Masked Diffusion Language Agents for Tool-Integrated Chemical Reasoning

Mengdi Liu[1,2], Chenghao Jia[1,2], Hong Chang[1,2], Shiguang Shan[1,2]
[1] State Key Laboratory of AI Safety, Institute of Computing Technology, CAS, China
[2] University of Chinese Academy of Sciences (CAS), China
{liumengdi23z, jiachenghao25z, changhong, sgshan}@ict.ac.cn

Abstract

Designing tool-integrated reasoning agent systems for complex chemical tasks remains a fundamental challenge. Although recent chemical agents have demonstrated promising performance through the integration of external tools, they are still built on the autoregressive paradigm, which is inherently constrained by causal attention and left-to-right generation. As a result, they often struggle with bidirectional molecular understanding, global coordinated tool planning and multi-source evidence integration. To address this challenge, we introduce CHEMDIFFAGENT, the first masked diffusion language agent for tool-integrated chemical reasoning. It reformulates chemical reasoning as iterative denoising over interleaved reasoning and tool-use trajectories, enabling tool-use decisions to be jointly determined under full-trajectory context and thereby better aligning the agent's decision boundary with its knowledge boundary. We further analyze how this paradigm benefits agentic reasoning through harder infilling subproblems during training and flexible arbitrary-order decoding during inference. Then we develop a two-stage post-training framework, comprising agentic supervised fine-tuning and variance-reduced preference optimization, to enhance tool-use and reasoning capabilities. We also construct a comprehensive benchmark covering six chemistry tasks across both single-turn and multi-turn tool-use scenarios. Under matched budgets and evaluation, CHEMDIFFAGENT consistently outperforms autoregressive agents, achieving an overall improvement of approximately 20%. Notably, our 8B model achieves performance comparable to, in some cases surpassing, significantly larger frontier models, such as GPT 5.5 and Claude Opus 4.7. Code is available at https://github.com/bilemond/ChemDiffAgent.

1 Introduction

With the rapid development of reasoning capabilities [14, 29, 12, 22, 11, 18, 16], large language models (LLMs) have shown growing potential for solving complex chemistry tasks [36, 30, 8, 4, 15]. However, chemical reasoning often goes beyond parametric knowledge, requiring an agent to interpret domain-specific modalities, orchestrate tool invocations with valid arguments, and synthesize the returned response into subsequent reasoning steps. This process, which we refer to as ***tool-integrated chemical reasoning***, tightly couples internal reasoning with external tool use, enabling agents to systematically coordinate introspection and interaction. In this sense, an effective chemistry agent should align its tool-use decision boundary with its actual knowledge boundary: recognizing the limits of its parametric knowledge, judiciously deciding when and which tool to invoke, and reconciling potentially conflicting evidence from multiple sources.

Correspondence to: Hong Chang <changhong@ict.ac.cn>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

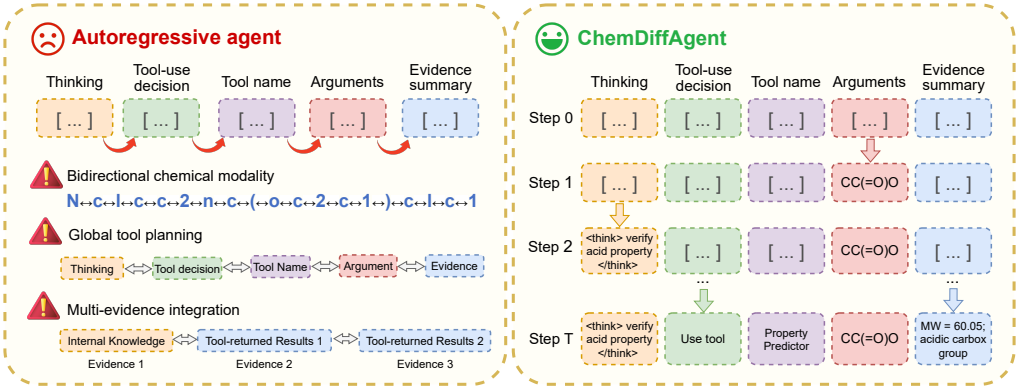

Figure 1: Compared with autoregressive agents that commit to tool-use trajectories sequentially, ChemDiffAgent jointly refines chemical modality thinking, tool-use decisions, arguments, and evidences to better handle bidirectional chemical modalities and globally coordinated tool use.

Recently, chemistry agents have demonstrated promising advances by integrating external tools such as molecular property predictors, reaction engines and cheminformatics libraries [4, 15, 5, 24]. However, they rely on the autoregressive (AR) paradigm, where the unidirectional context modeling and left-to-right generation are inherently misaligned with the nature of chemical reasoning (as shown in Fig. 1). First, chemistry-specific modalities such as SMILES are *bidirectional*, with *complex dependencies* among tokens at different positions. Second, tool-integrated chemical reasoning is *globally coupled*, as the choice of tool, its arguments, and the reasoning goal mutually depend on each other. Third, complex chemical tasks frequently require integrating *multiple sources of evidences* into a single coherent answer. Nevertheless, autoregressive agents are inherently constrained by strict left-to-right token commitment, which precludes joint coordination across the sequence and consequently leads to locally biased tool selection and unstable multi-step execution.

To address these limitations, we introduce ChemDiffAgent, the first masked diffusion language agent for tool-integrated chemical reasoning. It reformulates chemical reasoning as iterative denoising over interleaved reasoning and tool-use trajectories. This formulation allows the agent to jointly determine intermediate thinking, tool-use decisions, tool names, arguments, and multi-evidence within each block under bidirectional context. As a result, it enables more reliable and flexible chemical reasoning through a unified denoising formulation, rather than being committed prematurely.

We further analyze why this paradigm is well suited for agentic chemical reasoning from both training and inference perspectives. During agentic training, ChemDiffAgent exposes the model to harder infilling subproblems over interleaved reasoning and tool-use trajectories. During inference, the arbitrary-order decoding within each block provides greater flexibility for coordinating interdependent decisions, such as selecting a tool, filling its arguments and integrating evidence before committing to the next action. Building on this analysis, we develop a two-stage post-training framework comprising agentic supervised fine-tuning (SFT) and variance-reduced preference optimization (VRPO) to enhance tool-use decision-making and chemical reasoning capabilities.

We also construct a comprehensive benchmark covering six chemistry tasks across single-turn and multi-turn tool-use scenarios. Under matched budgets and constraints, ChemDiffAgent consistently outperforms autoregressive agents, achieving an overall improvement of approximately 20%. Notably, our 8B model achieves performance comparable to, and in some cases surpassing, substantially larger frontier models such as GPT-5.5 and Claude Opus 4.7. These results suggest that masked diffusion provides a promising foundation for building more reliable, globally coordinated, and tool-aware agents for complex scientific reasoning.

In summary, our contributions are as follows:

- We introduce ChemDiffAgent, the first masked diffusion language agent for tool-integrated chemical reasoning. It formulates chemical reasoning as an iterative denoising process over interleaved reasoning and tool-use trajectories, enabling global attention and coordination.

- We provide a theoretical analysis revealing why the diffusion paradigm offers advantages over autoregressive generation in tool-integrated chemical reasoning. Building on this analysis, we develop a two-stage post-training framework, comprising Agentic SFT and Agentic VRPO.

- We construct a comprehensive benchmark covering six chemistry tasks spanning single-turn and multi-turn tool-use scenarios. Experiments demonstrate that CHEMDIFFAGENT substantially outperforms autoregressive agents and achieves performance comparable to, or even surpassing, larger frontier models such as GPT-5.5 and Claude Opus 4.7.

2 Related Work

2.1 Chemistry Agents

Recent research has demonstrated promising advances in integrating LLMs with external chemistry tools [36, 30, 8, 4, 15] to overcome the unreliability of purely parametric knowledge on domain tasks. Early efforts, such as StructChem [28], broke down chemical reasoning into structured phases, including formula generation and step-by-step reasoning. ChemCrow [5], ChemToolAgent [43], and others [23, 24] further enhanced LLMs by integrating expert-designed tools for tasks like molecular design and drug development. To enable better ***tool planning and execution***, CheMatAgent [38] presents the Hierarchical Evolutionary MCTS framework, while SciToolAgent [10] leverages a scientific tool knowledge graph that encodes relationships among hundreds of tools. To solve complex real-world problems, ***multi-agent systems*** such as MT-Mol [17], and ChemMAS [39] decompose chemical workflows across role-specialized agents. Additionally, ***self-evolving agents*** including ChemAgent [34] and S1-NexusAgent [35] accumulate experience through self-updating memory libraries. Despite great progress, these methods rely on autoregressive models, which lack a global understanding of chemical sequences and are constrained by their sequential generation process. To the best of our knowledge, we are the first to introduce a dLLM chemistry agent, *enabling bidirectional chemical modality understanding, global tool planning and multi-evidence integration*.

2.2 Diffusion Large Language Models and DLLM Agents

Diffusion Language Models. Inspired by discrete diffusion models [2, 6, 25, 20, 45], dLLMs have emerged as a promising alternative to AR models (ARMs). LLaDA [26], an 8B diffusion language model trained from scratch, achieves performance competitive with LLaMA3-8B. Dream7B [41] introduces a comprehensive training framework that leverages AR-based LLM initialization and context-adaptive noise scheduling to scale diffusion language models. More recently, industrial efforts such as Gemini Diffusion [9], Mercury [13], and Seed-Diffusion [33] have further scaled dLLMs and demonstrated their potential for efficient inference.

Block Diffusion Language Models. Furthermore, hybrid architectures such as Block Diffusion Language Models (BDLMs) [1, 7, 37, 3] have become a significant research focus. BDLMs employ attention mechanisms combining intra-block bidirection with inter-block causal. This architecture natively supports KV Cache and variable-length text generation and keeps the non-autoregressive generation ability within each block, allowing parallel decoding in arbitrary orders. SDAR [7], a series of BDLMs ranging from 1.7B to 8B parameters, has demonstrated general-purpose capabilities comparable to the latest open-source ARMs.

DLLM Agents. Recent works have started to investigate diffusion large language models (dLLMs) as agent backbones. DLLM-Agent [46] reports over 30% speedup with fewer interaction rounds at comparable accuracy on BrowseComp-zh and web open-ended tasks compared to AR backbones. DLLM-Searcher [44] combines agentic post-training with a Parallel-Reasoning-and-Acting paradigm that overlaps decoding and tool-waiting, matching mainstream search agents with a 15% speedup. However, these works are confined to web searches. To our knowledge, our work is the first to explore diffusion language agents for scientific reasoning, with a focus on *complex chemical reasoning scenarios involving multi-step, multi-modal, and multi-tool coordination*.

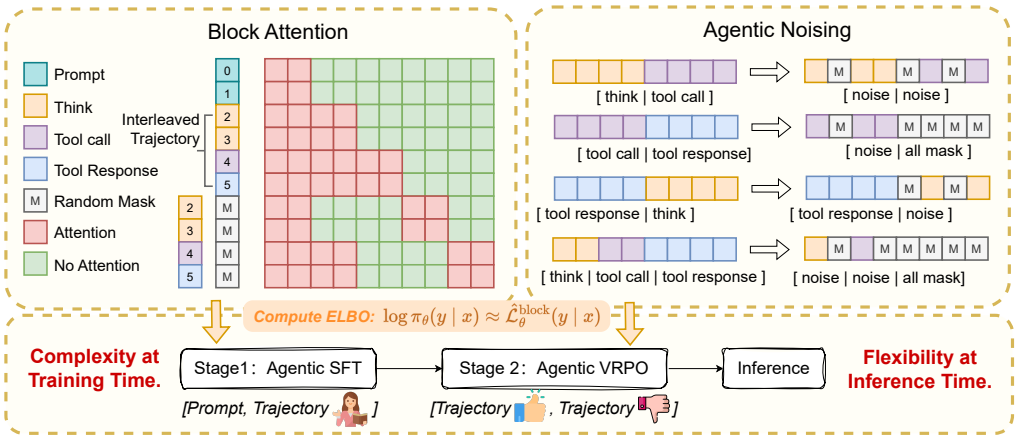

Figure 2: Overview of CHEMDIFFAGENT. In training, both Agentic SFT and Agentic VRPO use Block Attention and Agentic Noising to compute the Agentic ELBO. It learns from ***complex*** block-infilling tasks at training time and enables ***flexible*** order-agnostic reasoning at inference time.

3 Preliminaries

Diffusion Large Language Models. Diffusion Large Language Models (dLLMs), or more specifically Masked Diffusion Models (MDMs), define a forward process that gradually corrupts the clean input by replacing tokens with the mask token M. Given the prompt x and the clean completions y, the forward process $q_t(y_t|y,x)$ at time t is defined as follows:

$$q_t(y_t|y,x) = \prod_{i=1}^{L} q_t(y_t^i|y^i,x) \quad \text{where} \quad q_t(y_t^i|y^i,x) = \begin{cases} 1-t, & y_t^i = y^i, \\ t, & y_t^i = \mathrm{M}. \end{cases} \tag{1}$$

In the reverse process, the model predicts the original values of the masked tokens to compute the reverse probability $p_\theta(\cdot|y_t,x)$. To learn this distribution effectively, dLLMs typically adopts the Evidence Lower Bound (ELBO) $\mathcal{L}_\theta(y|x)$ as a surrogate objective to approximate the exact log-likelihood $\log \pi_\theta(y|x)$ [27, 32, 31]:

$$\mathcal{L}_\theta(y|x) \triangleq \mathbb{E}_{t\sim\mathcal{U}[0,1]}\mathbb{E}_{y_t\sim q_t(y_t|y,x)}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[y_t^i=\mathrm{M}]\log p_\theta(y^i|y_t,x)\right] \le \log \pi_\theta(y|x). \tag{2}$$

For BDLMs, the input y is partitioned into K continuous blocks $[y^1,\ldots,y^K]$, each of length B. The corresponding ELBO of is defined as:

$$\mathcal{L}_\theta^{block}(y|x) \triangleq \mathbb{E}_{t\sim\mathcal{U}[0,1]}\mathbb{E}_{y_t\sim q_t(y_t|y,x)}\left[\frac{1}{t}\sum_{k=1}^{K}\sum_{i=1}^{B}\mathbf{1}[y_t^{k,i}=\mathrm{M}]\log p_\theta(y^{k,i}|y_t,y^{<k},x)\right]. \tag{3}$$

4 CHEMDIFFAGENT

In this section, we propose CHEMDIFFAGENT, a masked diffusion language agent for tool-integrated chemical reasoning. The overview framework is shown in Figure 2.

4.1 Theoretical Foundations of Flexible Tool Reasoning

Iterative Denoising over Interleaved Trajectory. Given a prompt x, CHEMDIFFAGENT produces an interleaved trajectory $\mathcal{H} = (r_1, c_1, o_1, \ldots, r_T, c_T, o_T, a)$, where r_t is a reasoning segment, $c_t = (\tau_t, \alpha_t)$ a call to tool τ_t with arguments α_t, $o_t = \tau_t(\alpha_t)$ its response, and a the final answer. Unlike autoregressive agents that commit to each token from left to right, CHEMDIFFAGENT models each reasoning–action trajectory as a denoising process over masked blocks. This allows tool names, arguments, control decisions, and reasoning processes to be jointly refined under full block context.

Training Objective Design for Flexible Infilling. DLLM agents learn to reconstruct masked content from partially observed contexts. This exposes the model to different combinations of available information rather than a single prefix-based prediction pattern. To characterize this supervision, partition a clean completion y into K contiguous blocks $[y^1, \ldots, y^K]$, where $y^k = (y^{k,1}, \ldots, y^{k,B})$ and $L = KB$. For a non-empty block set $A \subseteq [K]$, let $y[A]$ replace all tokens in blocks indexed by A with the mask token M. Under uniform sampling of the number of masked blocks and of the block subset conditional on its size, the reconstruction objective is

$$\mathcal{L}_\theta^{\mathrm{block}} = \mathbb{E}_{(x,y)\sim p_{\mathrm{data}}} \left[\sum_{\emptyset \neq A \subseteq [K]} \frac{\sum_{k\in A} \sum_{i=1}^{B} \log p_\theta(y^{k,i} \mid y[A], y^{<k}, x)}{K\binom{K}{|A|}|A|B} \right]. \tag{4}$$

Here, p_{data} is the data distribution and p_θ the model's conditional distribution. By optimizing $\mathcal{L}_\theta^{\mathrm{block}}$, the model learns to reconstruct missing tokens conditioned on arbitrary visible context combinations. Crucially, this objective directly leads to flexible inference: because p_θ learns valid posteriors under any mask pattern A, it establishes the foundational capacity to evaluate and decode along arbitrary trajectory paths during generation. The detailed derivation is provided in Appendix D.1.

Adaptive path advantage under non-uniform subproblem difficulty. Different parts of a reasoning problem can have different conditional difficulties, and resolving one part may simplify another. Let $C = (X, Y^{<k})$ denote the preceding context, $U = (U_1, \ldots, U_B)$ the current block, and $S_t = \{i_1, \ldots, i_{t-1}\}$ the positions resolved along a path $\pi = (i_1, \ldots, i_B)$. The prediction difficulty at each position, measured by conditional cross-entropy, consists of conditional entropy and model approximation error. Summing along the path gives the expected negative log-likelihood

$$\mathcal{R}_\theta(\pi) = H(U \mid C) + \mathbb{E}_{p_{\mathrm{data}}} \left[\sum_{t=1}^{B} \delta_\theta(i_t \mid C, U_{S_t}) \right], \tag{5}$$

where H denotes conditional entropy and $\delta_\theta(i \mid C, U_S) = D_{\mathrm{KL}}(p_{\mathrm{data}}(U_i \mid C, U_S) \| p_\theta(U_i \mid C, U_S))$ is the model's approximation error. The accumulated approximation error depends on the intermediate contexts. Since the admissible paths Π_{flex} include $\pi_{\mathrm{AR}} = (1, \ldots, B)$, $\inf_{\pi\in\Pi_{\mathrm{flex}}} \mathcal{R}_\theta(\pi) \leq \mathcal{R}_\theta(\pi_{\mathrm{AR}})$ for a fixed predictor. Flexible ordering can therefore exploit differences in subproblem difficulty to seek a path with smaller accumulated error. The detailed derivation is provided in Appendix D.2.

Strict ordering advantage under non-uniform decision risk. Beyond prediction difficulty, errors at different positions can have different downstream consequences. Given context C, let Z be an earlier high-risk decision in the AR order and R a later lower-risk decision in the same block that provides information about Z. Let $w_Z > w_R > 0$ be their fixed downstream error-cost weights. For Bayes-optimal conditional predictions, the risk-weighted log losses are $\mathcal{J}_{\mathrm{AR}} = w_Z H(Z \mid C) + w_R H(R \mid C, Z)$ and $\mathcal{J}_{R\to Z} = w_R H(R \mid C) + w_Z H(Z \mid C, R)$. Their difference satisfies

$$\mathcal{J}_{\mathrm{AR}} - \mathcal{J}_{R\to Z} = (w_Z - w_R) I(Z; R \mid C), \tag{6}$$

where I denotes conditional mutual information. If $I(Z; R \mid C) > 0$, resolving R before Z strictly reduces this weighted risk. Thus, flexible ordering can first resolve lower-risk content and use more information to guide a higher-risk decision. The detailed derivation is provided in Appendix D.3.

4.2 Agentic Supervised Fine-Tuning

Directly applying existing dLLM backbones to chemistry-agent tasks exposes two practical limitations. First, the model must reliably follow the executable tool-calling protocol required by the chemistry tool environment. Second, it must perform multi-step chemical reasoning rather than produce a single response. We therefore introduce Agentic SFT to adapt dLLMs to chemistry-oriented tool reasoning.

Teacher Trajectory Construction. For each query Q, we use a high-performance teacher model to generate a candidate chemistry-agent trajectory $\mathcal{H}_{\text{teacher}}$. The teacher is allowed to select from a small set of relevant chemistry tools rather than the full tool pool, which reduces irrelevant exploration while still requiring tool selection. We perform trajectory-level rejection sampling and retain a trajectory

only if it satisfies three criteria: (1) the final answer is correct; (2) the reasoning process is clear and complete; and (3) every tool selection and argument is correct and strictly follows the required format. The filtered query–trajectory pairs $(Q, \mathcal{H}_{\text{teacher}})$ are then used as the Agentic SFT training set.

Block Attention for Tool-Use Trajectories. To enable efficient training over noised trajectories, we adopt a block-level bidirectional attention scheme tailored to tool-use data, as illustrated in Fig. 2. Specifically, we concatenate the query, the clean teacher trajectory, and a noised copy of the trajectory into a single input sequence, $[Q, \mathcal{H}_{\text{teacher}}, \mathcal{H}_{\text{noisy}}]$. This design allows all noised blocks to be supervised in a single forward pass, conditioned on the corresponding clean trajectory.

Agentic Noising. Chemistry trajectories interleave *agent-controlled* decisions, such as `think` and `tool_call` steps, with *external observations* from `tool_response` spans. Agentic Noising perturbs only agent-controlled regions, leaving tool responses untouched as grounded context. To prevent train–inference leakage from intra-block bidirectional attention, we fully mask any `tool_response` token that would otherwise be visible within the same bidirectional block.

Agentic ELBO. The training objective should also reflect the fact that only agent-side decisions are supervised. We therefore modify Eq. 3 and define the Agentic ELBO $\hat{\mathcal{L}}_\theta^{block}$ as:

$$\hat{\mathcal{L}}_\theta^{block}(y \mid x) \triangleq \mathbb{E}_{t\sim\mathcal{U}[0,1], y_t\sim\hat{q}_t(y_t|y,x)} \left[\frac{1}{t}\sum_{k=1}^{K}\sum_{i=1}^{B}\mathbf{1}\Big[(y_t^{k,i}=[\mathrm{M}])\wedge(y^{k,i}\notin[\mathrm{R}])\Big]\log p_\theta(y^{k,i}\mid y_t^k, y^{<k}, x)\right], \tag{7}$$

where [R] denotes the `tool_response` regions. The loss is computed only on positions that are masked and do not belong to external tool observations. These positions are excluded from the loss because they are not generated by the agent.

Training Objective. Instead of optimizing the standard autoregressive negative log-likelihood, we use the proposed Agentic ELBO to approximate $\log \pi_\theta(y \mid x)$ under the dLLM objective, where x denotes the prompt and y corresponds to the teacher trajectory $\mathcal{H}_{\text{teacher}}$. The final SFT loss is:

$$\mathcal{L}_{SFT} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[-\hat{\mathcal{L}}_\theta^{block}(y \mid x)\big]. \tag{8}$$

4.3 Agentic Variance-Reduced Preference Optimization

Agentic SFT improves the model's capability of executable chemistry tool calling, but it does not explicitly optimize preference between successful and failed multi-step trajectories. Motivated by prior post-training methods for dLLMs [47], we further introduce Agentic VRPO to improve long-horizon chemistry-agent decision making. The key idea is to contrast a successful trajectory with a failed one under the same query, with both evaluated via the Agentic ELBO.

Variance-Reduced Agentic Scoring. For each trajectory y, we sample M corrupted contexts through Agentic Noising. Let $\xi_j = (t_j, y_{t_j})$ denote the noise time and corrupted trajectory in the j-th sample. The corresponding agent-only reconstruction score is

$$\ell_\theta(y \mid x; \xi_j) = \frac{1}{t_j}\sum_{k=1}^{K}\sum_{i=1}^{B}\mathbf{1}\Big[y_{t_j}^{k,i}=\mathrm{M},\ (k,i)\notin\mathcal{O}(y)\Big]\log p_\theta(y^{k,i}\mid y_{t_j}^k, y^{<k}, x), \tag{9}$$

where $\mathcal{O}(y)$ is the set of tool-response positions. Agentic Noising masks future tool observations, while the indicator excludes external observations from scoring. Using the same corrupted input for the policy and fixed reference model, we estimate their Agentic ELBO difference as

$$\widehat{\Delta}_M(y \mid x) = \frac{1}{M}\sum_{j=1}^{M}\left[\ell_\theta(y \mid x; \xi_j) - \ell_{\text{ref}}(y \mid x; \xi_j)\right]. \tag{10}$$

Table 1: Single-turn success rate performance across six chemistry task families: composition inference, name conversion, property prediction, retrosynthesis, molecular optimization, and forward synthesis. Red numbers denote absolute gains of SDAR-8B+SFT over Qwen3-8B+SFT, and of SDAR-8B+SFT+RL over SDAR-8B+SFT. The **best** and second-best results are highlighted.

Method	Comp.	Name Conv.	Prop. Pred.	Retro.	Mol. Opt.	Fwd. Synth.	Total
Strong LLMs with ReAct							
GLM-5.1	39.8	19.6	40.6	38.5	0.0	30.0	32.8
Gemini 2.5 Flash	77.9	40.2	67.7	0.0	79.2	75.0	59.8
DeepSeek-V3.2	83.5	71.6	90.6	50.8	88.7	78.3	77.0
GPT-5.5	71.4	56.9	64.6	96.9	64.2	91.7	70.6
Claude Opus 4.7	90.9	88.2	87.5	**100.0**	**100.0**	**100.0**	90.9
Chemistry agents							
ChemCrow (Claude Opus 4.7)	91.2	92.2	83.3	95.4	**100.0**	86.7	89.6
ChemCrow (DeepSeek-V3.2)	55.0	53.2	53.0	72.6	43.1	28.3	72.4
S1-NexusAgent(GPT-5.5)	71.4	78.8	77.5	71.8	38.5	33.3	76.7
Autoregressive 8B model							
Qwen3-8B + SFT	76.6	53.9	87.5	78.5	74.5	63.3	69.1
Diffusion large language model (w/o SFT)							
SDAR-8B	49.4	1.0	17.7	7.7	11.3	5.0	23.3
ChemDiffAgent (ours)							
SDAR-8B + SFT	91.3	78.4	94.8	96.9	75.5	98.3	89.0
SDAR-8B + SFT + RL	**91.8** (+0.5)	**97.1** (+18.7)	**95.8** (+1.0)	96.9 (+0.0)	79.2 (+3.7)	98.3 (+0.0)	**91.8** (+2.8)

Training Objective. The aggregated scores define the preference objective:

$$\mathcal{L}_{\mathrm{VRPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\mathrm{pref}},\,\Xi}\left[\log\sigma\left(\beta\left[\widehat{\Delta}_M(y_w \mid x) - \widehat{\Delta}_M(y_l \mid x)\right]\right)\right], \tag{11}$$

where Ξ collects the sampled contexts for both trajectories, σ is the sigmoid function, and $\beta > 0$ controls the preference scale. Scores are aggregated to reduce reliance on individual context samples.

5 Experiments

5.1 Dataset and Experimental Setup

Single-turn dataset construction. We first collect chemistry datasets covering six fundamental molecular- and reaction-centered tasks, including composition inference, name conversion, property prediction, retrosynthesis, molecular optimization, and forward synthesis. We construct the single-turn split by preserving the original task semantics and using a strong teacher model to generate an execution trajectory. The trajectory construction details are shown in section 4.2.

Multi-turn dataset construction. For multi-turn construction, we formulated each instance as a *graph-conditioned tool-use task*. Inspired by SciToolAgent [10], we first build a chemistry tool graph, where nodes represent tools and edges denote the input–output compatibility. We then perform random walks over the graph to sample subgraphs and use a strong teacher model to compose a query that combines the original task with all sampled tools. Trajectories are subsequently constructed by executing the tool chain in a forward manner. We retain a candidate only if its tool-calling trajectory matches the sampled subgraph, followed by manual inspection for query coherence, valid tool use, and answer faithfulness. The resulting multi-turn examples contain execution-verified final answers and gold tool-calling graph. Based on the length of the gold calling chain, we partition it into three difficulty levels, namely Easy, Medium, and Hard, corresponding to 2, 3–5, and 6–7 tool-calling rounds respectively. More construction details are provided in Appendix A.

Evaluation metrics. We report two main metrics: (i) answer correctness, judged by GPT-4o on whether the final response solves the task, and (ii) tool selection accuracy, the fraction of the m expected tools in the gold chain invoked by the model. Full protocols are in Appendix B.

Baselines. We compare ChemDiffAgent with a broad set of baselines. We first train Qwen3-8B as a comparable-scale autoregressive baseline. Note that our backbone SDAR-8B is continually

pretrained from Qwen3-8B and therefore shares the same parameter scale and pretraining lineage. Both AR and DLLM agents are under fair comparison, including the same SFT data, training epochs, context length, and tool interface. We also evaluate SDAR-8B without SFT as the vanilla diffusion language model baseline. Moreover, we evaluate LLMs under ReAct framework [40], including GLM-5.1, Gemini 2.5 Flash, DeepSeek-V3.2, GPT-5.5, and Claude Opus 4.7. We then evaluate chemistry agents, including ChemCrow [5] and S1-NexusAgent [35].

Implementation details. All experiments are conducted on 8 NVIDIA A100 GPUs. For ChemDiffAgent, we set both the block size and the number of denoising steps to 64. All baselines are equipped with the required tools. Additional implementation details are provided in Appendix C.

5.2 Benchmarking Chemistry Agents Performance

Single-turn tool-grounded chemistry reasoning. Table 1 reports single-turn performance across six chemistry task families. We highlight three observations: (1) Under matched model scale, training data, tool budget, and inference constraints, SDAR-8B with SFT substantially outperforms its autoregressive counterpart Qwen3-8B with SFT, improving the overall score of 19.9%. This indicates that the diffusion paradigm provides a stronger foundation for tool-integrated chemical reasoning than autoregressive. (2) Although our RL stage is trained on multi-turn data, our method further improves the overall score from 89.0% to 91.8% (+2.8%). This indicates that preference optimization over agentic trajectories transfers effectively to simpler single-turn, tool-grounded reasoning settings. (3) ChemDiffAgent attains the best overall score of 91.8%, surpassing both strong frontier models equipped with ReAct (e.g., GPT-5.5 and Claude Opus 4.7) and dedicated chemistry agent frameworks such as ChemCrow [5], despite relying on only an 8B backbone.

Multi-turn tool orchestration and reasoning. We then analyze multi-turn tool orchestration and reasoning performance. Results are reported in Table 2. We observe that: (1) Under the same SFT setting, ChemDiffAgent outperforms its autoregressive counterpart by 18.3%, indicating that the diffusion paradigm retains a substantial advantage on long-horizon chemistry tool orchestration and reasoning. (2) After RL training, ChemDiffAgent surpasses almost all strong frontier models including GPT 5.5, despite using only an 8B backbone. (3) As the number of required rounds grows, all models exhibit a clear performance drop; however, ChemDiffAgent after RL remains competitive with Claude Opus 4.7 on the Hard split, confirming the strength of our approach in long-horizon tool orchestration. We also report the tool selection accuracy across difficulty levels, reported in Fig. 3. We can see that ChemDiffAgent consistently even surpasses frontier models like GPT-5.5 and Claude Opus 4.7 in tool selection accuracy, particularly as task complexity increases.

Table 2: Success rate on multi-turn tool-use setting. Easy, Medium, Hard correspond to 2, 3–5, 6–7 tool-calling rounds.

Method	Easy	Med.	Hard	Total
Strong LLMs w/ ReAct				
GLM-5.1	15.1	5.7	1.9	10.0
Gemini 2.5 Flash	21.8	22.1	6.5	17.8
DeepSeek-V3.2	81.6	53.3	6.5	46.8
GPT-5.5	92.0	63.3	31.8	61.1
Claude Opus 4.7	98.9	76.4	44.9	72.8
Chemistry agents				
ChemCrow (Claude Opus 4.7)	100.0	75.4	48.6	73.5
ChemCrow (DeepSeek-V3.2)	64.5	36.5	9.9	35.5
S1-NexusAgent(GPT-5.5)	73.6	60.3	15.0	50.9
Autoregressive 8B model				
Qwen3-8B + SFT	74.5	42.8	20.5	44.0
Diffusion large language model (w/o SFT)				
SDAR-8B	14.9	11.0	8.4	11.2
ChemDiffAgent (ours)				
SDAR-8B + SFT	75.9	66.8	43.0	62.3
SDAR-8B + SFT + RL	78.2 (+2.3)	69.9 (+3.1)	44.9 (+1.9)	64.6 (+2.3)

Tool-use behaviour analysis. ChemDiffAgent demonstrates more efficient and accurate tool-use. As shown in Table 3, its tool-calling rounds closely align with the expected steps across various task complexities. This suggests that the agent can adaptively stop once it completes the necessary actions, rather than making redundant calls. In contrast, models like Gemini 2.5 Flash often execute more rounds than necessary, while GPT-5.5 frequently fails to invoke sufficient tool calls to solve the task. Moreover, ChemDiffAgent maintains compact trajectory lengths, achieving the shortest single-turn output length. These results show that ChemDiffAgent avoids long-winded reasoning

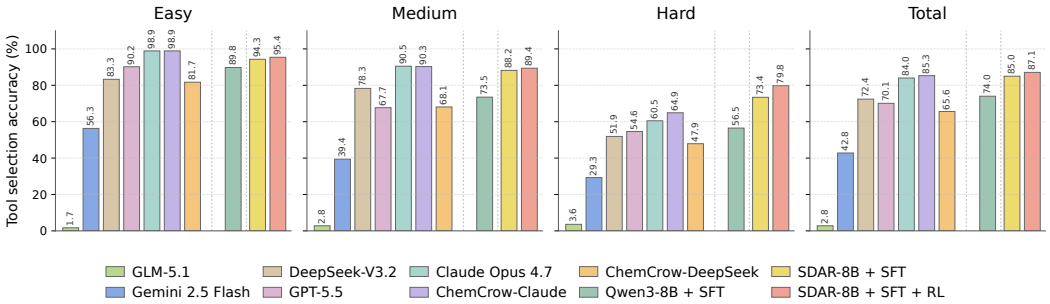

Figure 3: Multi-turn tool selection accuracy across difficulty levels. Easy, Medium, and Hard correspond to 2, 3–5, and 6–7 tool-calling rounds, respectively.

Table 3: Efficiency analysis of tool-use trajectories. Left: average executed tool rounds under different expected rounds. Right: average output length in single-turn and multi-turn settings.

(a) Executed tool-calling rounds

Method	2	3	4	5	6	7	Avg.
Claude Opus 4.7	**1.99**	3.29	4.16	5.50	5.65	6.62	3.90
Gemini 2.5 Flash	5.74	5.81	5.63	7.00	6.85	6.36	6.00
DeepSeek-V3.2	2.78	4.28	5.11	7.33	7.09	7.32	4.84
GPT-5.5	1.90	2.88	**4.05**	6.00	4.76	5.52	3.42
Qwen3-8B(SFT)	3.67	5.35	5.26	7.33	7.50	7.58	5.60
ChemDiffAgent(SFT)	2.18	3.44	3.89	**4.83**	**5.97**	6.93	4.07
ChemDiffAgent(RL)	2.18	**3.09**	3.58	**4.83**	6.12	**7.01**	3.93

(b) Average trajectory length

Method	Single	Multi
Claude Opus 4.7	456.8	1228.2
Gemini 2.5 Flash	1012.8	2370.3
DeepSeek-V3.2	2821.6	3684.2
GPT-5.5	573.0	1515.4
Qwen3-8B(SFT)	648.8	1769.4
ChemDiffAgent(SFT)	571.2	1746.8
ChemDiffAgent(RL)	548.2	1747.7

or redundant tool use. Instead, it generates clear, concise, and goal-oriented trajectories, making scientific problem-solving much more efficient.

5.3 Case Study

Visualization of relative decoding order. Figure 4 illustrates the token-level denoising sequence of a tool-calling block in ChemDiffAgent. Parts of the action framework, including tool-call delimiters and argument structure, are established relatively early, while selected tokens in the SMILES argument remain masked until later stages. The resulting order interleaves structural and chemical decisions, rather than completing the argument strictly from left to right. This behavior is desirable for tool-use agents because a valid action is not determined by the tool name alone: the tool name, argument schema, control tokens, and exact chemical input must be mutually consistent. Therefore, ChemDiffAgent can condition low-level chemical details on a more stable global action structure, reducing premature commitment to incorrect arguments and improving the executability of tool calls. The example thus illustrates how flexible ordering allows specific chemical decisions to be made after additional within-block context has been established. A complete example is provided in Appendix E.

AR Agent vs. ChemDiffAgent Behavior Analysis. We compare the qualitative behaviors of AR agents and ChemDiffAgent in representative multi-turn chemistry tasks (see Appendix F). Our observations highlight three key advantages of ChemDiffAgent: ***(1) Global Tool Selection:*** Unlike AR agents that repeat redundant tools due to local state extension, ChemDiffAgent employs global task decomposition to enable coordinated tool selection aligned with the final goal. ***(2) Multi-source Evidence Integration:*** While AR models suffer from lossy aggregation of heterogeneous data, our model utilizes slot-level fusion to integrate diverse chemical evidence into a coherent reasoning block without detail loss. ***(3) Robust Self-Refinement:*** ChemDiffAgent mitigates error propagation by validating reasoning intents and refining uncertain arguments before execution, effectively avoiding the "hallucination traps" common in AR models.

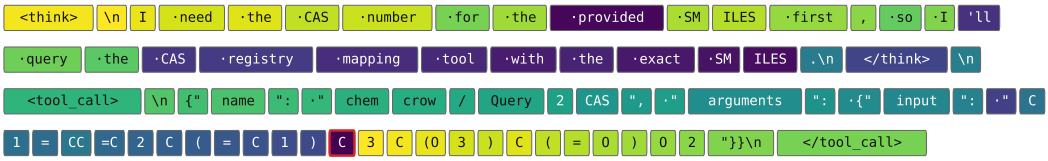

Figure 4: Heatmap showing the relative decoding order of each token within each block of an example response. The last token of each block is highlighted with a red box. For brevity, only a single turn is shown; the complete response is provided in Fig 5.

6 Conclusion and Limitations

Conclusion. We present ChemDiffAgent, the first masked diffusion language agent for tool-integrated chemical reasoning that addresses the inherent limitations of AR. By reformulating reasoning as an iterative denoising process over interleaved trajectories, ChemDiffAgent enables bidirectional context understanding and globally coordinated planning through a specialized two-stage post-training framework. Under matched supervised fine-tuning conditions, ChemDiffAgent achieves gains of approximately 20 percentage points over comparable-scale AR agents, with our 8B model achieving competitive performance against frontier models such as GPT-5.5 and Claude Opus 4.7. Behavioral analyses further demonstrate more accurate tool selection and better evidence integration across multi-step workflows. These results highlight masked diffusion as a promising foundation for scientific agents that coordinate reasoning, tool use, and evidence integration, motivating further exploration of this paradigm across broader scientific domains.

Limitations. We acknowledge several limitations of the present work. First, regarding model and domain scope, our evaluation of ChemDiffAgent is constrained to an 8B parameter scale, a fixed toolset, and purely computational chemistry workflows. While results at this scale clearly demonstrate the advantages of the diffusion paradigm over autoregressive (AR) baselines, scaling our framework to larger backbones, broader scientific domains, and wet-lab-grounded workflows remains an important direction for future research. Second, from an infrastructure perspective, diffusion-based language models (dLLMs) still face practical deployment bottlenecks. Compared to mature AR ecosystems, serving infrastructures for dLLMs—such as efficient caching, speculative decoding, and inference optimization—remain relatively underdeveloped. Moreover, training dLLMs becomes substantially more challenging as model size and agentic reasoning horizons increase, which may make AR models more practical for current production deployments. We hope our findings provide a foundation for future work to address these system-level challenges and further unlock the potential of diffusion agents in autonomous scientific discovery.

Broader Impacts

ChemDiffAgent aims to advance the development of reliable scientific reasoning agents, with the potential to accelerate chemical discovery, lower the expertise barrier for accessing computational chemistry tools, and assist researchers in tasks ranging from molecular property prediction to reaction planning. By integrating external tools rather than relying solely on parametric memory, our agent grounds its outputs in verifiable computations, which we view as a step toward more trustworthy AI assistants in scientific domains. Its ability to coordinate multiple tools may also reduce manual workflow construction and make chemistry more accessible to researchers across disciplines. Inspectable tool-use trajectories could further help researchers trace intermediate results and reproduce computational workflows. The accompanying benchmark provides a common basis for evaluating scientific tool use and may support further research into reliable diffusion-based agents.

Acknowledgments and Disclosure of Funding

This work is partially supported by the Robotic AI-Scientist Platform program of Chinese Academy of Sciences (CAS), the Strategic Priority Research Program of CAS (Grant No. XDB1580300) and the CAS Project for Young Scientists in Basic Research Grant No. YSBR-111.

References

[1] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

[3] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*, 2025.

[4] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[5] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024.

[6] Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and A. Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, 2022.

[7] Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

[8] Ernest Davis and Scott Aaronson. Testing GPT-4 with wolfram alpha and code interpreter plug-ins on math and science problems. *arXiv preprint arXiv:2308.05713*, 2023.

[9] DeepMind. Gemini diffusion, 2025.

[10] Keyan Ding, Jing Yu, Junjie Huang, Yuchen Yang, Qiang Zhang, and Huajun Chen. SciToolAgent: a knowledge-graph-driven scientific agent for multitool integration. *Nature Computational Science*, 5(10):962–972, 2025.

[11] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rStar-Math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

[12] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[13] Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

[14] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[15] Nikita Janakarajan, Tim Erdmann, Sarath Swaminathan, Teodoro Laino, and Jannis Born. Language models in molecular discovery. *arXiv:2309.16235*, 2023.

[16] Chenghao Jia, Mengdi Liu, Hong Chang, Shiguang Shan, and Xilin Chen. Mast: Motif-augmented diffusion with search tree for spectroscopic molecular structure elucidation. *arXiv preprint arXiv: 2610.12067*, 2026.

[17] Hyomin Kim, Yunhui Jang, and Sungsoo Ahn. Mt-mol: Multi agent system with tool-based reasoning for molecular optimization. *Artificial Intelligence Repository*, 2025.

[18] Mengdi Liu, Xiaoxue Cheng, Zhangyang Gao, Hong Chang, Cheng Tan, Shiguang Shan, and Xilin Chen. Protinvtree: Deliberate protein inverse folding with reward-guided tree search. *Advances in Neural Information Processing Systems*, 38:28337–28360, 2026.

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[20] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning, ICML*, 2024.

[21] Jieyu Lu and Yingkai Zhang. Unified deep learning model for multitask reaction predictions with explanation. *Journal of Chemical Information and Modeling*, 62(6):1376–1387, 2022.

[22] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[23] Yubo Ma, Zhibin Gou, Junheng Hao, Ruochen Xu, Shuohang Wang, Liangming Pan, Yujiu Yang, Yixin Cao, and Aixin Sun. SciAgent: Tool-augmented language models for scientific reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 15701–15736. Association for Computational Linguistics, 2024.

[24] Andrew D McNaughton, Gautham Krishna Sankar Ramalaxmi, Agustin Kruel, Carter R Knutson, Rohith A Varikoti, and Neeraj Kumar. CACTUS: Chemistry agent connecting tool usage to science. *ACS Omega*, 9(46):46563–46573, 2024.

[25] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *arXiv preprint arXiv:2211.00802*, 2023.

[26] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

[27] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *ArXiv preprint arXiv:2406.03736*, 2024.

[28] Siru Ouyang, Zhuosheng Zhang, Bing Yan, Xuan Liu, Yejin Choi, Jiawei Han, and Lianhui Qin. Structured chemistry reasoning with large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[29] Qwen. Qwq: Reflect deeply on the boundaries of the unknown. *URL https://qwenlm. github. io/blog/qwq-32b-preview*, 2024.

[30] Mayk Caldas Ramos, Christopher J Collison, and Andrew D White. A review of large language models and autonomous agents in chemistry. *arXiv:2407.01603*, 2024.

[31] Subham 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. In *Advances in Neural Information Processing Systems*, 2024.

[32] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

[33] Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference, 2025.

[34] Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, et al. ChemAgent: Self-updating memories in large language models improves chemical reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[35] NexusAgent Team. S1-nexusagent: a self-evolving agent framework for multidisciplinary scientific research. *arXiv preprint arXiv:2602.01550*, 2026.

[36] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[37] Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion llm. 2025.

[38] Mengsong Wu, YaFei Wang, Yidong Ming, Yuqi An, Yuwei Wan, Wenliang Chen, Binbin Lin, Yuqiang Li, Tong Xie, and Dongzhan Zhou. Chematagent: Enhancing llms for chemistry and materials science through tree-search based tool learning. *arXiv preprint arXiv:2506.07551*, 2025.

[39] Cheng Yang, Jiaxuan Lu, Haiyuan Wan, Junchi Yu, and Feiwei Qin. From what to why: A multi-agent system for evidence-based chemical reaction condition reasoning. *arXiv preprint arXiv:2509.23768*, 2025.

[40] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[41] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

[42] Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. Llasmol: Advancing large language models for chemistry with a large-scale, comprehensive, high-quality instruction tuning dataset. *arXiv preprint arXiv:2402.09391*, 2024.

[43] Botao Yu, Frazier N Baker, Ziru Chen, Garrett Herb, Boyu Gou, Daniel Adu-Ampratwum, Xia Ning, and Huan Sun. Tooling or not tooling? the impact of tools on language agents for chemistry problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7620–7640, 2025.

[44] Jiahao Zhao, Shaoxuan Xu, Zhongxiang Sun, Fengqi Zhu, Jingyang Ou, Yuling Shi, Chongxuan Li, Xiao Zhang, and Jun Xu. Dllm-searcher: Adapting diffusion large language model for search agents. *arXiv preprint arXiv:2602.07035*, 2026.

[45] Lingxiao Zhao, Xueying Ding, Lijun Yu, and Leman Akoglu. Improving and unifying discrete & continuous-time discrete denoising diffusion. *ArXiv preprint arXiv:2402.03701*, 2024.

[46] Huiling Zhen, Weizhe Lin, Renxi Liu, Kai Han, Yiming Li, Yuchuan Tian, Hanting Chen, Xiaoguang Li, Xiaosong Li, Chen Chen, et al. Dllm agent: See farther, run faster. *arXiv preprint arXiv:2602.07451*, 2026.

[47] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *ArXiv preprint arXiv:2505.19223*, 2025.

A Benchmark Construction Details

Source datasets and normalization. We construct our benchmark from four types of chemistry-oriented resources. SMolInstruct [42] provides instruction-style molecular tasks, ChemTool-Bench [38] provides tool-oriented chemistry and materials science examples, MT-Mol [17] contributes molecular optimization tasks, and T5Chem-style reaction data [21] provides reaction prediction and synthesis-related examples. Together, these resources cover six representative task families: composition inference, name conversion, property prediction, retrosynthesis, molecular optimization, and forward synthesis.

All raw instances are converted into a unified schema:

$$z = (\tau, q, x, y, t_a), \tag{12}$$

where τ is the task type, q is the natural-language query, x is the raw chemical input, y is the reference output, and t_a is the anchor tool. The anchor tool is the minimal chemistry tool required by the original task. For example, a molecular formula task is associated with a composition-related tool, a name conversion task is associated with a molecule identifier or canonicalization tool, and a reaction prediction task is associated with a forward synthesis or reaction prediction tool. This anchor tool is later used to verify whether a generated trajectory remains faithful to the original seed.

Single-turn split. The single-turn split is constructed by directly converting each normalized seed into a tool-use query. We execute a ReAct-style rollout [40] under a restricted tool environment where only the anchor tool is available. This setting prevents the model from solving the task through unrelated tools and makes the expected tool path unambiguous. A sample is retained if and only if the rollout successfully invokes the anchor tool and the final answer is consistent with the reference output.

We use task-specific equivalence rules rather than raw string matching whenever possible. For molecular structures, we canonicalize SMILES before comparison. For molecular formulas, we normalize element counts before matching. For scalar descriptors, we allow a small numerical tolerance. For classification-style predictions, we normalize labels before comparison. These rules reduce false negatives caused by superficial formatting differences while still requiring the result to be chemically equivalent. The retained examples form an execution-grounded single-tool evaluation split.

Chemistry tool dependency graph. For multi-turn construction, we first build a chemistry tool dependency graph inspired by SciToolAgent [10]. Each node represents a chemistry tool, and an edge $t_i \to t_j$ indicates that the output of tool t_i can serve as an input or useful intermediate representation for tool t_j. For example, a name resolution tool may produce a canonical SMILES string that can be consumed by property prediction, molecular optimization, or reaction analysis tools. Similarly, a reaction prediction tool may produce products that can be further analyzed by molecular property or constraint-checking tools.

Given a seed instance $z = (\tau, q, x, y, t_a)$, we sample an executable subgraph $\mathcal{G}_s$ from the dependency graph. The sampled subgraph must contain the anchor tool t_a. During sampling, we avoid degenerate single-chain expansions when possible and encourage subgraphs with branching or converging structures. This design produces tasks that require agents to coordinate multiple tools and aggregate evidence from different computational branches.

Shortcut filtering and minimal executable dependency graphs. We construct the "gold" tool-calling graph as a **minimal executable dependency graph**, rather than an arbitrary reference trajectory. We abstract shortcut filtering as a graph-search problem. For each task, we specify a starting node, the required terminal nodes, and the intermediate nodes required to complete the task. Given a sampled executable tool-use path containing m tool calls, we check whether the dependency graph contains a valid path with fewer than m calls that still covers all required terminal and intermediate nodes.

Specifically, starting from the initial node, we perform a bounded breadth-first search (BFS) over depths 1 to $m - 1$. This enumerates the executable tool-use paths that can be completed with fewer than m calls. We then check whether any such path contains all predefined required terminal and intermediate nodes. If such a path exists, we consider it a shortcut and rewrite or remove the

corresponding task; otherwise, the original task is retained. Since our tasks contain at most seven tool calls, this bounded BFS can be performed efficiently. This filtering makes the stored gold graph a minimal executable dependency structure and reduces ambiguity from redundant alternative trajectories.

Graph-conditioned query rewriting. After sampling $\mathcal{G}_s$, we use a strong teacher model to rewrite the original query into a graph-conditioned chemistry-agent task. The teacher receives the original seed input–output pair and the sampled tool subgraph. It is instructed to satisfy three constraints. First, the rewritten query must preserve the semantic core of the original seed task. Second, solving the rewritten query should require the tools in $\mathcal{G}_s$, rather than merely mentioning them. Third, the final answer should remain grounded in the original reference output through the anchor tool.

This rewriting step turns static chemistry examples into executable multi-step agent tasks. For molecule-centered seeds, the generated task may require resolving molecular identifiers, computing complementary properties, comparing optimized candidates, and synthesizing a final decision. For reaction-centered seeds, the generated task may require predicting reaction outcomes, analyzing products, checking constraints, and producing a consolidated conclusion. Thus, the generated tasks require intermediate representation conversion, multi-step chemical reasoning, and tool-result aggregation.

Execution verification. We verify each generated multi-turn task by executing it with a ReAct-style rollout in a restricted environment containing only the tools in $\mathcal{G}_s$. The restricted tool set reduces interference from irrelevant tools and ensures that a valid trajectory must follow the sampled dependency structure. A candidate is retained only if it satisfies all of the following conditions:

1. the anchor tool t_a is invoked;
2. the runtime input to t_a matches the original raw input x after task-specific normalization;
3. the output of t_a matches the original reference output y under the corresponding equivalence rule;
4. the final answer is consistent with the verified tool evidence.

Candidates that fail any of these checks are discarded. This filtering step is important because a rewritten query may otherwise drift away from the original seed, use the sampled tools only superficially, or produce a trajectory that is not executable in the tool environment.

Resulting benchmark. The resulting benchmark contains two complementary settings. The single-turn split evaluates whether an agent can select and execute the correct chemistry tool for a canonical task. The multi-turn split evaluates whether an agent can coordinate multiple interdependent tools, track intermediate states, and synthesize a final answer. Each retained instance contains an execution-verified final answer. For multi-turn examples, we additionally store the expected calling chain or tool tree, which supports evaluation of tool selection, intermediate execution faithfulness, and long-horizon tool-use behavior.

B Evaluation Metrics

Answer correctness. We evaluate final-answer correctness using GPT-4o as the judge. Given a query Q, a reference answer $A^\star$, and a model prediction $\hat{A}$, the judge determines whether $\hat{A}$ correctly solves the task with respect to $A^\star$. The answer correctness score for the i-th example is defined as

$$s_i^{\text{ans}} = \mathbb{I}\big[\text{Judge}(Q_i, A_i^\star, \hat{A}_i) = \text{correct}\big], \tag{13}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The final answer accuracy over a dataset $\mathcal{D}$ is computed as

$$\text{Acc}_{\text{ans}} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} s_i^{\text{ans}}. \tag{14}$$

Tool selection accuracy. Each example is associated with a gold calling chain or tool-calling sequence tree constructed as a minimal executable dependency graph through the shortcut filtering described in Appendix A. Thus, the gold graph is not an arbitrary reference trajectory: an alternative route to the same final answer would normally involve redundant tool calls under the benchmark construction. We extract the expected tool set from this minimal gold structure and compare it with the tools invoked by the model. Let

$$\mathcal{T}_i^\star = \{t_{i,1}^\star, t_{i,2}^\star, \ldots, t_{i,m_i}^\star\} \tag{15}$$

denote the set of m_i expected tools for the i-th example, and let $\hat{\mathcal{T}}_i$ denote the set of tools invoked by the model during its rollout. The tool selection score for this example is computed as

$$s_i^{\text{tool}} = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathbb{I}\big[t_{i,j}^\star \in \hat{\mathcal{T}}_i\big]. \tag{16}$$

This metric only checks whether each expected tool name appears in the model trajectory at least once. It does not evaluate tool arguments, execution outputs, or the exact order of tool calls. Answer correctness remains the primary success criterion. If an agent nevertheless discovers a correct executable solution with fewer tool calls than the stored gold structure, the answer is still considered valid and the shorter trajectory is considered more efficient; the tool-selection score is used only as a complementary diagnostic of whether the expected dependencies were covered.

The overall tool selection accuracy is the average score over all examples:

$$\text{Acc}_{\text{tool}} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} s_i^{\text{tool}}. \tag{17}$$

Grouped evaluation by trajectory length. For multi-turn analysis, we group examples according to the length of the gold calling chain. In the main experiments, Easy, Medium, and Hard correspond to 2, 3–5, and 6–7 expected tool-calling rounds, respectively. For each group $\mathcal{D}_g \subset \mathcal{D}$, we compute answer accuracy and tool selection accuracy as

$$\text{Acc}_{\text{ans}}(g) = \frac{1}{|\mathcal{D}_g|} \sum_{i \in \mathcal{D}_g} s_i^{\text{ans}}, \tag{18}$$

and

$$\text{Acc}_{\text{tool}}(g) = \frac{1}{|\mathcal{D}_g|} \sum_{i \in \mathcal{D}_g} s_i^{\text{tool}}. \tag{19}$$

The total score is computed over all evaluated examples rather than by taking an unweighted average over groups.

C Implementation Details

All experiments are conducted on 8 NVIDIA A100 GPUs. ChemDiffAgent uses SDAR-8B as the diffusion language model backbone. For the main inference setting, we set the block size and the number of denoising steps to 64. The maximum number of tool-calling rounds is set to 7 unless otherwise specified.

For Agentic SFT, we train SDAR-8B on the constructed chemistry-agent trajectories using the Agentic ELBO objective. Only the agent-generated `think` and `tool_call` regions contribute to the training loss, while external `tool_response` regions are excluded. We train with AdamW [19] using learning rate 1×10^{-5}. The maximum prompt length is set to 4096, and the maximum generation length is set to 8192.

For Agentic VRPO, we construct preference pairs from failed trajectories generated by the SFT model and corresponding correct trajectories verified by tool execution. We use the Agentic ELBO to estimate trajectory likelihoods for both the policy and reference models. The preference scaling coefficient is set to $\beta = 0.1$, and training uses AdamW with learning rate 5×10^{-7}. For likelihood estimation, we use 2 Monte Carlo samples. The maximum prompt length, completion length, and total sequence length are set to 3072, 2560, and 6144, respectively.

For evaluation, all non-agent baselines are run under the same ReAct-style tool-use protocol with access to the required chemistry tools. Open-source chemistry agents are evaluated with their original execution settings. Answer correctness is judged by GPT-4o, and tool selection accuracy is computed from the expected tool-calling sequence as described in Appendix B.

D Theoretical Analysis of Flexible Ordering

We provide a detailed theoretical analysis of the advantage of flexible generation order in DLLMs over fixed autoregressive (AR) decoding. In structured reasoning, different subproblems can have different conditional difficulties and downstream risks, and resolving one part may provide useful information for another. Flexible ordering allows a DLLM to exploit these dependencies by choosing how partially resolved states are traversed, rather than committing to a predefined left-to-right sequence. We first show that the BDLM objective provides training support for block-level infilling. We then analyze two complementary advantages of flexible decoding: an adaptive path advantage under non-uniform subproblem difficulty and a strict ordering advantage under non-uniform decision risk.

D.1 Block-Level Infilling Support

We first establish that the BDLM training objective explicitly provides learning signals over different block-level mask configurations. This provides the training basis for resolving different parts of the output under partially observed completion states.

Derivation of Eq. 4. Let the clean completion y be partitioned into K contiguous blocks $[y^1, \dots, y^K]$, where each block $y^k = (y^{k,1}, \dots, y^{k,B})$ has length B and $L = KB$. For a non-empty block mask set $\emptyset \neq A \subseteq [K]$, let $y[A]$ denote the corrupted completion obtained by replacing all tokens in the blocks indexed by A with the mask token M.

Consider a discrete block-masking process that first samples the number of masked blocks $m \in \{1, \dots, K\}$ and then uniformly samples a subset $A \subseteq [K]$ with $|A| = m$. Conditioned on m, each block mask set has probability

$$\tilde{q}(A \mid |A| = m) = \frac{1}{\binom{K}{m}}. \tag{20}$$

For a fixed corrupted completion $y[A]$, the reconstruction loss averages over the tokens in the masked blocks:

$$\frac{1}{|A|B} \sum_{k \in A} \sum_{i=1}^{B} \log p_\theta \left(y^{k,i} \mid y[A], y^{<k}, x \right). \tag{21}$$

Expanding the expectation over the block-masking process gives

$$\mathcal{L}_\theta^{\text{block}} = \mathbb{E}_{(x,y)\sim p_{\text{data}}} \left[\sum_{m=1}^{K} \frac{1}{K} \sum_{\substack{A \subseteq [K] \\ |A|=m}} \frac{1}{\binom{K}{m}} \frac{1}{mB} \sum_{k \in A} \sum_{i=1}^{B} \log p_\theta \left(y^{k,i} \mid y[A], y^{<k}, x \right) \right]. \tag{22}$$

Since $m = |A|$, this can be equivalently written as

$$\mathcal{L}_\theta^{\text{block}} = \mathbb{E}_{(x,y)\sim p_{\text{data}}} \left[\sum_{\emptyset \neq A \subseteq [K]} \frac{1}{K} \frac{1}{\binom{K}{|A|}} \frac{1}{|A|B} \sum_{k \in A} \sum_{i=1}^{B} \log p_\theta \left(y^{k,i} \mid y[A], y^{<k}, x \right) \right]. \tag{23}$$

Thus, up to the global normalization constant $1/K$, the BDLM objective is a linear combination of reconstruction losses over all non-empty block-level mask sets $A \subseteq [K]$. The model therefore receives supervision over a family of partially observed completion states, providing the block-level infilling capability required for flexible generation order at inference time.

D.2 Adaptive Path Advantage under Non-Uniform Subproblem Difficulty

Structured reasoning contains subproblems with different conditional difficulties, and resolving one part may simplify another. Let X be the input history, $Y = (Y_1, \dots, Y_L)$ the output, S the resolved

positions, and p and q_θ the data and model distributions. We define the prediction difficulty of Y_i as its conditional cross-entropy:

$$\begin{aligned} d_\theta(i \mid X, Y_S) &:= \mathbb{E}_{Y_i \sim p(\cdot \mid X, Y_S)} \left[-\log q_\theta(Y_i \mid X, Y_S) \right] \\ &= H(Y_i \mid X, Y_S) + \delta_\theta(i \mid X, Y_S), \end{aligned} \tag{24}$$

where

$$\delta_\theta(i \mid X, Y_S) = D_{\mathrm{KL}} \left(p(Y_i \mid X, Y_S) \| q_\theta(Y_i \mid X, Y_S) \right). \tag{25}$$

Here, H is the conditional entropy and δ_θ is the model's approximation error.

Let $\pi = (i_1, \ldots, i_L)$ be a generation path and $S_t = \{i_1, \ldots, i_{t-1}\}$. Its expected loss can be written as

$$\mathcal{L}_\theta(\pi) = H(Y \mid X) + \mathbb{E}_p \left[\sum_{t=1}^{L} \delta_\theta(i_t \mid X, Y_{S_t}) \right]. \tag{26}$$

The entropy term $H(Y \mid X)$ is invariant to the generation order, while the accumulated approximation error depends on the intermediate states induced by the path. AR follows the fixed path $\pi_{\mathrm{AR}} = (1, \ldots, L)$, whereas DLLM decoding can select from a larger set of paths Π_{DLLM} according to the current partially resolved state. Therefore,

$$\mathcal{L}^*_{\mathrm{DLLM}} = \inf_{\pi \in \Pi_{\mathrm{DLLM}}} \mathcal{L}_\theta(\pi) \leq \mathcal{L}_\theta(\pi_{\mathrm{AR}}) = \mathcal{L}_{\mathrm{AR}}. \tag{27}$$

This gives DLLMs an adaptive path advantage under non-uniform subproblem difficulty. When some unresolved parts are easier to predict, or when resolving them provides useful context for harder parts, a DLLM can choose a path with smaller accumulated approximation error by resolving these parts first. In contrast, AR must commit to the next position in its fixed order even when that position remains difficult under the currently available information.

D.3 Strict Ordering Advantage under Non-Uniform Decision Risk

Prediction difficulty alone does not capture the fact that errors at different positions can have different downstream consequences. We therefore further consider non-uniform decision risk. Let C denote the available context, Z an earlier high-risk decision in the fixed AR order, and R a later lower-risk decision that provides information about Z. Let $w_Z, w_R > 0$ denote their expected downstream error costs.

Under execution-weighted conditional log loss, the AR order $Z \to R$ has risk

$$\mathcal{J}_{\mathrm{AR}} = w_Z H(Z \mid C) + w_R H(R \mid C, Z), \tag{28}$$

whereas resolving R before infilling Z gives

$$\mathcal{J}_{R \to Z} = w_R H(R \mid C) + w_Z H(Z \mid C, R). \tag{29}$$

Their difference is

$$\mathcal{J}_{\mathrm{AR}} - \mathcal{J}_{R \to Z} = (w_Z - w_R) I(Z; R \mid C). \tag{30}$$

Consequently, if

$$I(Z; R \mid C) > 0 \qquad \text{and} \qquad w_Z > w_R, \tag{31}$$

then

$$\mathcal{J}^*_{\mathrm{DLLM}} \leq \mathcal{J}_{R \to Z} < \mathcal{J}_{\mathrm{AR}}. \tag{32}$$

Thus, flexible ordering yields a strict advantage when a later, lower-risk decision provides information about an earlier, higher-risk decision. Rather than committing immediately to the high-risk position, a DLLM can first resolve the more reliable decision and then use the resulting information to reduce uncertainty at the more consequential one.

Implications for chemistry reasoning. Tool use and molecular generation involve decisions with different conditional difficulties and downstream consequences. Flexible ordering allows these differences to influence the order of token resolution within each block, while preserving the sequential interaction between the agent and its tools.

For tool use, some components of a call may be directly supported by the prompt or previous observations, whereas others require further reasoning. For example, an available molecular identifier or an explicit property constraint may be easier to determine than an uncertain tool choice or parameter. When these components occur in the same block, a DLLM can resolve informative content first and use it as context for the remaining positions. This illustrates the adaptive path advantage: different intermediate contexts can lead to different accumulated prediction errors. The consequences of errors also vary. A mistaken tool name, molecular argument, or consequential parameter can change the computation performed and affect subsequent reasoning. When lower-risk content provides information about such a decision, resolving it first can reduce uncertainty before the call is finalized and executed.

For SMILES generation, local token choices participate in structural constraints involving atom identities, bonds, branches, and ring closures. Some positions may be strongly constrained by a reference molecule or the requested modification, while others admit several plausible choices. Within a block, flexible ordering can first resolve well-supported structural elements and then use the resulting partial structure to guide uncertain positions. These decisions can also have unequal consequences: an incorrect bond or ring-closure token may invalidate the representation or alter molecular connectivity, while other choices may have a smaller effect on the task objective. Deferring a consequential position is therefore potentially useful when other resolvable positions provide relevant information.

The two analyses capture complementary benefits of this flexibility. The adaptive path analysis concerns how intermediate contexts affect model approximation error, whereas the risk-weighted analysis concerns where information is most valuable given unequal error costs. An easier decision is not necessarily a lower-risk decision; a useful ordering must account for both its informational contribution and its consequences. Together, these analyses motivate flexible within-block resolution and establish conditions for improvements in prediction loss and, under Bayes-optimal conditional predictions, risk-weighted log loss.

E Denoising Visualization of Tool-use Decisions

Interpreting the denoising order. Figure 5 shows a representative tool-calling block generated by ChemDiffAgent. Unlike autoregressive models that commit to tokens strictly from left to right, ChemDiffAgent generates the whole block through iterative denoising. The color of each token indicates the relative step at which the token is resolved within the block: lighter tokens are determined earlier, while darker tokens are determined later. This allows us to inspect not only the final generated text, but also the temporal structure of the model's decision process.

The visualization reveals a non-left-to-right generation pattern. Parts of the action structure, such as tool-call delimiters and argument formatting, are resolved relatively early, while certain tokens within the SMILES argument are resolved at later stages. It reflects the flexibility of masked diffusion decoding, which allows individual token predictions to be conditioned on information resolved elsewhere within the same block. Consequently, ChemDiffAgent can determine interdependent components of a tool call in a flexible order instead of committing to each token sequentially.

Why this behavior is beneficial for chemistry-agent tasks. Tool use in chemistry agents is not merely a matter of selecting a tool name. A valid action must jointly satisfy multiple constraints: the control token must indicate an external action, the tool namespace must match the executor, the argument schema must be valid, and the chemical string must be exact. For example, a CAS-query tool call requires not only selecting the correct CAS lookup tool, but also passing a valid molecular representation in the expected argument field. If any part of this structured action is inconsistent, the tool call may fail even when the model has selected the semantically correct tool.

The denoising pattern in Figure 5 suggests that ChemDiffAgent handles this structured decision in a natural way. By resolving the action scaffold early, the model establishes a stable context for the entire tool call. By resolving some argument tokens later, it avoids committing too early to uncertain



Figure 5: Token-level denoising order of a representative ChemDiffAgent tool-calling block. Lighter colors indicate tokens resolved earlier, while darker colors indicate tokens resolved later.

contents before the surrounding action format is fixed. This is particularly useful for chemical strings such as SMILES, where small token-level mistakes can change the molecule or make the input invalid. Thus, the model can condition local argument refinement on the already-formed global tool-use structure, improving the chance that the final action is executable.

Comparison with autoregressive generation. Autoregressive agents generate tool calls token by token. Once an AR model emits a tool name, an opening argument field, or a partial chemical string, subsequent tokens must be generated conditioned on these previous commitments. This left-to-right commitment can be problematic for tool-use agents because different parts of an action are mutually dependent. The correct tool may depend on the intended argument type, the argument may depend on the selected tool schema, and the stopping or control decision depends on whether the action is already complete. However, AR generation forces these decisions to be serialized.

ChemDiffAgent instead performs block-level generation. Within a block, the tool name, control tokens, argument schema, and chemical string can be optimized together through denoising. This makes the action less like a rigid left-to-right string and more like a jointly formed structured object. The model can first settle high-confidence structural tokens and then refine lower-confidence argument tokens while still conditioning on the surrounding context. This delayed refinement reduces premature commitment and provides a mechanism for aligning the tool-use decision with the model's current knowledge state.

Connection to decision boundary alignment. This visualization supports our view that agent behavior is a decision-making process rather than simple tool calling. An agent must decide when internal reasoning is sufficient, when external tools are needed, which tool should be used, and how to construct a valid executable action. In this sense, the model must align its decision boundary—when and how to act—with its knowledge boundary—what information it can reliably infer internally. The observed denoising order is consistent with this interpretation. ChemDiffAgent first forms the action-level structure and then refines uncertain chemical details, suggesting that external actions are

committed only after a stable tool-use context has emerged. This block-level refinement provides a structural advantage over autoregressive generation for chemistry-agent tasks, where successful execution depends on the consistency between reasoning intent, tool selection, argument format, and exact chemical input.

F Case Studies

We provide representative case studies to illustrate qualitative differences between autoregressive agents and ChemDiffAgent. These cases focus on three common failure modes in chemistry-agent tasks: tool selection, information aggregation, and error correction.

Case Study 1: Tool Selection

Problem.

Given a chemical compound name, the agent needs to first retrieve its PubChem CID, then use the CID to obtain the molecular formula, and finally compute the mass percentage of a specified element in that formula.

AR Agent

Reasoning Style: Local Tool Continuation

- Starts from the compound name and selects a lookup-style tool.
- After obtaining an intermediate identifier X, it continues to call a similar name/CID lookup tool.
- Repeatedly searches around the same identifier instead of switching to the formula-conversion tool.
- Fails to enter the required computation stage:
  Compound Name $\rightarrow$ CID $X \nrightarrow$ Molecular Formula $Y \nrightarrow$ Element Percentage Z
- The trajectory terminates with incomplete or irrelevant intermediate information.

DLLM Agent

Reasoning Style: Global Task Decomposition Before Execution

- First identifies that the question contains three distinct tool stages:
  - entity resolution
  - identifier-to-formula conversion
  - formula-based numerical computation
- Builds a global dependency plan:
  Compound Name $\rightarrow$ CID $X \rightarrow$ Molecular Formula $Y \rightarrow$ Element Percentage Z
- Selects a different tool for each semantic subtask.
- Avoids repeatedly querying the same identifier-level tool.
- Produces the final answer only after all required variables are available.

Why AR Fails. The AR agent makes tool decisions based mainly on the latest visible state. After obtaining X, it remains trapped in the entity-lookup stage and keeps selecting locally similar tools, rather than globally recognizing that the next required operation is formula conversion followed by numerical calculation.

Case Study 2: Information Summarization

Problem.

For a molecule represented by a SMILES string, the agent needs to collect several heterogeneous properties, including predicted biological activity, solubility, molecular descriptors, functional groups, and a long side-effect profile, and return them as a single structured JSON object with fixed keys.

AR Agent

Reasoning Style: Sequential Accumulation with Fragile Memory

- Queries property A, then property B, then property C in sequence.
- Correctly obtains several intermediate results:
  $A, B, C, D, \ldots$
- Encounters a verbose tool response for side effects, containing many category-probability pairs.
- During final assembly, compresses the long response too aggressively.
- Some values are converted into inconsistent formats:
  - textual Yes/No labels become booleans
  - numeric strings become rounded floats
  - one predicted probability summary is partially omitted
- The final JSON is structurally plausible but does not faithfully preserve all required information.

DLLM Agent

Reasoning Style: Latent Slot-Level Aggregation

- Before answering, constructs a slot table for the required output keys:
  $$\{k_1, k_2, k_3, \ldots, k_n\}$$
- Associates each queried tool output with a specific slot.
- Treats the long side-effect profile as a structured field rather than as disposable explanation text.
- Maintains type and key consistency across the final answer:
  - activity labels remain aligned with the expected schema
  - descriptor dictionaries remain nested
  - verbose side-effect information is preserved
- Produces a complete JSON object after all slots have been filled.

Why AR Fails. The AR agent can collect information step by step, but its final answer depends heavily on the last few visible states. When the output contains many heterogeneous fields and a long textual profile, AR is prone to lossy summarization, schema drift, and value-type mismatch. DLLM performs better because it maintains a global latent representation of the required output slots.

Case Study 3: Error Correction

Problem.

Given a reaction description, the agent must predict the major product, verify or canonicalize the generated product representation, and then compute several downstream molecular properties based on the corrected product.

AR Agent

Reasoning Style: Commit Early and Continue

- Predicts a candidate product P_0 from the reaction.
- Treats P_0 as reliable and immediately uses it for downstream property tools.
- Some downstream calls return unstable or inconsistent behavior because P_0 is noisy.
- Instead of revising the product, the agent retries similar property queries.
- The trajectory becomes:
  Reaction $\rightarrow$ P_0 $\rightarrow$ Property Query $\rightarrow$ Retry $\rightarrow$ Retry
- Final answer is based on an uncorrected product representation.

DLLM Agent

Reasoning Style: Prediction–Validation–Correction

- Predicts an initial product candidate P_0.
- Before computing downstream properties, checks whether P_0 should be validated or canonicalized.
- Detects that the intermediate product representation may be unreliable.
- Applies a correction step:

$$P_0 \rightarrow P_1$$

  where P_1 is the validated/canonical product.
- Uses P_1, not P_0, for all subsequent property computations.
- Produces a final answer consistent with the corrected molecular representation.

Why AR Fails. The AR agent commits to the first generated intermediate result and lacks a strong mechanism for revising earlier choices. Once the wrong or noisy product P_0 enters the context, later steps inherit the error. DLLM is better in this case because its global planning allows an explicit correction stage between product prediction and downstream property computation.

Summary. Across the three cases, we observe that the main difference between AR agents and DLLM agents lies in how they organize intermediate decisions. AR agents tend to extend the trajectory from the most recent visible state, which makes them effective for simple local continuation but vulnerable to tool-selection traps, lossy aggregation, and early error propagation. In contrast, DLLM agents can form a more global latent plan before committing to individual actions. This allows them to coordinate tool dependencies, preserve heterogeneous information in structured outputs, and insert validation or correction steps before downstream computation. These qualitative observations are consistent with our quantitative results: ChemDiffAgent achieves stronger tool selection accuracy and more stable multi-turn performance, especially in tasks requiring long-range coordination across multiple tools.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims made, including the contributions made in the paper and important assumptions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code and pre-trained models will be made publicly available upon acceptance of the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets used in this work—including pretrained models, datasets, and software libraries—are properly credited in the main text or references.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as part of the core methodology in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.