Name	Description	Example
Answer attempt	The response contains a complete answer attempt to the question that can be extracted verbatim.	The dog is 50 meters away from the house.
Clarification	The response is a brief single question that directly inquires about one aspect of the query.	To calculate the distance, I need to know how long the dog ran. Could you provide more information about that?
Interrogation	The response contains multiple questions addressed to the user.	I cannot answer the question without knowing (1) speed, (2) duration, and (3) starting position. Please tell me about these points and I can calculate the distance!
Discussion	The response discusses the question in detail without answering, asking, or refusing to answer.	The question is trying to measure the distance between the dog and the house. We can calculate based on this equation: [Equation]. [. . .]
Hedging	The response provides multiple answer candidates based on hypotheticals (ifs, cases).	1. If the dog was originally in the house, it would be 50 meters away now. 2. If the dog was at the park, it would be 100 meters away from the house now.
Refusal	The response refuses to answer the question without a follow-up question or a request.	I can't answer your question because I don't have sufficient information.
Missing	The response is empty.	[blank]

Table 8: Definition of turn categories. We include the description in the prompt to categorize assistant responses.

Appendix G Assistant Response Categorization

We categorize each assistant response into one of the seven categories to capture the answer attempt and evaluate if that is the case, as well as to understand the model behavior tendency. Herlihy et al. [27] defined seven turn categories for LLM responses and classified them using LLM, uncovering that GPT-4 prefers answering directly even when the query is underspecified. Motivated by this study, we similarly define seven response categories which we list in Table 8, together with example responses. Key differences are discussion and answer attempt; we observed many responses containing large body of text formulating the question in our preliminary experiments, which led to redefining "Miscellaneous" from [27] into "Discussion" in our experiment. "Direct Response" in [27] corresponds to our "Answer Attempt."

Appendix H Model Access

We accessed models that were used in the experiments from various vendors. The short form names we used throughout the paper, the corresponding versions, and the providers are summarized in Table 9. Except for the exploration with various temperatures (Section 7.2), we set the temperature to $T = 1.0$ and used the default values for the rest of configurable hyperparameters. We set the maximum response length to 1,000 tokens for all models, and did not observe models exceeding this limit frequently when generating responses. For thinking models (o3, Deepseek-R1), we increased the limit to 10,000 tokens to account for the additional test-time compute (thinking tokens).

Appendix I Task-specific Implementation details

We provide task implementation details. For each task, we specify: (1) the selection of original single-turn fully-specified instruction, (2) the evaluation metric that was repurposed from the original dataset, (3) and what the initial system messages consists of (if any).

Short Form	Name	Version	Access Provider
4o	GPT-4o	`gpt-4o-2024-11-20`	OpenAI / Microsoft API
4o-mini	GPT-4o-mini	`gpt-4o-mini-2024-07-18`	OpenAI API
4.1	GPT-4.1	`gpt-4.1-2025-04-14`	OpenAI / Microsoft API
o3	o3	`o3-2025-04-16`	OpenAI / Microsoft API
3-Haiku	Claude 3 Haiku	`claude-3-haiku-20240307`	Amazon Bedrock
3.7-Sonnet	Claude 3.7 Sonnet	`claude-3-7-sonnet-20250219`	Amazon Bedrock
2.5-Flash	Gemini 2.5 Flash	`gemini-2.5-flash-preview-04-17`	Gemini API
2.5-Pro	Gemini 2.5 Pro	`gemini-2.5-pro-preview-03-25`	Gemini API
3.1-8B	Llama-3.1-8B-Instruct	N/A	Local Ollama
3.3-70B	Llama-3.3-70B-Instruct	N/A	Amazon Bedrock
4-Scout	Llama-4-Scout-17B-16E	N/A	Together AI
CMD-A	Command-A	`command-a-03-2025`	Cohere API
R1	Deepseek-R1	N/A	Amazon Bedrock
OLMo2	OLMo2-13B	N/A	Local Ollama
Phi-4	Phi-4	N/A	Local Ollama

Table 9: Specific model versions used as part of our experiments. For each model, we define the exact `Version` of the model accessed (for models that have versioning) and the `Access Provider` to facilitate result reproducibility.

I.1 Code

The Code instructions are sourced from a combination of HumanEval [10], a dataset of 164 basic Python programming problems given the function header and the docstring that specifies the problem, and LiveCodeBench [31], an evolving dataset of Python algorithmic challenges. In particular, we source from the "call-based" problem subset in LiveCodeBench v5, with the difficulty of either "Easy" and "Medium", to align the solution formats between the two sources.

We first sharded all HumanEval problems following the protocol mentioned in Appendix C, obtaining 45 high quality sets of shards that meet the criteria. The rest of the dataset were discarded because of being simplistic, leaving little room to construct sufficient number of shards for a problem. Subsequently, we shuffled and sharded the aforementioned subset from LiveCodeBench until obtaining 100 valid sharded instructions.

We follow the original prompts used by the benchmark authors as much as possible for the single-turn (FULL and CONCAT) evaluation. Specifically, FULL prompt from HumanEval includes the function header and the docstring provided as `prompt` in HumanEval dataset, and FULL & CONCAT from LiveCodeBench includes `starter_code` consisting of the function signature.

Both HumanEval- and LiveCodeBench-derived problems come with test cases which we use to compute the functional accuracy of the answer attempt by the LLMs. We re-use the evaluation codebase maintained by Jain et al. [31], which (1) wraps the candidate function in a test module, (2) execute given the inputs, and (3) checks the equivalence of the output from the expected output, with a default timeout set to prevent the evaluator from getting trapped during evaluation (*e.g.*, brute-force implementation may not pass under the set time budget). In case when multiple code blocks are present in a response, the answer extraction module selects the last function definition in the last markdown code block.

I.2 Database

The Database instructions are sourced from the validation portion of the Spider dataset [86]. We note that though a more recent version of Spider has been released (Spider 2.0 [44]), the instructions in the second iteration are more advanced and represent less typical database use, and we select instructions from the more realistic Spider 1.0.

The authors of Spider categorized queries into four levels of difficulty (EASY, MEDIUM, HARD, XHARD), based on the syntax complexity of a reference SQL query. We filtered out queries of EASY complexity, as they tended to yield fewer than three shards when processed. The rest of the 433 natural language queries in Spider were gradually sharded until reaching a total of 107 valid sharded instructions.

Each original instruction in Spider supplies a database schema, represented in SQL as a series of table schema (i.e., each define a series of columns including name, type, and optional index). We include the database schema as part of the system message (i.e., prior to the first turn of conversation), and informing the LLM that users will provide natural-language queries that must be answered using a database with the provided schema.

Each original instruction in Spider is paired with a reference SQL solution. We follow Zhong et al. [90] for the evaluation methodology. For a given original instruction, the candidate and reference SQL queries are executed on a fixed set of databases, and exact match of the results on all databases is required to mark the candidate as successful (Score = 100). If a discrepancy is observed on any test database, the candidate is incorrect (Score = 0). One limitation of SQL execution is that false positives can occur: two queries can return the same output on a given database, even when they are not semantically equivalent. Zhong et al. [90] found that by evaluating on an increased number of databases, false positives become negligible. Finally, any invalid candidate that does not successfully execute (e.g., syntax error) is considered incorrect (Score = 0).

I.3 Actions

The Actions instructions are sourced from the released test portion of the Berkeley Function Calling Leaderboard V3 (BFCL) [85]. BFCL V3 consists of three sub-genre of instructions: (1) Parallel, (2) Multiple, and (3) Multiple-Parallel. Initial experimentation with the sub-genres identified Parallel as the most suited for sharding, as Parallel instructions specify multiple subtasks that should be used and combined into a single action that accomplishes the entirety of the instruction. We shuffled all the BFCL V3 Parallel instructions, and sharded gradually until we obtained 105 valid sharded instructions.

We note that though a more recent iteration of BFCL includes multi-turn instructions, it differs from sharding experiments as it does not involve underspecification, with each turn having an independent intermediate solution (which we call episodic multi-turn conversations). Our implementation in comparison shards original instructions allowing us to simulate multi-turn underspecified conversations for this task setting. The Background section (Section 2) discusses the relationship between episodic and underspecified multi-turn conversation more in-depth.

Each instruction in BFCL comes with tool set documentation, a JSON object that specifies the set of available actions (APIs) for the assistant to complete user instructions. We include the tool set documentation as part of the system message, along with a message indicating that user queries will require the use of the provided tools to be completed.

Each instruction in BFCL comes with a reference answer, consisting of the API calls that should be called to accomplish the user instruction. The maintainers of BFCL have released an evaluation toolkit that assesses semantic equivalence between a candidate answer and the reference answer. We leverage the official evaluation toolkit, assigning a score of S=100 for candidate answers that are considered semantically equivalent to the reference answer, and a score of S=0 otherwise. When the evaluation toolkit is not able to parse a candidate answer (e.g., a syntax error), the candidate is considered incorrect (S=0).

I.4 Math

The Math instructions are sourced from the "main" portion of the GSM8K dataset [14]. We did not perform a filter on the original 8,700 instructions. We shuffled the instructions and sharded incrementally until we obtained 103 valid sharded instructions. Each GSM8K is paired with a numerical reference answer. We used the official toolkit released alongside GSM8K to standardize numerical answers (i.e., strip formatting, etc.). Standardized candidate numerical answers can then be compared through exact match to the reference answer. If the toolkit detects a match, the candidate answer is considered correct (Score=100), and incorrect otherwise (Score = 0). A short, single-sentence system prompt is used to indicate to the assistant that it will be solving mathematical problems.

I.5 Data-to-Text

The Data-to-Text instructions are based on instructions in the released test set ToTTo dataset [59]. In ToTTo, fully-specified instructions have the following information elements: (1) a HTML-formatted table extracted from a Wikipedia page, (2) a subset of cells in the table that have been *highlighted*, (3) the name of the Wikipedia page that included the Table, (4) the name of the Section in the Wikipedia page that included the Table. Given these elements, the task objective is to generate a caption for the Table specifically focusing on the highlighted cells and considering the available meta-data. Instructions were shuffled and sharded incrementally until we obtained 120 valid sharded instructions.

For each instruction, we generate sharded instructions by assigning different information elements to individual shards. The first shard consists of the initial HTML-formatted table without highlighting. The second shard provides an updated