list everyone strictly worse than the kept tier; deletions are mirrored on the A-side. If any agent's list becomes empty, no super-stable matching exists. … This pruning forced several pairs (B_2 can only take A_1, B_5 only A_8, B_7 only A_2), reducing the instance to a small feasible edge set. From the remaining edges I constructed a perfect matching and validated super-stability directly.*

At Small size, 49 of 60 trajectories (82%) produce a super-stable matching; of the remaining 11, 10 are requests that timed out and one returns $\{\}$. No Small trajectory produces a well-formed matching that fails super-stability.

At Modest size, the picture changes qualitatively. Only $1/60$ (1.7%) produces a super-stable matching; 52 (87%) produce no matching, and 7 produce a well-formed matching that fails super-stability (4 are not even weakly stable, 3 are strongly stable but not super-stable). Most of the 52 are false claims that no super-stable matching exists: 28 return $\{\}$ under the empty-hint prompt and 18 map every agent to null under the no-hint prompt; 3 requests timed out. The model continues to invoke Irving–Manlove by name, but at this size it mostly concludes, incorrectly, that the instance is infeasible.

Infeasible instances. On infeasible super-stable instances, GPT-5.2 produces either a forced-pair impossibility chain or an Irving-style certificate in which it runs the edge-deletion procedure and reports that some agent's list became empty. A typical Irving-style certificate runs:

> *The reductions force $A_1 - B_9$ (mutual top), A_9 must be with B_1 (deletions eliminate all other feasible partners), and B_{10} ends up only feasibly matchable with A_2. But then the pair (A_9, B_{10}) is unavoidable as a blocking pair: A_9 is indifferent between B_1 and B_{10}, so A_9 weakly prefers B_{10} to B_1; B_{10} is indifferent between A_9 and A_2, so B_{10} weakly prefers A_9 to A_2. Therefore the instance admits no super stable matching.*

As on the other infeasible cells, the rate depends on the prompt variant. Under the empty-hint prompt: $20/30$ (67%) at Small and $29/30$ (97%) at Modest, all 49 correct. The apparent increase with size is an effect of timeouts: 9 of the 30 Small requests timed out, and 20 of the 21 that completed claim non-existence. Under the no-hint prompt, the model maps every agent to null, with an impossibility argument, in $13/30$ at Small (15 requests timed out) and $23/30$ at Modest. At Modest size the model also claims non-existence on 46 of the 60 feasible super-stable trajectories, so the high rate on infeasible instances does not by itself show that it distinguishes the two.

Discussion. A consistent picture emerges. On feasible cells, GPT-5.2 succeeds at Small size by deploying non-mechanical heuristics that exploit instance-specific structure: the constructive chain heuristic for strict core, deferred acceptance with forced-pair pre-processing for strongly stable, and Irving–Manlove edge-deletion for super stable. As the instance scales from $n = 10$ to $n = 30$ each of these heuristics degrades, and the rate of trajectories that return no matching rises sharply, mostly through false claims that no solution exists. The model does not have a fallback strategy that degrades gracefully: it has only the same heuristics applied with less fidelity, plus an increasing tendency to declare the instance infeasible.

On infeasible cells, GPT-5.2 produces correct impossibility arguments on a substantial fraction of instances under the empty-hint prompt (80% on strict-core Small, $68\%/62\%$ on strongly-stable, $67\%/97\%$ on super-stable). Without the hint it never returns $\{\}$, but in 65 of 169 infeasible-cell trajectories it maps every agent to null and argues in the scratchpad that no solution exists. The forced-pair impossibility chain is therefore in the model's repertoire under both prompts; the explicit empty-answer option makes the model reach the conclusion more often and fixes the form in which it reports it. The strict-core chain collapses at Modest under both prompts, because the chains are longer and the coalitions larger. On the matching cells at Small size many requests timed out, so the rates there are lower bounds. The practical implication is that infeasibility-handling rates on benchmarks like ours depend on the prompt phrasing and on which answer formats the scoring accepts as a no-solution claim, and care must be taken not to conflate either with model capability.

K.4 LLM-JUDGE SETUP

The LLM-judge stage extracts structured features from each scratchpad without itself doing any algorithmic verification. Its output is then joined to the deterministic verifier's stratum labels for the per-cell tables above. We use DeepSeek V4 Pro as the judge with temperature 0 and JSON-mode response formatting. Each judge call sees: (i) the original problem statement that the model was given, (ii) the model's scratchpad, (iii) the model's parsed final answer, (iv) a fixed list of per-cell open probes asking for verbatim quotes of specific trace features, and (v) a canonical-strategy check that asks whether the trace follows a notion-specific reference strategy (yes / partial / no). The probe fields and the canonical-strategy descriptions were defined in advance from a preliminary inspection of $\sim$50 scratchpads and were held fixed during the full run. The judge is instructed to produce verbatim quotes wherever feasible, to avoid inventing content that is not in the scratchpad, and to return `null` or `[]` for fields with no relevant content.

The full prompt template, with placeholders shown in {*italic*}, is:

> **System.** You are an expert annotator analyzing reasoning traces from a language model solving combinatorial matching and allocation problems (housing markets / stable matching with ties). For each trace you will answer a set of open-ended probe questions about objective features of the trace, and then a canonical-strategy check. Always output a single JSON object and nothing else. Use verbatim quotes from the scratchpad where requested. Do not invent content not present in the scratchpad.
>
> **User.**
> PROBLEM CONTEXT (the prompt the model was given):
> """ *prompt_text* """
>
> MODEL'S REASONING TRACE (scratchpad):
> """ *scratchpad* """
>
> MODEL'S FINAL ANSWER (parsed from `<answer>` tag):
> """ *answer* """
>
> PROBE QUESTIONS:
> (1) FIELD `algorithm_name`: Verbatim name(s) of the algorithm the model invokes (e.g. "Top Trading Cycles", "Gale–Shapley"). If none stated, write null. Maximum 8 words.
> (2) FIELD `cycles_shown`: List of explicit cycles or rounds shown in the scratchpad, each as a chain string of the form "$A_i \to C_j \to A_k \to \cdots \to A_i$". If none, write `[]`.
> (3) FIELD `tie_break_rule`: Verbatim quote of the sentence(s) describing how ties within indifference classes are broken. If unspecified, null.
> (4) FIELD `verification_text`: Verbatim quote of the sentence(s) where the model verifies the result beyond a generic property-citation. If only a generic property-citation is present, write the exact string `PROPERTY_CITATION_ONLY`. If none, null.
> (5) FIELD `forced_assignments_pre_algorithm`: List of any "agent must be assigned house" deductions stated before the main algorithm is run, each with a quoted reason. If none, `[]`.
> (6) FIELD `final_claim`: One of `allocation_produced`, `claims_no_solution`, `abstains_or_refuses`. Quote the closing sentence of the scratchpad.
>
> CANONICAL STRATEGY CHECK. Indicate whether the trajectory follows the canonical strategy below: "yes" (clearly and substantively), "partial" (some elements but deviates in others), or "no". Quote 1–2 verbatim sentences as evidence. If partial or no, briefly note in 1 sentence what differs.
> Canonical strategy: *strategy_name*.
> Description: *strategy_description*.
>
> OUTPUT JSON OBJECT WITH EXACTLY THESE FIELDS:

```
{
  "algorithm_name": <answer>,
  "cycles_shown": <answer>,
  "tie_break_rule": <answer>,
  "verification_text": <answer>,
  "forced_assignments_pre_algorithm": <answer>,
  "final_claim": <answer>,
  "canonical_strategy_match": "yes" | "partial" | "no",
  "canonical_strategy_evidence": "<verbatim quote(s)>",
```

```
  "canonical_strategy_deviation":  "<one sentence or null>"
}
```

The probe questions vary slightly across the six (*notion*, *feasibility*) cells (for instance, the matching cells substitute "forced pairs" for "trading cycles" in probe (2)), but the system prompt, the output-schema requirements, and the canonical-strategy check template are shared across all cells. The complete cell-specific probe definitions are released alongside the code.

L PROMPT-LEVEL MITIGATION

If the epistemic failure were a surface prompting artifact, an instruction warning the model about indeterminacy should remove it. We test this with four levels of prompt mitigation on the bundle comparison problem, run on both RS-comparable and RS-incomparable cases. A genuine fix has to keep accuracy high on both, since a deployed user does not know in advance which kind of query they are issuing.

- Level 1 bare, no hint of indeterminacy.
- Level 2 multiple choice with an explicit "not enough information" option.
- Level 3 caution. general = source-agnostic ("preferences may not entail an answer, do not assume one"). specific = names the structure being tested.
- Level 4 few-shot demonstration ending in "cannot be determined"

Table 27 shows that no prompt calibrates the model. Every intervention that raises undetermined accuracy also lowers accuracy on the determined queries: the general caution moves GPT-5.2 from $1.00 \mid 0.00$ to $0.43 \mid 0.80$ and Gemini-2.5-P from $1.00 \mid 0.00$ to $0.47 \mid 0.93$. The error is relocated, not removed.

Of the prompts tried, no single prompt is safe across models. The specific caution lifts Claude and Gemini but collapses OSS-120B to $0.27 \mid 0.93$, and the explicit abstain option is ignored by three of four models, which keep imposing the lexicographic default.

Few-shot demonstrations raise undetermined accuracy sharply when they come from the same source of indeterminacy, but the gain does not transfer. When the bundle comparisons are instead given demonstrations about a different source (an item the agent never ranked), Table 28 shows they help almost not at all. The model reproduces the pattern it was shown rather than acquiring a general ability to flag indeterminacy.

This is why the failure is not a surface prompting artifact. Prompting moves the model along a trade-off between over-commitment and over-abstention, and a caution aimed at one kind of indeterminacy does not carry over to another. This matters in practice, because a deployed user does not know in advance whether a query is undetermined, or which kind of indeterminacy it involves, so they cannot supply the matching caution beforehand. The cross-setting experiment tests this. When the demonstrations describe a different source than the query, performance falls back to the baseline, showing the model has not learned a general ability to recognize indeterminacy but only to imitate the case it was shown.

Table 27: Accuracy on bundle comparison problems, with different levels of prompt mitigation. Levels: 1 bare, 2 MCQ with an explicit "not enough information" option, 3 caution (general = source-agnostic, specific = names the structure), 4 few-shot demos. Each is run on 30 instances, with accuracy reported as determined | undetermined

Model	L1 bare	L2 MCQ	L3 general	L3 specific	L4 few-shot
GPT-5.2	$1.00 \mid 0.00$	$0.97 \mid 0.00$	$0.43 \mid 0.80$	$0.73 \mid 0.27$	$0.93 \mid 0.10$
Claude-4.5-S	$1.00 \mid 0.00$	$1.00 \mid 0.00$	$0.97 \mid 0.57$	$1.00 \mid 0.83$	$1.00 \mid 0.97$
Gemini-2.5-P	$1.00 \mid 0.00$	$1.00 \mid 0.00$	$0.47 \mid 0.93$	$0.97 \mid 0.50$	$1.00 \mid 0.90$
OSS-120B	$1.00 \mid 0.00$	$0.83 \mid 0.37$	$0.73 \mid 0.63$	$0.27 \mid 0.93$	$0.97 \mid 0.83$

Table 28: Accuracy on the undetermined bundle comparisons of Table 27, with few-shot demonstrations. Levels: L1 bare, L4 few-shot demos (in-domain = examples about bundle comparisons; cross-domain = examples about the other source of indeterminacy, an item the agent never ranked). Each is run on 30 instances.

Model	L1	L4 in-domain	L4 cross-domain
GPT-5.2	0.00	0.10	0.13
Claude-4.5-S	0.00	0.97	0.00
Gemini-2.5-P	0.00	0.90	0.00
OSS-120B	0.00	0.83	0.00

Table 29: Code-based task accuracy by domain, problem size, and feasibility (F = feasible and I = infeasible). Each cell shows the percentage of correct responses out of 10 instances. Dashes indicate conditions not evaluated. Timeouts and invalid responses are counted as incorrect.

	Shapley-Scarf Market						Matching Market					
	Small		Modest		Medium		Small		Modest		Medium	
Model	F	I	F	I	F	I	F	I	F	I	F	I
Gemini-2.5	10%	0%	10%	0%	–	–	20%	70%	0%	60%	–	–
Claude-4.5-S	100%	100%	0%	70%	–	–	90%	100%	30%	100%	–	–
GPT-5.2	100%	100%	90%	70%	100%	30%	100%	100%	90%	90%	0%	10%

M Code-Assisted Reasoning: Approaches by Model and Size

The accuracy achieved by LLMs on algorithmic reasoning questions, when using code to solve the problem, are provided in Table 29.

This section summarises the dominant strategy each model produces in the code-assisted condition, broken down by notion and instance size. Three patterns recur. First, at $n = 10$, GPT-5.2 and Claude Sonnet 4.5 follow a common verify-then-fall-back pattern: the generated code first attempts the textbook algorithm for the notion (Top Trading Cycles, deferred acceptance, or Irving's super-stable algorithm), checks whether the resulting object satisfies the notion's condition, and on a failed check falls back to exhaustive enumeration over all candidate allocations or matchings combined with a complete blocking-coalition or blocking-pair check. The fallback is sound at $n = 10$ because the search space is small enough to enumerate. Second, at $n = 30$ and above the same models switch to heuristic strategies (randomised tie-breaking with verification, backtracking with constraint propagation, exponential-weight reductions to maximum-weight matching), and the soundness of the fallback degrades as size grows. Third, Gemini 2.5 Pro is the outlier: it does not use exhaustive enumeration at any size, and its code typically applies a textbook algorithm directly with no verification step.

M.1 GPT-5.2

GPT-5.2's code shows the cleanest version of the verify-then-fall-back pattern. On strict core with ties at $n = 10$, it first runs Top Trading Cycles (TTC) with first-in-list tie-breaking, verifies the output against a complete blocking-coalition oracle, and falls back to exhaustive enumeration of all $n!$ allocations on a verification failure, returning $\{\}$ when the enumeration finds no allocation in the strict core. At $n = 30$ the model switches to randomised TTC: it repeats TTC with random tie-breaking until verification succeeds. This is sound on feasible instances and remains correct on all $n = 50$ feasible instances we evaluated, but it cannot prove non-existence, so on infeasible $n = 30$ instances the model occasionally falls back to a MILP encoding of the strict-core polytope, with mixed soundness. At $n = 50$ infeasible instances the randomised loop times out before any code completes. On super-stable matching, GPT-5.2 progresses from exhaustive enumeration at $n = 10$, to backtracking with constraint propagation at $n = 30$, to a faulty implementation of Irving's algorithm SUPER at $n = 50$ that omits the step of breaking all engagements of multiply-engaged agents, and consequently produces false NE claims on every $n = 50$ instance.

Table 30: GPT-5.2: dominant approach in the code-assisted condition, by task and instance size. For notions with feasibility variation, E and NE rows are shown separately where approaches differ.

Notion	n	E/NE	Dominant approach
Core (HA, strict)	10–50	–	Top Trading Cycles (correct)
Strict Core (HA, ties)	10	E/NE	TTC attempted; on verification failure, falls back to exhaustive enumeration with complete blocking-coalition check (sound)
	30	E	Randomised TTC: repeat with random tie-breaking until a strict-core allocation is found and verified
	30	NE	Heuristic search failure; MILP in some responses; not always sound
	50	E	Randomised TTC; all instances correct
	50	NE	Timeout (no result produced)
Rank-Max. Alloc. (HA, ties)	10	–	Brute-force enumeration over all allocations
	30–50	–	Exponential-weight reduction to max-weight matching (one-shot; empirically correct)
Super Stable (SM, ties)	10	E/NE	Gale-Shapley variant attempted; on verification failure, falls back to exhaustive enumeration with complete super-blocking-pair check (sound)
	30	E/NE	Backtracking with constraint propagation (CSP); mostly correct
	50	E/NE	Algorithm SUPER attempted but incorrectly implemented: omits breaking all engagements of multiply-engaged agents, causing false NE claims on all instances
Rank-Max. Match. (SM, ties)	10–30	–	Exponential-weight reduction (one-sided A-side only); misunderstands two-sided definition

M.2 CLAUDE SONNET 4.5

Claude Sonnet 4.5 follows the same overall verify-then-fall-back pattern at $n = 10$ as GPT-5.2 (exhaustive enumeration of allocations or matchings, with complete blocking checks), but its heuristic strategies at $n = 30$ are more limited. On strict core with ties at $n = 30$, the generated code searches a random sample of allocations and verifies them against blocking coalitions of size ≤ 5 only. This yields low accuracy on feasible instances but accidentally high accuracy on infeasible ones, since a small blocking coalition is usually found quickly. On super stable at $n = 30$, the code runs Gale-Shapley with random tie-breaking and post-hoc super-stability verification, which gives partial accuracy on feasible instances and again finds an obstruction quickly when none exists. On rank-maximal matching, Claude's code (like GPT-5.2's) operates on a one-sided ranking only, indicating that both models misread the two-sided definition.

M.3 GEMINI 2.5 PRO

Gemini 2.5 Pro does not use exhaustive enumeration at any size, and its code typically lacks an explicit verification step. On strict core with ties, it runs TTC with an arbitrary tie-breaking rule and treats the output as the answer regardless of whether it lies in the strict core, which means it never identifies infeasible instances. On super stable, the code is an ad-hoc modification of Gale-Shapley with no super-stability check, and at $n = 30$ Gemini has a high rate of false NE claims regardless of ground-truth feasibility. On rank-maximal matching at $n = 30$, Gemini's code is a greedy per-rank algorithm that achieves 0% accuracy. The textbook algorithm for the simplest notion in this study (Core in strict-preference housing markets) is correctly implemented, with two cycle-detection bugs at $n = 30$.

N REFINEMENT EXPERIMENT DETAILS

Setup. The refinement experiment tests whether LLMs can correct their answers when given verification-based feedback. We focus on two solution notions that require generating valid allocations or matchings:

- **Strict core** (Shapley–Scarf housing market): agents own one house each and can trade. A valid strict core allocation has no group of agents (coalition) that could redistribute their

Table 31: Claude-4.5-Sonnet: dominant approach in the code-assisted condition, by task and instance size.

Notion	n	E/NE	Dominant approach
Core (HA, strict)	10–30	–	Top Trading Cycles (correct; minor cycle-detection bugs fixed in later iterations)
Strict Core (HA, ties)	10	E/NE	TTC attempted; on verification failure, falls back to exhaustive enumeration with full blocking-coalition verification (sound)
	30	E	Heuristic random allocation search with limited coalition-size checks (≤ 5); low accuracy
	30	NE	Heuristic: blocking coalition found quickly; mostly correct
Rank-Max. Alloc. (HA, ties)	10	–	Mixed: some exhaustive enumeration, some greedy rank-by-rank
	30	–	Greedy rank-by-rank fixation (incorrect; 0% accuracy)
Super Stable (SM, ties)	10	E/NE	Gale-Shapley variant attempted; on verification failure, falls back to exhaustive enumeration of all matchings with super-blocking-pair check (sound)
	30	E	Random Gale-Shapley tie-breaking with post-hoc verification (heuristic; partial accuracy)
	30	NE	Heuristic failure after many random trials; often correct
Rank-Max. Match. (SM, ties)	10–30	–	Iterative rank addition with one-sided matching; misunderstands two-sided definition

Table 32: Gemini-2.5: dominant approach in the code-assisted condition, by task and instance size.

Notion	n	E/NE	Dominant approach
Core (HA, strict)	10–30	–	Top Trading Cycles (correct; two failures at $n = 30$ due to cycle-detection bugs)
Strict Core (HA, ties)	10–30	–	TTC with arbitrary tie-breaking and no verification step (incorrect; never identifies infeasible instances)
Rank-Max. Alloc. (HA, ties)	10–30	–	Greedy sequential maximum matching per rank (incorrect heuristic; near-zero accuracy at $n = 30$)
Super Stable (SM, ties)	10	E	Custom iterative elimination or modified Gale-Shapley variants
	10	NE	Heuristic failure or custom elimination; partially correct
	30	E/NE	Ad-hoc Gale-Shapley modifications; high rate of false NE claims regardless of ground truth
Rank-Max. Match. (SM, ties)	10	–	Exponential-weight Hungarian (one-sided; misunderstands two-sided definition)
	30	–	Greedy iterative per-rank matching (incorrect heuristic; 0% accuracy)

endowed houses among themselves so that every member weakly improves and at least one strictly improves.

- **Super stable matching** (bipartite matching market): agents on two sides are matched one-to-one. A super stable matching has no blocking pair—a pair of agents who both weakly prefer each other to their current partners.

For each notion, we include 30 feasible instances (a valid solution exists) and 30 infeasible instances (no valid solution exists), for a total of 60 instances per model per notion. We evaluate two models: OSS-120B on Small instances ($n = 10$) and GPT-5.2 on Modest instances ($n = 30$). All instances use the Generation task format, where the model must produce a JSON allocation or matching (or `{}` if no solution exists).

Interaction protocol. Each instance is evaluated using a multi-turn conversation with up to three attempts:

1. **Attempt 1**: The model receives the original task prompt and produces an answer.

2. **Attempt 2** (if Attempt 1 is wrong): The model receives feedback explaining why its answer is incorrect, then produces a new answer. The full conversation history is preserved.
3. **Attempt 3** (if Attempt 2 is also wrong): The model receives another round of feedback and makes a final attempt, again with full history.

If any attempt produces a correct answer, the instance is marked as a success and no further attempts are made. If any attempt times out or produces an error, the remaining attempts are skipped.

Feedback design. The feedback message is constructed automatically by running the same programmatic verifier used to score the initial experiments. The content of the feedback depends on what the model got wrong.

When the model returns `{}` on a feasible instance (incorrectly claiming no solution exists):

> *"Your answer is incorrect. A valid [notion] solution does exist for this instance, but you returned an empty allocation. You have [N] attempts remaining. Please try again carefully and return your answer enclosed in `<answer> </answer>` tags."*

This tells the model that a solution exists without revealing what it is.

When the model returns a wrong allocation for strict core:

The verifier computes all blocking coalitions and reports them. For example:

> *"Your answer is incorrect. Your allocation has 2 blocking coalitions under the strict core. Here are the blocking coalitions: {A2, A10, A5, A8, A3, A9, A7}; {A2, A10, A5, A9, A7}. Each coalition can reshuffle their endowed items among themselves so that every member weakly improves and at least one strictly improves. You have [N] attempts remaining."*

The feedback includes:

- The total number of blocking coalitions.
- The members of each coalition (up to 3 coalitions are shown; if there are more, a note indicates the total).
- A plain-language explanation of what a blocking coalition means.

When the model returns a wrong matching for super stable:

The verifier computes all blocking pairs and reports them. For example:

> *"Your answer is incorrect. Your matching has 5 blocking pairs under super stability. Here are the blocking pairs: (A1, B10), (A3, B10), (A4, B10), (A4, B9), (A8, B9). In each pair, both agents weakly prefer each other to their current partners. You have [N] attempts remaining."*

The feedback includes:

- The total number of blocking pairs.
- The specific pairs (up to 5 are shown; if there are more, a note indicates the total).
- A plain-language explanation of the blocking condition.

When the model's response cannot be parsed as valid JSON:

> *"Your answer could not be parsed as a valid JSON allocation. Make sure you return a JSON object mapping each agent to an assigned alternative, enclosed in `<answer> </answer>` tags. You have [N] attempts remaining."*

Key design choices.

- **Conversation history is preserved.** Each attempt is sent as a new message in the same multi-turn conversation. The model sees its own prior responses and the feedback it received, without needing to re-read the original problem description.
- **Feedback is informative but not prescriptive.** The feedback identifies specific violations (which coalitions block, which pairs block) but does not suggest how to fix the answer. The model must figure out the repair strategy on its own.
- **Verification is programmatic.** Correctness is determined by the same algorithmic verifiers used in the main experiments—not by string matching or LLM-based judging. For strict core, the verifier exhaustively checks all possible coalitions. For super stability, it checks all agent pairs for the weak blocking condition.
- **Feasibility feedback is asymmetric.** When an instance is feasible but the model returns { }, the feedback reveals that a solution exists. When an instance is infeasible and the model returns a wrong allocation, the feedback shows the blocking coalitions or pairs—which may indirectly signal that no valid solution exists (e.g., seeing hundreds of blocking coalitions suggests infeasibility). However, the feedback never explicitly states that no solution exists.

Results. Figure 14 shows the outcome of the refinement experiment. Refinement is effective only when the model produces a candidate solution to begin with. When its dominant failure mode is over-abstention, as on matching markets, additional rounds do not change the outcome.

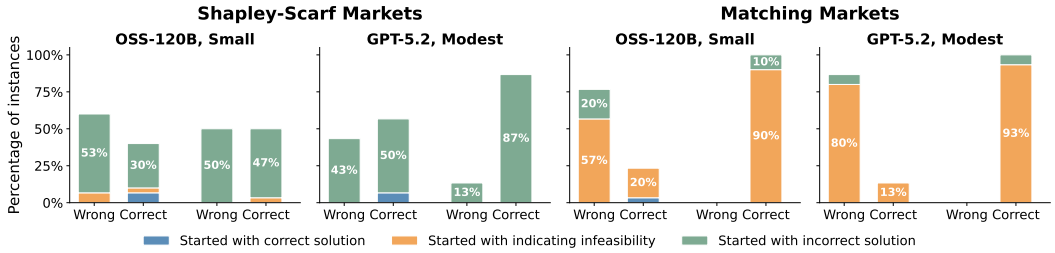


Figure 14: The effect of allowing models to refine their solutions using feedback. Each bar shows the fraction of instances that were ultimately answered correctly or incorrectly after up to three attempts with verification feedback.

O LLM-Judge Analysis Details

We use a separate LLM-as-judge pass to label the reasoning portion of each response, i.e., which algorithm or solution concept the responding model appears to have invoked, independently of whether the final allocation is correct. We run this in two settings. The **generation** setting classifies the algorithm a model used to produce a solution when no specific target notion is provided. The **selection** setting classifies the criterion a model prioritized when picking among several candidate solutions. The two settings share the same judge model, prompt scaffolding, and answer format; they differ in the option set and in how "intended" and "achieved" are scored.

Judge configuration. All judging in this appendix is done with Gemini-2.5-Flash (the reasoning-strategy analysis of Appendix K uses a separate judge, described there) through the `google-genai` Python client, using the client's default sampling parameters (no temperature, top-p, or other generation arguments are set). The judge is shown two pieces of context: the original prompt that was given to the model under study, and that model's full response. It is asked to select a single option from a closed list and return its choice in the format `<answer>X</answer>`. The judge is explicitly instructed *not* to verify the algorithm or criterion by computing solutions itself; its label must be based only on explicit textual mentions or descriptions in the response. If no listed option matches, it must return the catch-all "no algorithm mentioned"/"some other notion" option. Each judge call is wrapped in a three-attempt retry to absorb transient API failures, and the parsed letter is post-processed by stripping the `<answer>...</answer>` tags; we additionally fall back to extracting the letter inside `\boxed{...}` when the judge returns its answer in LaTeX form.

Previously I asked another LLM the following question:
"⟨ORIGINAL PROMPT⟩"
Here is its response:
"""
⟨MODEL RESPONSE⟩
"""
Your task is to identify the approach or algorithm that the model has used to answer the question. Among the following options, select the most appropriate option for the model's response:
A: Top Trading Cycles (TTC)
B: Hungarian Min-cost Algorithm
C: Hopcroft–Karp Maximum Matching Algorithm
D: Searching through all possibilities
E: Deferred Acceptance (Gale–Shapley)
F: Strong stable matching algorithm (Irving)
G: Super stable matching algorithm (Irving)
H: Serial Dictatorship
I: Some other algorithm
J: No algorithm mentioned/described
DO NOT try to verify the algorithm that the LLM has used by computing solutions yourself. Only rely on explicit mentions or descriptions of the algorithms in options A–I. If no algorithm is explicitly mentioned or described, select option J. Return your answer in the following format: `<answer>option</answer>`. For example, if the correct option were option X, you would have to return `<answer>X</answer>`.

Figure 15: Judge prompt used in the generation setting.

Coverage and aggregation. Both settings sweep over the cross-product of model, instance size $\in \{10, 30\}$, domain $\in \{$House Allocation, Shapley-Scarf Markets, Matching Markets$\}$, and preference type $\in \{$strict and complete, strict and incomplete, with ties and complete$\}$, restricted to the relevant subset of each model's responses (the underspecified-instance subset for the generation setting and the selection-task subset for the selection setting).

O.1 Generation Setting

In the generation setting, the judge classifies the algorithm the model used to produce a single solution to an underspecified instance. The option set is restricted to algorithmic procedures: Top Trading Cycles, the Hungarian min-cost algorithm, Hopcroft–Karp matching, exhaustive search, Gale–Shapley deferred acceptance, Irving's strong-stable and super-stable matching algorithms, serial dictatorship, "some other algorithm", and "no algorithm mentioned/described". A response is counted as having *intended* the canonical algorithm for its domain when the judge's choice matches a fixed mapping: house allocation $\leftrightarrow$ Top Trading Cycles, object allocation $\leftrightarrow$ serial dictatorship, and stable matching $\leftrightarrow$ Gale–Shapley. The *achieved* count is computed independently from the model's allocation: for instances with complete preferences, we use the binary correctness label produced by our standard verifier; for instances with incomplete preferences, we recompute the core allocation and compare it against the model's allocation, restricted to the agents that the reference solution actually matches.

A representative judge prompt for this setting is shown in Figure 15. The placeholders ⟨ORIGINAL PROMPT⟩ and ⟨MODEL RESPONSE⟩ are filled in from the row being judged.

O.2 Selection Setting

In the selection setting, the responding model is presented with a set of candidate solutions to an instance, each pre-labeled with the solution concept it satisfies, and is asked to choose one. The judge then classifies which solution concept or criterion the model prioritized in its written justification. The option set covers eleven concepts: (weak) core, utilitarian welfare, egalitarian welfare, rank-maximality, strict core, (weak) stability, strong stability, super stability, Pareto-optimality, maximum-cardinality Pareto-optimal matching, and a catch-all "some other notion or criterion". When a

Previously I asked another LLM the following question:
"⟨ORIGINAL PROMPT⟩"
Here is its response:
"""
⟨MODEL RESPONSE⟩
"""
Your task is to identify the solution concept or criterion that the model prioritizes while answering the question. Among the following options, select the most appropriate option for the model's response:
A: Core: the solution is in the core (if preferences are strict), or in the weak core (if preferences have ties).
B: Utilitarian welfare: the solution minimizes the sum of ranks of matched agents compared to all other options.
C: Egalitarian welfare: the solution has the smallest "worst" rank among all options.
D: Rank-maximality: the solution maximizes the number of agents matched to their first preference; conditional on that, it maximizes the number matched to their second preference, and so on.
E: Strict core: there is no coalition of agents who can re-assign their endowments such that every agent in the coalition is weakly better off and at least one agent is strictly better off.
F: (Weak) stability: the solution has no weakly blocking pair, i.e., no pair of unmatched agents that strictly prefer each other to their matched partners.
G: Strong stability: the solution has no strongly blocking pair, i.e., no pair of unmatched agents in which one agent weakly prefers the other to its current match and the other strictly prefers the first to its current match.
H: Super stability: the solution has no super blocking pair, i.e., no pair of unmatched agents that weakly prefer each other to their matched partners.
I: Pareto-optimality: there is no other solution in which some agent strictly improves without making another agent worse off.
J: Maximum-cardinality Pareto-optimal: the solution maximizes the number of agents matched to a ranked alternative and is Pareto-optimal.
K: Some other notion or criterion.
DO NOT try to verify the property that the model claims to prioritize by computing solutions yourself. Only rely on explicit textual mentions or descriptions of the criteria in options A–J. If multiple criteria are mentioned, use your discretion to identify which one is given the first priority by the model. If no criterion is explicitly mentioned or described, select option K. Return your answer in the following format: `<answer>option</answer>`. For example, if the correct option were option X, you would have to return `<answer>X</answer>`.

Figure 16: Judge prompt used in the selection setting.

response references multiple concepts, the judge is instructed to select the one the model treats as the first priority.

For each row we record three quantities. *Intended* is the concept identified by the judge from the model's reasoning. *Actual* is the concept satisfied by the candidate the model ultimately picked, recovered by mapping each candidate's pre-assigned label onto the same eleven-option set. *Achieved* is an indicator equal to one if and only if the intended and actual concepts agree, i.e., the label of the chosen candidate is the concept its reasoning claims to prioritize. A candidate can satisfy several concepts (for example, a super-stable matching is also strongly and weakly stable), so this is a lower bound on how often the chosen solution satisfies the intended concept. Under incomplete preferences, the candidates labelled Core and MCPO were not checked against those concepts when the menus were built, and the Core candidate is not always a core allocation. The mapping from candidate labels to option letters is fixed and deterministic; in particular, candidates labeled either "Core" (under strict preferences) or "Weak core" (under preferences with ties) are both mapped to the (weak) core option.

A representative judge prompt for this setting is shown in Figure 16.

O.3 Additional judges and human validation

Three judge setups are used in this paper. The algorithm judge (Gemini-2.5-Flash) identifies the algorithm a model uses when no target notion is given, and supports Appendix H.3. The reasoning-strategy judge (DeepSeek V4 Pro) extracts features of GPT-5.2's scratchpads and supports Appendix K. The selection-side judge (Gemini-2.5-Flash) supports the intention-action analysis behind Figure 13.

The lexicographic-completion and premature-termination findings of Figure 3 are detected by a deterministic script and use no judge.

Independent judges. The reasoning-strategy and selection-side judges are each rerun with two additional models (GPT-5.6-Luna and Gemini-3.5-Flash) on a stratified subset. Three-judge Fleiss' κ is 0.64 for the reasoning-strategy judge and 0.82 for the intention-action judge. The lower agreement on the reasoning-strategy judgements comes from the "strongly stable" cell, where the boundary is genuinely ambiguous because the model sometimes gives a valid argument that is not the textbook one, and the judges score these borderline cases inconsistently. Pooling hides where the disagreement sits, so we report this per notion: all three judges agree on 0.95 of the strict-core traces and 0.94 of the super-stable traces, against 0.43 for strongly stable.

Human validation. Human annotation is also used to validate the judge responses. For 36 responses, one author independently chose which concept the model prioritized, using the same options and instructions the judge saw. Against the paper's judge, Gemini-2.5-Flash, human-vs-judge Cohen's kappa is 0.86, in line with the inter-judge agreement. The alternate judges agree a little less, 0.72 and 0.76, mostly on the incomplete house-allocation cell, where the original judge and the human read an implicit assign-everyone justification as max-cardinality intent while the other two judges do not. The human therefore agrees most with the exact judge behind the reported numbers, and the one divergent cell is the one we flag as judge-sensitive in Appendix H.5 rather than pool with the rest.

Feasibility oracle. Correctness is decided by deterministic checker functions that verify every model output against the target solution concept. Because these checkers are our own implementation, we cross-checked them with an independent brute-force solver that enumerates all allocations and matchings for small instances and decides existence directly from each concept's definition. Across 1,800 random instances (strict core, super-stable, and strongly-stable, with n up to 5), the oracle's existence verdicts, the solutions it returns, and the blocking-pair verifier agree with the brute-force solver in every case. We release this validation harness alongside the code.