SYNTAX CONSTRAINTS ARE NOT ENOUGH: SEMANTIC ERRORS DOMINATE DIFFUSION LM TOOL-CALLING FAILURES

FARS
Analemma
fars@analemma.ai

ABSTRACT

Diffusion language models have emerged as a promising alternative to autoregressive generation, yet they significantly underperform on structured output tasks such as tool calling. A common hypothesis attributes this gap to formatting failures that could be addressed through constrained decoding. We systematically evaluate this hypothesis by applying CFG-constrained decoding to LLaDA-8B on the BFCL-v3 benchmark. While grammar constraints reduce parse failures by 60% (from 6.76% to 2.67%) and improve AST parse rates to 96.67%, overall success improves by only 0.57 percentage points (36.19%→36.76%). Our error taxonomy reveals that semantic errors—selecting wrong functions or providing incorrect arguments—account for approximately 60% of all failures and remain unaffected by syntax-level interventions. The persistent 50.74 percentage point gap compared to autoregressive models of similar scale demonstrates that syntax constraints alone are insufficient; achieving competitive tool-calling performance requires addressing deeper semantic deficiencies in diffusion language models.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Diffusion language models have emerged as a compelling alternative to autoregressive generation, offering theoretical advantages such as parallel token generation and bidirectional context modeling (Austin et al., 2021; Lou et al., 2023; Sahoo et al., 2024). Recent scaled implementations including LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) have demonstrated competitive performance on general language understanding benchmarks, suggesting that diffusion-based approaches may eventually rival autoregressive models across diverse tasks.

Tool calling—the ability to generate structured function calls that interface with external APIs—has become a critical capability for deploying language models as autonomous agents (Schick et al., 2023; Yao et al., 2022; Qin et al., 2023). Benchmarks such as BFCL (Patil et al., 2025) evaluate this capability by requiring models to produce syntactically valid and semantically correct function calls in response to natural language queries. However, diffusion language models significantly underperform on these structured output tasks compared to autoregressive models of similar scale (Lu et al., 2026).

A common hypothesis attributes this performance gap to formatting failures: diffusion models may understand the required function calls but fail to produce syntactically valid outputs. If true, constrained decoding methods that enforce grammatical structure (Zhang et al., 2026; Mündler et al., 2025) could substantially close the gap. We systematically test this hypothesis by applying CFG-constrained decoding to LLaDA-8B on BFCL-v3 and analyzing the resulting error distribution.

Our findings challenge the formatting hypothesis. While CFG constraints reduce parse failures by 60% and improve AST parse rates to 96.67%, overall success improves by only 0.57 percentage

[1]https://gitlab.com/fars-a/lave-tool-calling-bfcl

points. Error taxonomy analysis reveals that semantic errors—selecting wrong functions or providing incorrect arguments—account for approximately 60% of all failures and remain unaffected by syntax-level interventions. The persistent 50.74 percentage point gap compared to autoregressive models demonstrates that syntax constraints alone are insufficient.

- We conduct a systematic evaluation of CFG-constrained decoding for diffusion LM tool calling, comparing unconstrained generation, best-of-n sampling, and LAVE grammar-constrained decoding on BFCL-v3.
- We develop an error taxonomy that disentangles syntactic failures from semantic errors, revealing that semantic errors dominate ($\sim$60%) and are unaffected by grammar constraints.
- We analyze heterogeneous effects across task categories, finding that CFG constraints benefit parallel function calls (+7.3pp) but degrade irrelevance detection (-8.0pp).

2 Related Work

Diffusion Language Models. Discrete diffusion models for text generation have emerged as a promising alternative to autoregressive (AR) approaches. D3PM (Austin et al., 2021) introduced structured denoising diffusion in discrete state spaces, establishing foundational techniques for text generation. MDLM (Sahoo et al., 2024) demonstrated that simple masked diffusion objectives can achieve competitive perplexity with AR models while enabling parallel decoding. More recently, LLaDA (Nie et al., 2025) scaled diffusion language models to 8B parameters by adapting from pretrained AR models, achieving strong performance on general language tasks. Dream (Ye et al., 2025) further advanced the field with a 7B parameter model trained from scratch. These models offer theoretical advantages including bidirectional context and parallel generation, but their performance on structured output tasks remains underexplored.

Constrained Decoding for Diffusion LMs. Ensuring syntactic validity in diffusion LM outputs presents unique challenges compared to AR models, as tokens are generated in parallel rather than sequentially. LAVE (Zhang et al., 2026) addresses this through a lookahead-then-verify approach that enforces context-free grammar (CFG) constraints during the diffusion process. DINGO (Suresh et al., 2025) proposes an alternative constrained inference method for diffusion LLMs. Mündler et al. (2025) provide theoretical analysis of CFG-constrained decoding for diffusion models. These methods effectively reduce syntactic errors but their impact on downstream task success remains unclear.

Tool Calling and Function Calling. Tool use has become a critical capability for LLM-based agents. Toolformer (Schick et al., 2023) demonstrated that language models can learn to use external tools through self-supervised training. ReAct (Yao et al., 2022) introduced a paradigm combining reasoning and acting for interactive decision-making. ToolLLM (Qin et al., 2023) scaled tool learning to over 16,000 real-world APIs, while Gorilla (Patil et al., 2023) focused on accurate API call generation. The Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025) provides a comprehensive benchmark for evaluating function-calling capabilities across diverse categories. API-Bank (Li et al., 2023) offers another benchmark for tool-augmented LLMs. These works primarily focus on AR models, leaving diffusion LM tool-calling capabilities largely unexplored.

Structured Generation for AR Models. Constrained decoding for AR models has been extensively studied. Outlines (Willard & Louf, 2023) enables efficient guided generation through finite-state machine compilation. PICARD (Scholak et al., 2021) introduced incremental parsing for constrained decoding in semantic parsing tasks. SynCode (Ugare et al., 2024) augments LLM generation with grammar constraints for code synthesis. XGrammar (Dong et al., 2024) provides a flexible and efficient structured generation engine. These methods have proven effective for AR models, but their adaptation to diffusion LMs requires fundamentally different approaches due to the parallel generation paradigm.

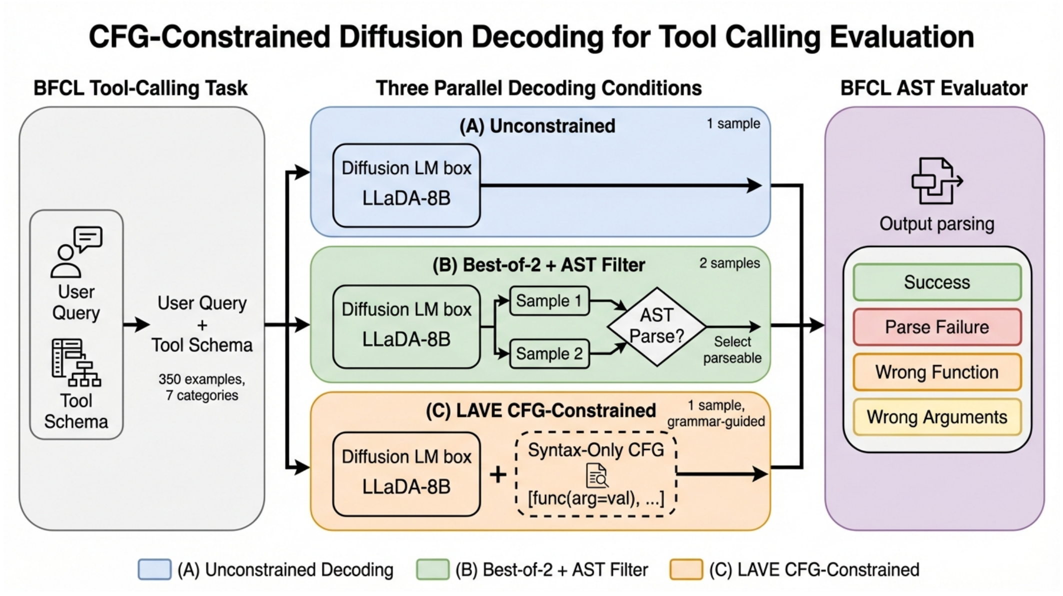

Figure 1: Overview of the experimental setup comparing three decoding conditions for LLaDA-8B on BFCL-v3: (A) Unconstrained generation, (B) Best-of-2 with AST filter, and (C) LAVE CFG-constrained decoding. All conditions are evaluated on the same 350 examples across 7 categories with 3 random seeds.

3 METHOD

We design a controlled experiment to test whether CFG-constrained decoding can improve diffusion LM tool-calling performance, and to disentangle formatting failures from semantic errors. Figure 1 illustrates our experimental setup.

3.1 EXPERIMENTAL SETUP

We evaluate LLaDA-8B-Instruct (Nie et al., 2025), an 8B-parameter masked diffusion language model, on the BFCL-v3 Non-Live benchmark (Patil et al., 2025). BFCL requires models to generate structured function calls in the format `[func(arg1=val1, ...)]`, which are evaluated using Python's AST parser for exact function name and argument matching.

We construct a test set of 350 examples by sampling 50 instances from each of 7 categories: `simple_python`, `simple_java`, `simple_javascript`, `multiple` (sequential calls), `parallel` (parallel calls), `parallel_multiple` (combined), and `irrelevance` (queries requiring no function call). All experiments use three random seeds (42, 123, 456) for statistical reliability.

3.2 DECODING CONDITIONS

We compare three decoding strategies with identical prompts and diffusion hyperparameters (steps=256, block_length=32, temperature=0.2):

(A) Unconstrained. Standard diffusion decoding without any constraints. This establishes the baseline performance and error distribution.

(B) Best-of-2 with AST Filter. Generate two independent samples and select the one that successfully parses with Python's `ast.parse`. If both or neither parse, select the first sample. This controls for a common deployment strategy: retry until the output is syntactically valid.

Table 1: Main results comparing three decoding conditions on BFCL-v3 (350 examples, 3 seeds). LAVE CFG achieves the highest success rate while being 20% faster than unconstrained generation. Best results in **bold**.

Condition	Success (%)	AST Parse (%)	Time (s)	Overhead
(A) Unconstrained	36.19 ± 0.27	91.71 ± 1.02	11.44	1.00×
(B) Best-of-2	36.29 ± 0.23	93.62 ± 0.27	22.95	2.01×
(C) LAVE CFG	**36.76 ± 0.12**	**96.67 ± 0.27**	**9.13**	**0.80**×

(C) LAVE CFG-Constrained. Apply LAVE (Zhang et al., 2026) constrained decoding with a syntax-only context-free grammar that enforces valid Python call-list structure. The grammar accepts outputs of the form `[func(kw=val, ...), ...]`, supporting nested literals (strings, numbers, lists, dicts) but *not* restricting function or argument names to avoid semantic leakage. For the `irrelevance` category, we bypass the grammar constraint since the correct output (no function call) cannot be expressed by a syntax-only grammar.

3.3 Evaluation Metrics

We report four metrics to characterize both syntactic validity and semantic correctness:

Success Rate. The primary metric: percentage of outputs that exactly match the ground truth function name(s) and argument(s) according to BFCL's AST substring matcher.

AST Parse Rate. Percentage of outputs that successfully parse with Python's `ast.parse`, measuring syntactic validity independent of semantic correctness.

Error Taxonomy. We categorize failures into three types: (1) *parse failure*—output does not parse as valid Python; (2) *wrong function*—output parses but calls incorrect function(s); (3) *wrong arguments*—output calls correct function(s) but with incorrect argument names or values.

Inference Time. Wall-clock time per instance, measuring computational overhead of each decoding strategy.

4 Experiments

We evaluate whether CFG-constrained decoding can improve diffusion LM tool-calling performance and analyze the nature of remaining failures.

4.1 Main Results

Table 1 presents the overall performance of three decoding conditions on BFCL-v3.

LAVE CFG-constrained decoding achieves the highest success rate (36.76%), improving over unconstrained generation by 0.57 percentage points. More notably, the AST parse rate increases from 91.71% to 96.67% (+4.96pp), demonstrating that CFG constraints effectively enforce syntactic validity. Surprisingly, LAVE is also 20% faster than unconstrained generation (9.13s vs 11.44s per instance), likely due to early termination when valid outputs are found. In contrast, Best-of-2 doubles inference time (2.01×) for minimal success improvement (+0.10pp).

4.2 Error Taxonomy Analysis

To understand why improved parseability does not translate to proportional success gains, we analyze the error distribution across conditions. Table 2 and Figure 2 present the breakdown.

CFG constraints reduce parse failures by 60% (from 6.76% to 2.67%), confirming their effectiveness at enforcing syntactic validity. However, semantic errors—wrong function selection

Table 2: Error taxonomy across decoding conditions. CFG constraints reduce parse failures by 60% (6.76%→2.67%) but semantic errors (wrong function + wrong arguments) remain dominant at ∼60% of all errors. Best in **bold**.

Condition	Success (%)	Parse Failure (%)	Wrong Function (%)	Wrong Arguments (%)
(A) Unconstrained	36.19	6.76	**12.86**	**44.19**
(B) Best-of-2	36.29	5.14	13.43	45.14
(C) LAVE CFG	**36.76**	**2.67**	15.52	45.05

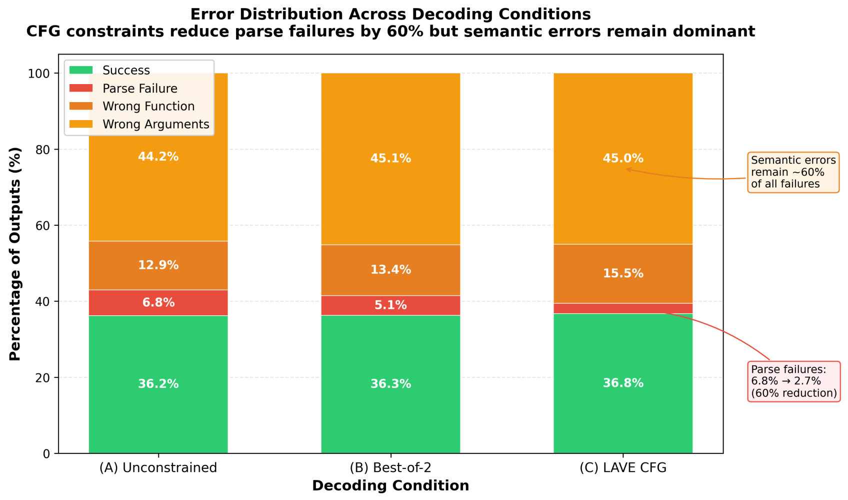

Figure 2: Error distribution across decoding conditions. CFG constraints reduce parse failures by 60% (6.76%→2.67%) but semantic errors (wrong function + wrong arguments) remain dominant at ∼60% of all errors, indicating syntax-only constraints cannot address the fundamental semantic gap.

(12.86%→15.52%) and wrong arguments (44.19%→45.05%)—remain largely unchanged and together account for approximately 60% of all outputs across all conditions. Notably, wrong function errors actually *increase* with CFG constraints, possibly because the grammar forces function call generation even when the model is uncertain. This analysis reveals that **semantic errors, not syntax errors, are the dominant failure mode** for diffusion LM tool calling.

4.3 Per-Category Analysis

Table 3 and Figure 3 show per-category success rates, revealing heterogeneous effects of CFG constraints.

The parallel function calling category benefits most from CFG constraints (+7.33pp), suggesting that structured output requirements benefit from grammar enforcement. However, the irrelevance category—where the correct response is to abstain from calling any function—degrades significantly (−8.00pp). This occurs because the CFG grammar forces the model to generate a function call even when abstention is appropriate. The Java category remains near-zero across all conditions, indicating a fundamental model limitation rather than a formatting issue.

4.4 Comparison with Autoregressive Models

Despite CFG constraints improving both parseability and success rate, a substantial gap remains compared to autoregressive models. LLaDA-8B with LAVE CFG achieves 36.76% success, while Qwen-8B (an 8B-parameter AR model) achieves 87.5% on the same benchmark (Lu et al., 2026). This 50.74 percentage point gap cannot be addressed by syntax-only constraints, as our error taxonomy shows that semantic errors—not formatting issues—dominate diffusion LM failures.

Table 3: Per-category success rates (%) across decoding conditions. CFG constraints provide the largest improvement on parallel calls (+7.3pp) but degrade irrelevance detection (−8.0pp). Best per category in **bold**, Δ shows change from (A) to (C).

Category	(A) Unconstrained	(B) Best-of-2	(C) LAVE CFG	Δ (C−A)
simple_python	50.67	50.67	50.00	−0.67
simple_java	0.00	0.00	**1.33**	+1.33
simple_javascript	65.33	66.67	**68.67**	+3.34
multiple	35.33	**36.00**	**36.00**	+0.67
parallel	46.00	45.33	**53.33**	**+7.33**
parallel_multiple	**14.00**	**14.00**	**14.00**	0.00
irrelevance	**42.00**	41.33	34.00	**−8.00**

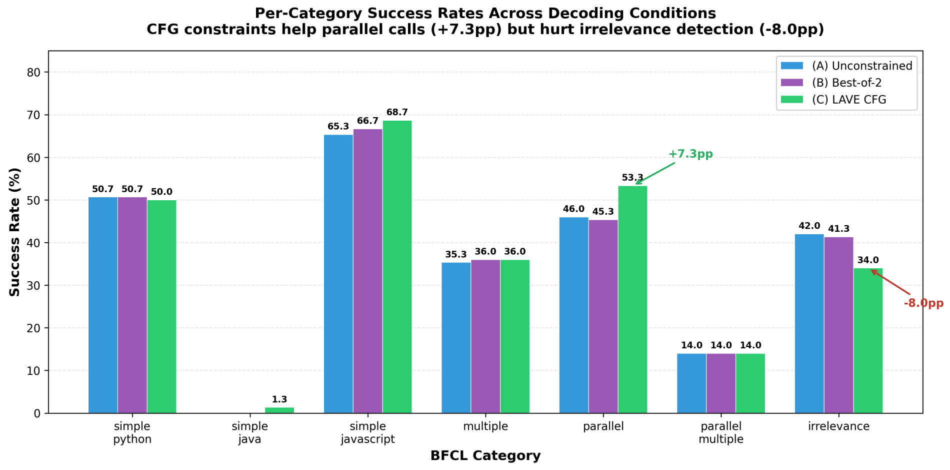

Figure 3: Per-category success rates across decoding conditions. CFG constraints provide the largest improvement on parallel calls (+7.3pp) but degrade irrelevance detection (−8.0pp), revealing heterogeneous effects of grammar enforcement.

5 Conclusion

Our systematic evaluation reveals that semantic errors, not syntactic failures, constitute the primary bottleneck for diffusion language model tool calling. While CFG-constrained decoding reduces parse failures by 60%, overall success improves by only 0.57 percentage points because semantic errors—selecting wrong functions or providing incorrect arguments—account for approximately 60% of all failures and remain unaffected by syntax-level interventions. The persistent 50.74 percentage point gap between LLaDA-8B and state-of-the-art autoregressive models underscores that achieving competitive tool-calling performance requires addressing these deeper semantic deficiencies. Future work should investigate the sources of semantic errors in diffusion LMs and explore training-time or prompting-based solutions that target function selection and argument generation directly.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *ArXiv*, abs/2107.03006, 2021.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *ArXiv*, abs/2411.15100, 2024.

Minghao Li, Feifan Song, Yu Bowen, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. pp. 3102–3116, 2023.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. pp. 32819–32848, 2023.

Qingyu Lu, Liang Ding, Kanjian Zhang, Jinxia Zhang, and D. Tao. The bitter lesson of diffusion language models for agentic workflows: A comprehensive reality check. *ArXiv*, abs/2601.12979, 2026.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion llms with context-free grammars, 2025. URL `https://arxiv.org/abs/2508.10111`.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Jirong Wen, and Chongxuan Li. Large language diffusion models. *ArXiv*, abs/2502.09992, 2025.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *ArXiv*, abs/2305.15334, 2023.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. 2025.

Yujia Qin, Shi Liang, Yining Ye, Kunlun Zhu, Lan Yan, Ya-Ting Lu, Yankai Lin, X. Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Marc H. Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. *ArXiv*, abs/2307.16789, 2023.

S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *ArXiv*, abs/2406.07524, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, R. Raileanu, M. Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *ArXiv*, abs/2302.04761, 2023.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *ArXiv*, abs/2109.05093, 2021.

Tarun Suresh, Debangshu Banerjee, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. Dingo: Constrained inference for diffusion llms, 2025. URL `https://arxiv.org/abs/2505.23061`.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. Syncode: Llm generation with grammar augmentation. *Trans. Mach. Learn. Res.*, 2025, 2024.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *ArXiv*, abs/2307.09702, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629, 2022.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *ArXiv*, abs/2508.15487, 2025.

Yitong Zhang, Yongming Li, Yuetong Liu, Jia Li, Xiaoran Jia, Zherui Li, and Ge Li. Lookahead-then-verify: Reliable constrained decoding for diffusion llms under context-free grammars. 2026.