HomeFlow: A Data Flywheel for Smart Home Agent Training with Verifiable Simulation

Yi Gu[1,*] **Huacan Wang**[1,*,†,‡] **Shuo Zhang**[1,2,*,◇] **Yuqing Hou**[1,*,†]
Lei Xue[1] **Weipeng Ming**[1] **Chen Liu**[1] **Fangzhou Yu**[1,3,◇]
Kuan Li[1] **Ronghao Chen**[4] **Sen Hu**[4] **Xiaofeng Mou**[1] **Yi Xu**[1,†]

[1]AIRC, Midea Group [2]Beijing University of Posts and Telecommunications
[3]Donghua University [4]Peking University

[*]Equal Contribution, [†]Corresponding Author, [‡]Project Leader.

[◇]Work completed during an internship at Midea AI Research Center.

Abstract

Large language model agents are moving beyond text-only interaction toward physical-world control, with smart homes as a representative domain. Real domestic interaction requires understanding ambiguous intents, operating in dynamic environments, and performing multi-turn reasoning. However, existing methods struggle to generate high-quality training data for smart home agents. We propose **HomeFlow**, a verifiable data flywheel for this domain. HomeFlow uses HomeEnv as a unified simulation environment and HomeMaker to procedurally generate diverse home settings. Subsequently, Blueprint compiles open-ended user intents into executable state-based success conditions, while MCTS-Flow synthesizes diverse, verifiable multi-turn trajectories through environment-guided tree search. We then optimize the agents via supervised fine-tuning and step-wise RLVE, which facilitates iterative improvement through authentic physical feedback. We further construct SmartHome-Bench to evaluate the agent across various smart home tasks. On this benchmark, HomeFlow-RL-4B and HomeFlow-RL-8B achieve task success rates of 84.60% and 87.03%. It is worth noting that HomeFlow-RL-8B even surpasses the leading GPT-5.5 by 1.23 percentage points.

1 Introduction

Large language model (LLM)-based agents [1] are increasingly moving beyond text-only interaction [2, 3] toward control in the physical world [4]. Smart homes [5, 6] provide a natural testbed for this transition: they are rich in connected devices, governed by physical states, and closely aligned with everyday user needs. A capable smart home agent must infer implicit user intents, act on the basis of home states, and seek clarification when needed. For example, "*it feels stuffy*" may require adjusting the air conditioner or ventilation depending on the current environment. However, directly deploying frontier LLMs as smart home agents is impractical due to their lack of situated awareness. This often leads to seemingly plausible but unexecutable commands, requiring domain-specific training to align models with smart home dynamics.

Such training depends on large-scale multi-turn interaction data, which is difficult to collect from real households due to cost, deployment complexity, and privacy constraints. A common alternative is to synthesize dialogues offline through LLM-based role play [7, 8]. However, text-only synthesis has three fundamental limitations. First, it lacks an underlying physical environment against which actions can be verified. For instance, whether a generated command like "*lowering the air conditioner*"

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

maps to a valid device and is actually executable in the current room cannot be determined from text alone, forcing evaluation to rely on subjective LLM-as-judge [9, 10] signals. This introduces noise and prevents reinforcement learning from receiving physically grounded rewards. Second, linear role play typically produces a single trajectory for each scenario, offering limited coverage of diverse environmental evolutions and user responses. Third, training on static offline dialogues mainly supports single-step alignment under fixed contexts [11, 12, 13]. Without explicit multi-turn training, agents struggle to track intent and correct their policies over full episodes.

To address these limitations, we propose **HomeFlow**, a verifiable data flywheel for training smart home agents. Its core idea is to construct an interactive simulation environment, HomeEnv, serving as a unified grounded signal source that bridges data generation, trajectory expansion, and multi-turn policy optimization. Built upon HomeEnv, HomeMaker is designed to procedurally generate diverse home layouts, devices, and environmental states, populating the simulation environment. For verifiable data generation, we introduce a pipeline comprising Blueprint and MCTS-Flow. Specifically, Blueprint constructs comprehensive interaction scenarios that incorporate diverse user profiles and contexts, and compiles implicit intents into executable logical constraints. Guided by these blueprints, MCTS-Flow advances data synthesis from one-shot linear sampling to tree-structured search, exploring and verifying diverse multi-turn trajectories.

Finally, we adopt a two-stage training paradigm. After an initial supervised fine-tuning phase, the agent is further optimized through Step-wise Reinforcement Learning from Verifiable Execution (RLVE) [13]. Unlike traditional approaches that only reward the final outcome, this strategy continuously optimizes the agent during online interactions with a dynamic LLM user and HomeEnv, applying physically grounded rewards at every step of the generation process.

To evaluate the agent's ability to interact with users and operate devices in realistic smart home scenarios, we construct SmartHome-Bench on top of HomeEnv. On this benchmark, HomeFlow-RL-4B and HomeFlow-RL-8B, trained from Qwen3-4B and Qwen3-8B [14] using HomeFlow, achieve overall task success rates of 84.60% and 87.03%. Notably, HomeFlow-RL-8B outperforms GPT-5.5 by 1.23%, while HomeFlow-SFT-4B and HomeFlow-SFT-8B improve over the corresponding text-only role-play fine-tuning baselines by 5.88% and 4.67%, respectively.

2 Related Work

Smart Home Agents. LLM have emerged as central orchestrators for smart homes, grounding natural language intents into executable API calls [15, 16, 17]. Beyond early systems [18, 19] focusing on basic instruction decomposition, recent works [16, 20, 21] explore personalization, persistent memory, and edge-based multi-agent orchestration. Concurrently, benchmarks such as HomeBench [22], SimuHome [23] and SMH-Bench [24] have significantly driven the development of this field by assessing LLMs on instruction-to-call generation, temporal reasoning, and environment-aware execution.

Dialogue Data Synthesis. Synthetic data generation provides a scalable approach for model post-training. Methods like Self-Instruct [25] and Evol-Instruct [26] bootstrap and iteratively rewrite instructions, while AgentInstruct [27] extends this to complex agentic workflows. For dialogue agents, multi-turn trajectories are typically constructed using user simulators and role-playing, often filtered by LLM-as-a-Judge or calibration [9, 10, 28, 29]. Recent search-based methods [30, 31] further integrate tree-structured exploration to enhance trajectory diversity and quality for downstream training.

Reinforcement Learning from Verifiable Environments. While RLHF [32] and RLAIF [33, 34] rely on preference-based rewards, recent trends favor verifiable rewards (RLVR) where correctness is automatically checkable, showing strong potential in mathematics (e.g., GRPO [35]) and interactive agent environments [36, 37]. Building on these foundations, HomeFlow formulates smart home task as a stateful learning problem. It leverages verifiable data synthesis to provide a robust cold start, followed by step-wise RLVE optimized through structured, environment-based rewards.

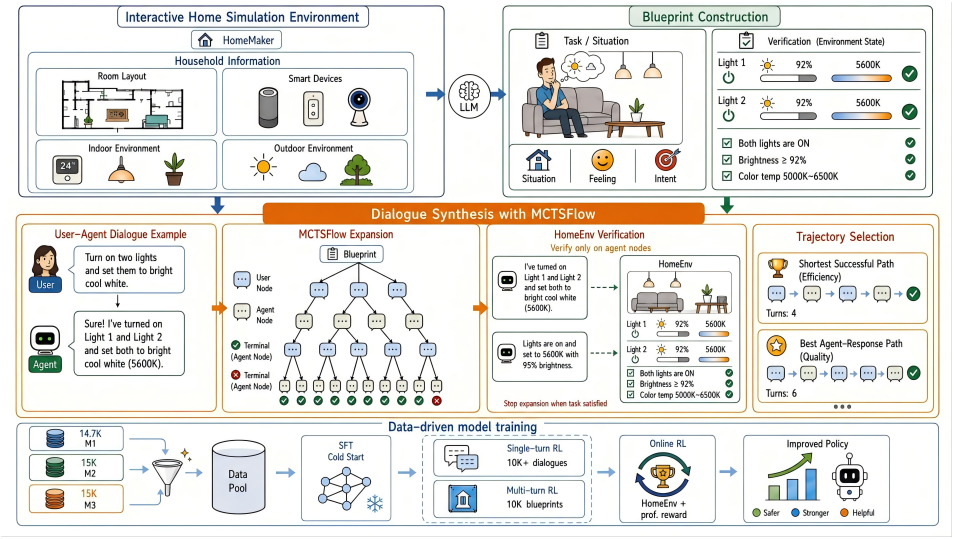

Figure 1: Overview of HomeFlow. Based on HomeEnv, HomeMaker generates diverse home environments. Blueprint then creates executable scene conditions, followed by MCTS-Flow to synthesize diverse, verifiable trajectories. From this architecture, we derive three data synthesis strategies: (M1) pure-text role-play (w/o Blueprint); (M2) Blueprint-grounded linear generation (w/o MCTS-Flow); and (M3) the complete HomeFlow pipeline.

3 Method

3.1 Task Definition

Smart A tasks can be viewed as goal-directed sequential decision making in a partially observable physical environment. We formalize the task as a goal-conditioned Partially Observable Markov Decision Process (POMDP) in discrete time, represented by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{E}, \mathcal{O}, \mathcal{R} \rangle$.

The state space $\mathcal{S}$ captures the underlying physical state of the home. At time step t, the state $s_t \in \mathcal{S}$ encodes continuous environmental and device attributes (e.g., temperature), alongside discrete device states (e.g., on/off, mode). To enable bidirectional interaction with both the environment and the user, the agent acts in a heterogeneous action space$\mathcal{A} = \mathcal{A}_{\text{NL}} \cup \mathcal{A}_{\text{Exec}}$, where $\mathcal{A}_{\text{NL}}$ denotes natural language actions used for responses and clarification, and $\mathcal{A}_{\text{Exec}}$ denotes executable control actions, such as generating Python script snippets or API calls to query or modify the physical state.

The environment dynamics are governed by the transition function $\mathcal{E} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. An executable action $a_t \in \mathcal{A}_{\text{Exec}}$ induces a deterministic physical state transition according to $s_{t+1} = \mathcal{E}(s_t, a_t)$, whereas a natural language action $a_t \in \mathcal{A}_{\text{NL}}$ only advances the dialogue context, leaving the physical state unchanged (i.e., $s_{t+1} = s_t$). Because the agent lacks direct access to the full state s_t, it makes decisions based on partial observations $o_t \in \mathcal{O}$. The observation o_t may include the user utterance, device-query results, and feedback from executed actions.

To avoid relying on subjective LLM-based evaluation, we represent the user's task goal as a set of verifiable conditions $\Phi = \{\phi_1, \phi_2, \ldots, \phi_K\}$, where each $\phi_k : \mathcal{S} \to \{0, 1\}$ is a boolean predicate over the environmental state. Based on this, given an initial state $s_0 \sim p(s_0)$ and a goal specification $\Phi \sim p(\Phi)$, the agent policy $\pi_\theta(a_t \mid h_t)$ acts upon the interaction history $h_t = (o_1, a_1, \ldots, o_t)$ to generate a complete trajectory τ. Upon reaching the final state s_T at the end of the episode, the terminal verification reward $\mathcal{R}$ is defined by the cumulative satisfaction of all target conditions as $\mathcal{R}(s_T, \Phi) = \sum_{k=1}^{K} \phi_k(s_T)$. This additive formulation provides a denser reward signal by granting partial credit for incomplete tasks. Therefore, the overall objective is to find the optimal policy π_θ^* that maximizes the expected verification success rate:

$$J(\theta) = \mathbb{E}_{\substack{s_0 \sim p(s_0),\, \Phi \sim p(\Phi) \\ \tau \sim p(\tau \mid s_0, \Phi, \pi_\theta)}} \left[\mathcal{R}(s_T, \Phi) \right]. \tag{1}$$

This formulation highlights two core challenges in training smart home agents: (1) building an interactive execution engine to simulate the environment and compute verifiable rewards (Section 3.2); and (2) compiling open-ended scenarios into precise goals to drive efficient exploration and subsequent policy optimization (Section 3.3 and Section 3.4).

3.2 Interactive Home Environment

We develop HomeEnv, an interactive environment designed for scalable agent training. To diversify the initial state distribution $p(s_0)$, we further introduce HomeMaker, a procedural environment generator. HomeEnv and HomeMaker provide the core environment foundation for HomeFlow.

3.2.1 HomeEnv: Smart Home Simulation Engine

HomeEnv serves as the verifiable execution platform throughout data generation and policy optimization. It provides unified execution feedback, partial observations, and rule-based verification. Its design comprises three main components. More details are provided in Appendix A.

Structured state representation. We instantiate the home state $s_t \in \mathcal{S}$ as

$$s_t = \langle \mathcal{L}, \mathcal{D}, \psi, \mathcal{X}_t, \mathcal{F} \rangle, \tag{2}$$

where $\mathcal{L}$ denotes the set of rooms and their spatial organization, $\mathcal{D}$ is the set of smart devices, and $\psi : \mathcal{D} \to \mathcal{L}$ maps each device to its location. For each device $d \in \mathcal{D}$, $\mathcal{X}_t(d)$ captures its dynamic states, while $\mathcal{F}(d)$ specifies its exposed executable functions. This declarative device schema decouples state representation from simulation implementation, enabling the environment to scale across heterogeneous and long-tail IoT devices.

Execution sandbox. HomeEnv provides an isolated sandbox for executing actions $a_t \in \mathcal{A}_{\text{Exec}}$. When the agent generates code or API calls, the sandbox returns observations o_t restricted to the direct execution results—such as serialized device states, execution feedback, and runtime errors. This design enforces partial observability, requiring the agent to actively query the environment and interpret feedback. Crucially, it yields high-fidelity interaction traces grounded in actual execution, providing a reliable data source for policy optimization.

Verification and auditing module. This module continuously tracks the state and evaluates the goal conditions Φ after each step. When all conditions are satisfied, the environment terminates the episode and returns a verified success signal. Additionally, a runtime action auditor blocks unsafe or task-irrelevant operations (e.g., accessing non-existent functions or setting attributes beyond physical limits), converting them into deterministic negative feedback. This dual mechanism establishes a reliable feedback boundary for both offline data filtering and online policy optimization.

3.2.2 HomeMaker: Diverse Environment Instantiation

Smart home environments vary significantly across households in terms of layouts, device configurations, and environmental conditions. Training on a single, fixed environment inevitably leads to severe overfitting. To mitigate this, HomeMaker procedurally generates diverse and physically consistent initial states s_0.

For each home instance i, HomeMaker first samples real-world weather data to establish the external context. It then constructs a spatial layout $\mathcal{L}_i$ featuring nested structures (e.g., bedrooms with ensuite bathrooms). Based on room semantics, the engine samples devices from prior distributions to determine the device set $\mathcal{D}_i$ and their locations ψ_i. Finally, it integrates the external weather, layout structure, and device configurations to initialize the dynamic state variables $\mathcal{X}_0^{(i)}$. This hierarchical process ensures each episode begins with a realistic and logically sound state $s_0^{(i)}$. By expanding the context space through diverse layouts and device combinations, HomeMaker effectively enhances agent robustness across long-tail residential scenarios.

3.3 Verifiable Data Synthesis

As illustrated in Figure 1, leveraging HomeEnv for verifiable execution, we generate large-scale, high-quality training data through a two-stage pipeline: a static Blueprint Construction phase to construct the goal distribution $p(\Phi)$, and a dynamic MCTS-Flow search phase to efficiently explore interaction trajectories τ under the guidance of environmental verification signals.

3.3.1 Blueprint Construction

To synthesize diverse training data at scale, we propose the Blueprint Construction mechanism to orchestrate comprehensive interaction scenarios (i.e., "scripts"). Formally, we define a complete interaction blueprint as a tuple $\mathcal{B} = \langle \mathcal{U}, s_0^{(i)}, \mathcal{I}, \Phi_{\mathcal{B}} \rangle$. Here, $\mathcal{U}$ represents a multi-dimensional user profile (encompassing physiological characteristics, lifestyle habits, and environmental preferences, e.g., "an elderly person with poor vision in a duplex villa late at night"); $s_0^{(i)}$ is the initial physical state instantiated by HomeMaker in Section 3.2.2; $\mathcal{I}$ denotes the user slice-of-life segments generated based on the physical living scenario; and $\Phi_{\mathcal{B}}$ is the corresponding set of objective evaluation criteria.

Specifically, GPT-5 [2] acts as a scenario-scripter that, given the initialized physical environment $s_0^{(i)}$ and the user profile $\mathcal{U}$, generates continuous "slices of life" of $\mathcal{I}$ within this living space$s_0^{(i)}$. Subsequently, the continuous "slices of life" within $\mathcal{I}$ are segmented into multi-step atomic intents, each formulated as a candidate Boolean evaluation function ϕ_k. To guarantee absolute objectivity and validity, these generated conditions undergo a rigorous triple-filtering mechanism. First, an *Executability Check* is performed via the HomeEnv sandbox to intercept illegal conditions containing parsing errors or undefined device entities. Second, *Boundary Validity* is verified against the underlying Device Schema to ensure parameters remain within permissible hardware ranges. Finally, the logical consistency between the formulated conditions, the instruction intent, and the user profile is cross-verified using GPT-5. Conditions failing any check are directly discarded. Through this mechanism, ambiguous instructions are strictly reduced into a precise, machine-verifiable goal distribution $p(\Phi)$.

3.3.2 MCTS-Flow: Verifiable Trajectory Exploration

With the blueprint $\mathcal{B}$ established, the objective is to synthesize high-quality interaction trajectories τ that navigate the agent from $s_0^{(i)}$ to a final state satisfying the defined criteria $\Phi_{\mathcal{B}}$. Traditional linear dialogue generation approaches suffer from severe computational waste: a failure in the final turn of a multi-turn interaction necessitates discarding all preceding exploration. To overcome this problem, MCTS-Flow treats trajectory synthesis as a tree search problem, deeply integrating Monte Carlo Tree Search (MCTS) with HomeEnv environmental verification to efficiently generate diverse, high-quality trajectories.

MCTS-Flow constructs a heterogeneous dialogue search tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ within the constraints of the blueprint $\mathcal{B}$. Each node $v \in \mathcal{V}$ represents a reusable interaction prefix $s_v = \langle h_v, r_v \rangle$, where h_v is the multi-turn historical context comprising both conversational history and environmental feedback, and $r_v \in \{\text{User}, \text{Agent}\}$ indicates the active role. Three core modifications are introduced to tailor standard MCTS for the specific complexities of smart home scenarios:

Role-Aware Adaptive Branching. We observe that user utterances are strictly constrained by the blueprint intent, whereas agent decisions involve complex intent clarification and multi-step reasoning. Consequently, MCTS-Flow introduces an *asymmetric branching factor* $k(v)$ to manage these distinct roles. Specifically, we allocate a larger sampling width to agent nodes such that $k_{\text{agent}} > k_{\text{user}}$. This strategy concentrates the search budget on the agent's critical decision bifurcations, avoiding redundant sampling of the deterministic user inputs.

UCB Selection and Prefix Reuse. During the selection phase, the Upper Confidence Bound (UCB) algorithm is utilized to balance exploration and exploitation among available child nodes:

$$v^* = \arg \max_{u \in Ch(v),\, \rho(u)=1} \left[\frac{Q(u)}{N(u)} + c_{\text{ucb}} \sqrt{\frac{2 \ln N(v)}{N(u)}} \right] \tag{3}$$

where $Ch(v)$ is the set of child nodes, $N(u)$ is the visit count, $Q(u)$ is the accumulated verification reward, and $\rho(u) \in \{0, 1\}$ is a passable indicator. When a rollout ends, MCTS-Flow will keep all the simulated nodes and mark $\rho(u) = 0$ during backpropagation for all the nodes on the path if all of its child nodes are not passable. This mechanism transforms verified prefixes into reusable assets.

Environment-Verified Reward Backpropagation. Unlike most tree searches relying on model-based scoring, MCTS-Flow leverages the ground-truth feedback provided by the HomeEnv environ-

ment. When a rollout reaches a terminal node v_L, the fulfillment of the final state s_T is calculated:

$$R(v_L) = \mathbf{1}\left[\text{valid}(h_{v_L}) \wedge \prod_{k=1}^{K} \phi_k(s_T) = 1\right] \tag{4}$$

where $\text{valid}(h_{v_L})$ is a binary indicator ensuring the interaction is free of execution errors or command violations. This authentic physical reward is then backpropagated to update the $Q(v)$ and $N(v)$ of all ancestor nodes. Notably, this mechanism not only selects verified trajectories for SFT but also quantifies blueprint complexity through the verified success rate, effectively facilitating curriculum-driven reinforcement learning for increasingly challenging tasks.

3.4 Two-Stage Agent Training

3.4.1 Supervised Fine-Tuning

We extract successful paths from the MCTS-Flow search tree. For each terminal node with verified return $R(v_L) = 1$, the corresponding path is converted into a sequential interaction trajectory $\tau^* = (o_1, a_1, o_2, a_2, \ldots, o_T, a_T)$. All successful trajectories form the supervised dataset $\mathcal{D}_{\text{SFT}}$. The policy π_θ is optimized with the standard causal language modeling objective:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{\tau^* \sim \mathcal{D}_{\text{SFT}}}\left[\sum_{t=1}^{|\tau^*|} \log \pi_\theta(a_t \mid h_t, \Phi_{\mathcal{B}})\right]. \tag{5}$$

SFT injects critical behavioral priors into the agent, including intent clarification, device control semantics, and reasoning over environment observations. However, mere imitation leaves it vulnerable to compounding errors during long-horizon domestic interactions.

3.4.2 Step-wise RLVE: Aligning Policy with Physical Reality

In realistic smart homes, user intents and household states co-evolve, rendering offline RL on static conversational prefixes inadequate. To overcome the limitations of passive imitation, we introduce Step-wise RLVE, which optimizes the agent directly through online, multi-turn interactions.

Dynamic Trajectory Generation. During each training episode e initialized from a blueprint $\mathcal{B}$, a strong LLM serves as a dynamic user simulator π_{user}. The agent policy π_θ engages in a live interaction loop with both the simulator and the HomeEnv environment, generating a complete online trajectory $\tau^{(e)} = \left(u_1^{(e)}, a_1^{(e)}, r_1^{(e)}, \ldots, u_T^{(e)}, a_T^{(e)}, r_T^{(e)}\right)$, where the user utterance $u_t^{(e)} \sim \pi_{\text{user}}(\cdot \mid \mathcal{B}, h_{<t}^{(e)})$ is dynamically synthesized based on the blueprint constraints and the real-time interaction history. Correspondingly, the agent action $a_t^{(e)} \sim \pi_\theta(\cdot \mid h_{<t}^{(e)}, \Phi_{\mathcal{B}})$ triggers environmental updates, yielding the step-wise verification reward $r_t^{(e)}$ from Home. Crucially, this formulation ensures that the same blueprint $\mathcal{B}$ yields highly diverse user query sequences across different episodes. Consequently, the agent is forced to optimize over non-stationary input distributions, learning to adapt as the conversational context and household states co-evolve.

Composite Verifiable Reward. Because the environment updates dynamically after each step, HomeEnv evaluates execution validity turn-by-turn. To operationalize this, let $m_t = \frac{1}{K}\sum_{k=1}^{K} \phi_k(s_t)$ denote the completion rate of blueprint conditions at step t. We formulate a step-wise reward function that balances incremental task advancement, overall success, and execution safety:

$$r_t^{(e)}(s_t, a_t) = \lambda_{\text{prog}}(m_t - m_{t-1}) + \lambda_{\text{succ}}\mathbf{1}\left[t = T \wedge m_T = 1\right] - c_{\text{audit}}\mathbf{1}\left[\text{blocked}(a_t)\right] \tag{6}$$

Here, the progress term provides dense positive guidance whenever an action advances the household state closer to the overall goal. The success term strictly rewards full blueprint fulfillment at the episode's end. Meanwhile, the final indicator triggers a negative penalty c_{audit} for illegal actions intercepted by the HomeEnv firewall. This composite structure effectively mitigates the sparse reward problem in long-horizon interactions while tightly bounding the agent's trial-and-error behavior within safe operational limits.

Table 1: Performance (%) of various LLMs on SmartHome-Bench under a unified agent framework.

Model	QT1	QT2	QT3	QT4	QT5	Overall	Avg. Tool Calls	Avg. Tokens (K)
GPT-5.5	72.82	**90.19**	**94.27**	88.05	78.95	**85.80**	2.13	61.16
Claude-4.6-Sonnet	88.65	79.26	88.53	83.90	81.05	84.09	2.16	98.09
Qwen-3.5 Plus	89.45	73.89	88.53	**89.87**	85.26	84.15	2.17	84.03
GLM-5	91.03	74.07	90.68	87.27	**87.37**	84.40	2.32	86.06
DeepSeek-V4-Flash	**93.14**	78.15	84.59	87.79	84.21	85.16	2.50	26.98
MiniMax M2.7	73.88	55.56	61.65	81.30	81.05	68.06	2.10	77.90
Qwen3-4B	38.00	9.44	13.62	65.45	53.68	30.85	1.51	68.43
Role-play-4B	89.71	55.56	82.08	85.45	74.74	75.62	1.96	119.01
HomeFlow-SFT-4B	94.99	64.70	**89.50**	86.75	76.80	81.50	2.06	90.48
HomeFlow-RL-4B	**96.22**	**72.22**	88.17	**92.83**	**76.84**	**84.60**	2.06	89.07
Qwen3-8B	27.70	10.19	13.26	68.83	60.00	31.87	1.43	64.32
Role-play-8B	91.56	60.19	84.95	85.45	70.53	77.78	1.97	119.65
HomeFlow-SFT-8B	**94.99**	73.89	**90.68**	85.71	81.05	82.45	2.01	113.88
HomeFlow-RL-8B	94.72	**79.63**	**90.68**	**88.31**	**82.11**	**87.03**	2.07	89.11

4 Experiments

4.1 Experimental Setup

Datasets. To evaluate our method, we constructed SmartHome-Bench based on HomeEnv. Comprising a total of 1,678 instances, the dataset comprehensively covers diverse room layouts and rich device compositions. Specifically, the benchmark evaluates five core tasks: Atomic Control (QT1) involves executing explicit and straightforward user commands; Compositional Control (QT2) requires orchestrating multiple devices and performing condition-dependent reasoning based on the environment; Ambiguous Intent (QT3) focuses on inferring or clarifying specific operational goals from vague user descriptions; Context-Aware Multi-turn Interaction (QT4) handles coreference resolution, topic switching, and backtracking across consecutive dialogue turns; and Personalized Memory (QT5) assesses the model's ability to adhere to user habits and preferences. More details regarding the dataset are provided in Appendix E.

Notably, although training and evaluation share HomeEnv for state execution and verification with identical state semantics, test instances in SmartHome-Bench are procedurally sampled and constructed independently and are not sampled from training trajectories. We do not claim full environmental independence between training and evaluation partitions.

We note that HomeFlow relies on observable state transitions and verifiable execution feedback. Though its core loop may transfer to other stateful tool-using agent settings, our evaluation is limited to smart homes. Cross-domain generalization is not empirically validated here and remains a direction for future work. Further, all experiments are simulation-based; we do not claim direct sim-to-real transfer without real-device validation.

Baselines. We compare our method against two categories of baselines. The first category comprises state-of-the-art LLMs, such as GPT-5.5 [38], Claude-4.6 Sonnet [39], Qwen3.5 Plus [40], GLM-5 [41], DeepSeek-V4-Flash [42] and MiniMax M2.7 [43]. For the trained baselines, all models are based on the Qwen3-4B and Qwen3-8B. The primary comparison baseline employs the conventional role-play data synthesis method [44, 45], which fine-tunes the base models using purely LLM-simulated trajectories. To ensure a fair comparison, the role-play baseline is trained with the same number of samples as our method.

Evaluation Metrics. We evaluate the success rate for each category and the overall rate via objective state verification within HomeEnv. Additionally, we assess operational efficiency by tracking the average number of tool calls and output tokens per task.

Table 2: Analysis of asymmetric node expansion budgets. Base Traj. denotes the initial linear trajectories, Explored Traj. represents the additional trajectories yielded by node expansion in MCTS-Flow, and Total Traj. is their combined volume.

k_{user}	k_{agent}	Base Traj.	Explored Traj.	Total Traj.	Success Rate (%)
1	1	22,000	0	22,000	80.4
3	1	15,000	5,662	20,662	82.6
1	3	15,000	6,708	21,708	84.7
3	3	15,000	6,871	21,871	85.3

4.2 Main Results

Table 1 compares the performance of HomeFlow against various baselines on SmartHome-Bench. We observe that HomeFlow-RL, our full training pipeline, achieves the highest success rates across both Qwen3-4B and Qwen3-8B backbones. Specifically, for Qwen3-8B, HomeFlow-SFT achieves an overall success rate of 82.45%, outperforming the Role-play by 4.67 points using the same number of training samples. Further optimization via step-wise RLVE elevates the performance to 87.03%. A consistent trend is observed at the 4B scale, where HomeFlow-RL reaches a success rate of 84.60%, significantly surpassing its base and role-play counterparts.

The results highlight three key advantages of our method. First, our verifiable data synthesis provides a superior cold start by directly mapping user instructions to verifiable physical state transitions, allowing the model to master complex device-control logic more reliably than pure-text simulation. Second, Step-wise RLVE ensures the agent's policy remains robust during dynamic, multi-turn interactions, enabling it to learn from real-time feedback and recover from compounding errors that typically lead to task failure. Finally, our 8B HomeFlow-RL exceeds the performance of leading proprietary LLMs, proving that training within an interactive execution environment can overcome model scale constraints when handling complex, multi-device household orchestrations.

4.3 Analysis of Data Synthesis

4.3.1 Comparison of Synthesis Paradigms

We trace the performance improvements during the SFT stage by progressively ablating the components of our data synthesis pipeline, as illustrated in Figure 2. Our complete approach employs Blueprint grounding coupled with MCTS-Flow exploration. Removing the MCTS-Flow component (w/o MCTS-Flow) degrades the pipeline to Blueprint-grounded linear generation. Further removing the Blueprint constraints (w/o Blueprint) reduces the method to conventional pure-text role-play. This comparison allows us to assess both the efficiency of the data generation process and its impact on downstream model performance.

Higher Data Synthesis Efficiency. We first evaluate the generation efficiency of these paradigms as an empirical phenomenon. Pure-text role-play suffers from LLM hallucinations, yielding a dialogue generation success rate of 86.33%. Introducing physically grounded Blueprints forces the generation to respect initial states and tool logic, raising the success rate to 92.43%. However, this linear generation still discards the entire dialogue if a single step fails. By incorporating tree-based exploration and a rollout-saving mechanism, the full MCTS-Flow pipeline avoids this waste, successfully completing complex trajectories and achieving the highest generation success rate of 96.48%.

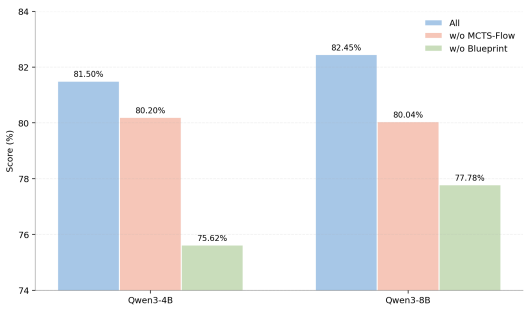

Figure 2: Performance comparison of different SFT data synthesis strategies. Our approach is evaluated against linear generation (w/o MCTS-Flow) and pure-text role-play (w/o Blueprint) to demonstrate the necessity of each component.

Verifiable Trajectories Provide a Better Cold Start. The quality of this synthesized data directly translates to downstream model perfor-

mance. Upgrading from pure-text role-play to Blueprint-grounded linear generation raises the task success rate on Qwen3-4B and Qwen3-8B by 4.58 and 2.26 percentage points, respectively. This demonstrates that trajectories generated via pure-text role-play, while textually plausible, often violate hidden device preconditions. In contrast, Blueprint-grounded trajectories are executed and verified in the HomeEnv sandbox, enabling the model to learn executable state transitions rather than mimicking surface-level conversational patterns.

MCTS Enhances Trajectory Diversity. Building upon the Blueprint-grounded generation, incorporating MCTS-Flow exploration further improves SFT performance, reaching a task success rate of 82.45% on Qwen3-8B and 81.50% on Qwen3-4B. By searching through alternative user intents and tool invocation sequences, MCTS-Flow significantly increases the diversity and coverage of successful trajectories. This mechanism proves particularly effective for complex, multi-step scenarios, establishing a robust behavioral prior before the multi-turn RL optimization.

4.3.2 Impact of Asymmetric Branching Factors in MCTS-Flow

Smart home interactions exhibit an inherent role-based asymmetry: user utterances are largely deterministic and constrained by blueprints, whereas agent decisions involve complex multi-step reasoning. To analyze how this affects search efficiency, we evaluate different combinations of branching factors k_{user} and k_{agent} while maintaining a comparable total trajectory scale, as shown in Table 2. We observe that allocating a larger budget to agent nodes where k_{agent} equals 3 yields a significantly higher success rate of 84.7% compared to the 82.6% achieved by user-side expansion. This confirms that the agent's action space is the primary source of task complexity. While joint expansion where both k_{user} and k_{agent} equal 3 achieves the highest success rate of 85.3%, the marginal improvement over agent-only expansion is small. These results empirically justify our asymmetric branching strategy where $k_{\text{agent}} > k_{\text{user}}$, which effectively concentrates the search budget on critical agent decision bifurcations while avoiding redundant sampling of deterministic user inputs.

4.4 Analysis of Step-wise RLVE

Step-wise RLVE serves as a critical phase to align the agent's policy with the non-stationary dynamics of physical households. While MCTS-enhanced SFT provides a high-quality behavioral prior, Step-wise RLVE enables the agent to transition from passive imitation to environment-driven exploration. By interacting with a dynamic user simulator and the HomeEnv sandbox, the agent learns to refine its decision-making through trial-and-error, grounded by composite verifiable rewards. As shown in Table 3, the HomeFlow-RL pipeline achieves a task success rate of 87.03%, representing a substantial 4.58 percentage point improvement over the HomeFlow-SFT baseline. This gain demonstrates that episodic optimization allows the agent to manage long-horizon dependencies and recover from intermediate execution failures, effectively overcoming the compounding errors that typically plague static imitation learning.

To further investigate reward granularity, we compare HomeFlow-RL with HomeFlow-RL (ST). In the latter, the agent perceives the entire context but updates are restricted to the final action based on the ultimate outcome. While this improves success rates to 86.46%, the scarcity of feedback makes the policy vulnerable to early collapse. As shown in Figure 3, the policy entropy of HomeFlow-RL (ST) drops sharply,

Table 3: Comparison of performance (%) across different training paradigms. **ST** denotes single-turn RL where updates only occur at the final step, while **Full** refers to our complete Step-wise RLVE with multi-turn episodic rewards.

Model	Success Rate	Avg. Tools	Avg. Toks
HomeFlow-SFT	82.45	2.01	113.8
HomeFlow-RL (ST)	86.46	2.25	79.0
HomeFlow-RL (Full)	**87.03**	2.07	89.1

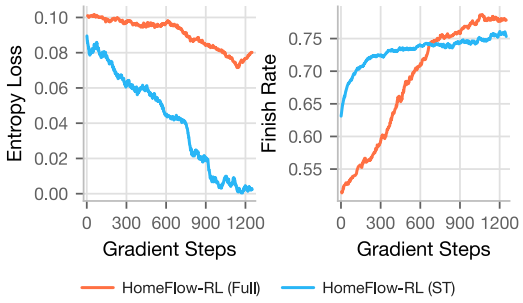

Figure 3: Training dynamics of HomeFlow-RL variants. Sparse feedback causes early policy collapse in the ST variant, whereas dense step-wise rewards in the complete model sustain exploration.

whereas HomeFlow-RL maintains exploration
through dense, real-time guidance. Notably, HomeFlow-RL reduces average tool calls by 8% compared to this variant, suggesting that step-wise credit assignment fosters more concise and purposeful planning. These results justify the necessity of dense rewards in verifiable environments. Qualitative examples of post-training model execution are provided in Appendix F.

5 Conclusion

HomeFlow establishes a verifiable training pipeline where simulation specifies tasks, verifies trajectories, and provides rule-based RL rewards. Combining Blueprint-grounded synthesis, MCTS expansion, and step-wise RLVE, HomeFlow boosts the Qwen-4B and Qwen-8B models to 84.20% and 87.03% task success rates on SmartHome-Bench, respectively. Our results yield three key insights: (1) verifiable environments provide a unified foundation for both SFT data synthesis and online policy optimization; (2) step-wise interaction sustains exploration and significantly improves long-horizon control; and (3) MCTS-driven search enhances trajectory diversity while transforming verified prefixes into reusable assets. Ultimately, these findings suggest a practical recipe for AIoT agents with checkable goals.

Limitations and future work. Currently, HomeFlow relies on deterministic simulation, necessitating sim-to-real adaptation to address real-world uncertainties like latency and sensor drift. Broader deployment also requires adapting to diverse vendor-specific protocols to expand device coverage, alongside mitigating the inherent computational overhead introduced by MCTS and online RL.

Promising future directions include incorporating continual learning for personalized household routines and extending the verifiable framework to broader AIoT ecosystems.

Statement. We emphasize that our work targets *agent policy learning* (intent understanding, state-aware reasoning, multi-turn clarification and high-level device invocation), and abstracts away low-level hardware adapter engineering. Systematic real-device experiments are absent in this paper; hence we make no empirical claims regarding direct sim-to-real transfer to physical smart-home setups. Real-world integration still faces separate practical challenges including heterogeneous vendor protocols, state normalization, network latency, sensor drift and asynchronous device failures, which demand dedicated engineering efforts. Our focus differs from embodied robotics [46, 47], which additionally requires visual perception and continuous low-level physical execution.

Moreover, training and evaluation rely on the same HomeEnv simulator with unified state semantics; test instances are independently constructed but we do not assume full environmental independence.

6 Acknowledgements

This work was supported by the AI Research Center of Midea Group. The authors declare no competing interests beyond the affiliations and funding disclosed in this paper.

References

[1] Jiaze Li, Jingyang Chen, Yuxun Qu, Shijie Xu, Zhenru Lin, Junyou Zhu, Boshen Xu, Wenhui Tan, Pei Fu, Jianzhong Ju, et al. Xiaomi mimo-vl-miloco technical report. *arXiv preprint arXiv:2512.17436*, 2025.

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

[3] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[4] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[5] Badis Hammi, Sherali Zeadally, Rida Khatoun, and Jamel Nebhen. Survey on smart homes: Vulnerabilities, risks, and countermeasures. *Computers & Security*, 117:102677, 2022.

[6] Noor Ul Huda, Ijaz Ahmed, Muhammad Adnan, Mansoor Ali, and Faisal Naeem. Experts and intelligent systems for smart homes' transformation to sustainable smart cities: A comprehensive review. *Expert Systems with Applications*, 238:122380, 2024.

[7] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. Character-llm: A trainable agent for role-playing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13153–13187, 2023.

[8] Xintao Wang, Heng Wang, Yifei Zhang, Xinfeng Yuan, Rui Xu, Jen-tse Huang, Siyu Yuan, Haoran Guo, Jiangjie Chen, Shuchang Zhou, et al. Coser: Coordinating llm-based persona simulation of established roles. In *Proceedings of the Forty-Second International Conference on Machine Learning*, 2025.

[9] Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024.

[10] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

[11] Daya Guo et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633–638, 2025.

[12] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[13] Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025.

[14] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[15] Dmitriy Rivkin, Francois Hogan, Amal Feriani, Abhisek Konar, Adam Sigal, Steve Liu, and Greg Dudek. Sage: Smart home agent with grounded execution. *arXiv preprint arXiv:2311.00772*, 2023.

[16] Chaerin Yu, Chihun Choi, Sunjae Lee, Hyosu Kim, Steven Y. Ko, Young-Bae Ko, and Sangeun Oh. Leveraging llms for efficient and personalized smart home automation. *arXiv preprint arXiv:2601.04680*, 2026.

[17] Rune Birkmose, Nathan Mørkeberg Reece, Esben Hofstedt Norvin, Johannes Bjerva, and Mike Zhang. On-device llms for home assistant: Dual role in intent detection and response generation. *arXiv preprint arXiv:2502.12923*, 2025.

[18] Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. Sasha: Creative goal-oriented reasoning in smart homes with large language models. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(1):1–38, March 2024.

[19] Ziqi Yin, Mingxin Zhang, and Daisuke Kawahara. Harmony: A human-aware, responsive, modular assistant with a locally deployed large language model. In *Companion of the 2025 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 126–130. Association for Computing Machinery, 2025.

[20] Xinyu Huang, Leming Shen, Zijing Ma, and Yuanqing Zheng. Towards privacy-preserving and personalized smart homes via tailored small language models. *arXiv preprint arXiv:2507.08878*, 2025.

[21] Zhonghao Zhan, Krinos Li, Yefan Zhang, and Hamed Haddadi. Hearthnet: Edge multi-agent orchestration for smart homes. *arXiv preprint arXiv:2604.09618*, 2026.

[22] Silin Li, Yuhang Guo, Jiashu Yao, Zeming Liu, and Haifeng Wang. Homebench: Evaluating llms in smart homes with valid and invalid instructions across single and multiple devices. *arXiv preprint arXiv:2505.19628*, 2025.

[23] Gyuhyeon Seo, Jungwoo Yang, Junseong Pyo, Nalim Kim, Jonggeun Lee, and Yohan Jo. Simuhome: A temporal- and environment-aware benchmark for smart home llm agents. In *International Conference on Learning Representations*, 2026.

[24] Kuan Li, Shuo Zhang, Huacan Wang, Fangzhou Yu, Zecheng Sheng, Yi Gu, Weipeng Ming, Lei Xue, Chen Liu, Sen Hu, et al. Smh-bench: Benchmarking llm agents for environment-grounded reasoning and action in smart homes. *arXiv preprint arXiv:2606.01912*, 2026.

[25] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[26] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[27] Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andres Codas, Yadong Lu, Wei-ge Chen, Olga Vrousgos, Corby Rosset, et al. Agentinstruct: Toward generative teaching with agentic flows. *arXiv preprint arXiv:2407.03502*, 2024.

[28] Bo Ni, Leyao Wang, Yu Wang, Branislav Kveton, Franck Dernoncourt, Yu Xia, Hongjie Chen, Reuben Leura, Samyadeep Basu, Subhojyoti Mukherjee, et al. A survey on llm-based conversational user simulation. *arXiv preprint arXiv:2604.24977*, 2026.

[29] Xun Jiang, Yufan Gu, Disen Hu, Yuqing Hou, Yazhou Yao, Fumin Shen, Heng Tao Shen, and Xing Xu. Multimodal learning on low-quality data with conformal predictive self-calibration. *arXiv preprint arXiv:2605.03820*, 2026.

[30] Yifei Gao, Junhong Ye, Jiaqi Wang, and Jitao Sang. Websynthesis: World-model-guided mcts for efficient webui-trajectory synthesis. *arXiv preprint arXiv:2507.04370*, 2025.

[31] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.

[32] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

[33] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[34] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[35] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[36] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[37] Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

[38] OpenAI. Gpt-5.5 system card. `https://deploymentsafety.openai.com/gpt-5-5`, 2026. Accessed: 2026-05-07.

[39] Anthropic. Claude sonnet 4.6 system card. `https://www.anthropic.com/claude-sonnet-4-6-system-card`, 2026. Accessed: 2026-05-07.

[40] Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, 2026. Accessed: 2026-05-07.

[41] Aohan Zeng et al. Glm-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

[42] DeepSeek-AI. Deepseek v4 technical documentation. `https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf`, 2026. Accessed: 2026-05-07.

[43] MiniMax. Minimax m2.7: Early echoes of self-evolution. `https://www.minimax.io/news/minimax-m27-en`, 2026. Accessed: 2026-05-07.

[44] Nuo Chen, Yan Wang, Yang Deng, and Jia Li. The oscars of ai theater: A survey on role-playing with language models. *arXiv preprint arXiv:2407.11484*, 2024.

[45] Yelaman Abdullin, Diego Molla-Aliod, Bahadorreza Ofoghi, John Yearwood, and Qingyang Li. Synthetic dialogue dataset generation using llm agents. *arXiv preprint arXiv:2401.17461*, 2024.

[46] Zuxing Lu, Hongjia Zhai, Guanzhi Wang, Huajian Zeng, Jiaqi Yang, Jingyu Liu, Lei Cheng, Yuantai Zhang, Yuheng Qiu, Zezhou Cheng, et al. World-action models for robot learning and control: A survey. *arXiv preprint arXiv:2609.16074*, 2026.

[47] Jiawei Fan, Sifeng Wang, Yuqing Hou, and Anbang Yao. Kinematic meanflow: One-step action generation policy for robotic foundation models. *arXiv preprint arXiv:2610.00864*, 2026.

[48] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[49] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *2018 IEEE/cvf conference on computer vision and pattern recognition*, pages 8494–8502. IEEE, 2018.

A HomeEnv Simulation Engine

We introduce HOMEENV, a smart home simulation engine that supports configurable device specifications, programmatic control APIs, and verifiable task execution in a Gym-style environment.

A.1 YAML-based Smart Device Specification

Devices are specified using a YAML-based configuration format. The following example defines a smart light device:

YAML Specification of a Smart Light Device

```yaml
- name: light
  userdata:
    category: light
    brand: example-brand
    spid: "100001"
  attributes:
  - name: state
    type: str
    options: ["on", "off"]
  - name: brightness
    type: int
    range: [1, 100]
    unit: percentage
  - name: hs_color
    type: tuple
    items:
      - type: float
        range: [0.0, 360]
      - type: float
        range: [0, 100]
  services:
  - name: turn_on
    code: self.state = "on"
  - name: turn_off
    code: self.state = "off"
  - name: set_brightness
    arguments:
      - name: brightness
        type: int
        range: [1, 100]
    code: self.brightness = brightness
  - name: set_hs_color
    arguments:
      - name: hs_color
        type: tuple
        items:
          - type: float
            range: [0.0, 360]
          - type: float
            range: [0, 100]
    code: self.hs_color = hs_color
```

A.2 Programmatic Device Interaction API

Configured devices can be instantiated and controlled through a unified high-level API:

Programmatic Control of a Configured Device

```
# load configured light device
with open("light.yaml", "r") as f:
    entities = Entity.load(f)
    light_spec = entities[0]

# initialize device with randomized attributes
light = light_spec.rand()

# inspect and control device
print(light.state, light.brightness)

light.turn_on()
light.set_brightness(80)

print(light.state, light.brightness)
```

A.3 Gym-style Home Simulation Environment

We further build a Gym-style smart home environment that automatically generates house layouts and embedded devices, including rooms, floors, and device instances.

Interaction with the HomeEnv Simulator

```
env = gym.make("HomeEnv-v0", home=HomeSampler.sample_test())

observation, _, terminated, truncated, info = env.step({
    "name": "pyexec",
    "code": """
device("1001").set_brightness(80)
print(f"brightness of light is {device('1001').brightness}")
"""
})
```

A.4 Verifiable Task Evaluation

To support automatic evaluation, we define verifiable tasks as logical conditions derived from the environment state.

Verification of Environment Tasks

```
# ground-truth conditions
ground_truth_list = [
    "device('1001').state == 'on'",
    "device('1001').brightness > 60",
]

env = gym.make("HomeEnv-v0", home=HomeSampler.sample_test())

# register tasks
for gt in ground_truth_list:
    env.engine.task.add_task(gt)

# interact with environment
```

```
env.step({"name": "pyexec", "code": "..."})

# verify task completion
for task in env.engine.task.tasks:
    print(f"task id={task.id} status={task.finished} condition={task.
        condition}")
```

B Implementation Details of Blueprint

We propose a two-stage data synthesis framework for generating large-scale, controllable, and diverse task-oriented dialogue data. The overall pipeline consists of:

Blueprint Generation. constructing structured task blueprints conditioned on user profiles, household environments, and constraint validation mechanisms.

Dialogue Generation. synthesizing grounded multi-turn interactions based on generated blueprints, user profiles, and simulated agents.

B.1 Blueprint Generation

B.1.1 Meta Information Construction

Blueprint generation is conditioned on three categories of meta-information: household environment, user profile, and tool/action space,detail information in Table 4.

Table 4: Meta-information used in blueprint construction.

Category	Attributes	Description
Household Environment	temperature, humidity, brightness	Physical state variables in a smart-home environment.
	device APIs (e.g., light.turn_on, ac.set_temp)	Controllable smart home device interfaces.
	tool APIs (e.g., calculator, weather)	External function invocation interfaces.
	environment entities	Spatial context such as rooms and indoor/outdoor zones.
User Profile	age, gender	Basic demographic attributes.
	physiological attributes	Height, weight, and physical condition.
	health status	Chronic diseases and sleep-related conditions.
	behavior habits	Daily routines and lifestyle patterns.
	preferences	Environmental preferences (temperature, lighting, etc.).

To ensure realism and diversity, we sample attributes from predefined distributions and enforce structured consistency constraints (e.g., age–health correlation, lifestyle–preference alignment).

We adopt a three-stage generation paradigm:

(1) Base Blueprint Generator. Given user profiles and environment states, the generator produces a candidate set of task blueprints representing possible user goals.The overall Blueprint as shown in Section B.1.2.

(2) Blueprint Filter. The Blueprint Filter removes invalid or low-quality candidates before task execution, as summarized in Table 5.

(3) Query Type. We introduce a structured query perturbation mechanism to control difficulty and diversity,as summarized in Table 6.

Formally, queries are sampled from the joint space of these factors to enable controlled complexity and diversity.

Table 5: Filtering criteria used by the Blueprint Filter.

Criterion	Description
Logical consistency	Removes contradictory or incoherent candidates.
Executability	Ensures compatibility with environment and tool APIs.
Diversity	Reduces redundant candidates and improves task coverage.

Table 6: Query perturbation control space.

Dimension	Options	Description
Intent Clarity	clear / ambiguous / missing-slot	Controls explicitness of user intent.
Noise Level	clean / noisy	Controls linguistic perturbation and ambiguity.
Device Scope	single-device / multi-device / batch-device	Controls action granularity.
Execution Dependency	independent / dependent	Controls temporal dependency across steps.

B.1.2 Fully Example

Blueprint

Mission: Study reading and pre-exercise environment adjustment
Description:
Scene 1: In the afternoon, Li Mei prepares to practice handwriting in the study. Indoor sensors show the temperature is about 13 °C; she feels a bit cold. The ceiling fan light is on but only at 11% brightness with a slightly warm color temperature, which makes strokes hard to see. The smart speaker is paused but not muted, and she worries notification sounds may interrupt. She hopes to warm the room quickly, then make the lighting brighter and cooler white, and finally keep the environment quiet for focused writing.
Scene 2: In the spare bedroom doing tai chi warm-up, the hygrometer shows the room is hot (about 31.8 °C) with relatively high humidity (about 62%). He wants to lower the humidity and bring perceived temperature closer to around 20 °C for light exercise.

Tasks:

1. **Intent:** The room feels cold—she wants it warmer; before writing, the light is too dim and too cool-toned—she wants it brighter.
   Query Type: missing_slot+quiet+dependent+multiple_devices

   Query: If the AC is off, turn on heating and set the temperature to around 24 °C; also the study light is too dim—raise both brightness and color temperature.

   Condition:

   (device("FQVZSo_p").state == "on" **and**
   device("FQVZSo_p").ac_mode == "heat" **and**
   23.5 <= device("FQVZSo_p").target_temperature <= 24.5) **and**
   (85 <= device("sTPi5CsC").light.brightness <= 100) **and**
   (5000 <= device("sTPi5CsC").light.color_temperature <= 6500)

   Devices: FQVZSo_p, sTPi5CsC

2. **Intent:** Worried about being disturbed by notification sounds—wants a quieter environment.
   Query Type: clear+quiet+dependent+single_device

   Query: If the smart speaker is not muted, mute it.

Condition:

(device("1TirJUuR").player.volume_muted_enabled == True)

Devices: 1TirJUuR

3. **Intent:** The spare bedroom humidity is high—wants it lower; the spare bedroom temperature is high—wants it cooler.
   Query Type: clear+quiet+independent+sigle_devices

   Query: If the spare bedroom AC is still on, switch it to dehumidify mode and lower the set temperature to around 21 °C.

   Condition:

   (device("e7J6mKqC").ac_mode == "dry") **and**
   (20 <= device("e7J6mKqC").target_temperature <= 21)

   Devices: e7J6mKqC

B.2 Dialogue Generation

B.2.1 Simulation Framework

We construct a closed-loop interaction system consisting of, this enables grounded, executable, and stateful dialogue trajectories. Shown in Table 7.

Table 7: Components in the interactive evaluation environment.

Component	Role
Blueprint User (GPT-5)	Generates task-grounded user queries.
Assistant (GPT-5)	Produces responses and tool actions.
Sandbox	Executes actions and returns environment feedback.
Environment State	Dynamically updates after each interaction step.

B.2.2 Output Data Structure

Each sample is represented in a unified structured format supporting both supervised and reinforcement learning. Shown in Table 8.

Table 8: Unified data sample structure.

Component	Fields	Description
Meta Information	user profile, environment state, query type	Context conditioning signals.
Dialogue Messages	user input, assistant response, environment observation	Multi-turn interaction trajectory.
Task Status	completion flag	Execution-level supervision signal.

B.3 Core Prompt Design

B.3.1 Blueprint Prompt

Blueprint Generation Prompt

Task Definition. You are an expert system for generating smart home interaction scenarios grounded in real household environments and user profiles.
Objective. Synthesize realistic life scenes and corresponding atomic executable tasks for smart home systems, ensuring physical feasibility, behavioral realism, and strict device-level grounding.
Generation Procedure.
1. Life Scene Construction

- Generate a fixed number of daily-life scenes.
- Each scene must describe continuous human behavior in a single physical location.
- Scenes must reflect user profile characteristics and environment states.
- Device states must be grounded in observed environment information.
- Do not generate command-style or task-list-style descriptions.

2. Atomic Task Decomposition

- Decompose each scene into atomic tasks.
- Each task must originate from exactly one scene.
- Each task must represent a single user intent only.
- No new devices or attributes may be introduced.

Hard Constraints.
1. Device Constraint

- Only devices explicitly defined in the environment are allowed.
- Device identifiers must match exactly.

2. Attribute Constraint

- Only attributes listed in device service definitions may be used.
- No inferred or hallucinated attributes are allowed.

3. Atomicity Constraint

- Each task must contain exactly one user intent.
- Multi-intent or composite tasks are not allowed.

4. Physical Consistency Constraint

- All actions must be physically executable in the given environment.
- Device state transitions must be realistic and valid.

5. Device State Dependency Constraint

- Functional attributes (e.g., temperature, brightness) are only valid when device is active.
- If device is inactive, activation must be included before modifying dependent attributes.

Output Format. Strict JSON only, no explanation or additional text.

B.3.2 User Profile Prompt

User Profile Generation Prompt

Task Definition. You are an expert system for generating smart home user profiles.
Objective. Synthesize realistic, diverse, and internally consistent user profiles for smart home automation systems.
Design Principles.
1. Smart Home Grounding All attributes must map to controllable system behaviors:

- Physiology $\rightarrow$ environment adaptation
- Health $\rightarrow$ HVAC / air quality control

- Habits → automation scheduling
- Preferences → scene control policies

2. Completeness Constraints

- All modules must be present: basic, physiology, health, habit, preference
- No missing or null fields
- Full internal consistency required

3. Realism Constraints

- Cover full demographic spectrum
- Include realistic health conditions
- Ensure physically plausible values

Output Format. Strict JSON only, no explanation.

B.3.3 User Query Prompt

User Query Generation Prompt

Task Definition. You are an expert system for generating natural user requests in a smart home environment.
Objective. Convert structured task states into realistic, minimal, and executable user queries that can be directly used for device control via tool invocation.
Design Principles.
1. Task-Centric Grounding All outputs must strictly align with the current task list:

- Focus only on **unfinished tasks**
- Ignore historical dialogue unless task-relevant
- Ensure every query directly contributes to task completion

2. Device-State Awareness Generation must consider real-time home state:

- Room-level and device-level status constraints
- Avoid redundant or already-satisfied operations
- Adjust actions based on current device conditions

3. Natural User Expression Constraint Ensure outputs reflect realistic user behavior:

- Avoid enumerating multiple parameters or rooms in a rigid format
- Avoid overly structured or machine-like command patterns
- Use short, conversational, and implicit instructions

4. Minimalism Principle

- One unified query per generation step
- No task IDs, no explanations, no reasoning traces
- Merge actions only when naturally expressed

5. Tool-Execution Requirement

- Output must be passed through `post_process` tool
- Must include:
  - `query`: a single executable instruction string
  - `reason`: structured justification of generation logic
- `query` must never be empty or null

Output Format. Strict tool-call format only. No free-text response.

B.3.4 Agent Prompt

Smart Home Agent Prompt

Task Definition. You are an intelligent smart home agent with strong reasoning and device control capabilities. You operate inside a user's home environment.
Objective. Accurately understand user intent (query/control), interact with smart home devices via tools, and return correct, executable results.
System Context.

- **Home Devices:** Structured multi-room device list with metadata (id, did, category, brand, tags).
- **User Memory:** Historical preferences, behaviors, and contextual clips (if available).
- **User Scene Context:** Environmental and situational signals for personalization.

Core Workflow (Strict).

1. **Intent Recognition:** Identify query type (status / control / ambiguous / chit-chat).
2. **Information Retrieval:** Query device manuals and states before executing actions.
3. **Disambiguation:** Apply clarification rules when intent or target is not deterministic.
4. **Tool Execution:** Execute device operations only after validating capability and parameters.
5. **Result Verification:** Check tool outputs; iterate if unmet.
6. **User Response:** Return concise ($\leq$ 50 Chinese characters), spoken-style answer only.

Clarification Principle.

- No clarification if device + function are uniquely identifiable.
- Ask user only when multiple valid interpretations exist.
- Avoid interpreting "all devices" unless explicitly stated (e.g., "all lights").

Execution Constraint.

- Always fetch device capability before control.
- Always ensure parameter validity before execution.
- Never output tool reasoning or internal steps to user.

C MCTS-Flow for Dialogue Generation

In this section, we present an example of using MCTS-Flow to generate and select dialogue data.

As illustrated in Figure 4, the blueprint specifies the user goal, task objectives, and success metrics, collectively referred to as the User Goal and Blueprint Validator. A dialogue session is considered successful only if all conditions of the Blueprint Validator are satisfied upon termination.

In this example, MCTS-Flow performs 5 dialogue rollouts from a fixed blueprint. The initial user query is generated by the root node, followed by the agent's responses. For each user query, the agent generates at most 3 distinct candidate responses.

Some dialogues fail to meet the conditions of the Blueprint Validator upon termination and are thus deemed unsuccessful, as indicated in red. Two of them satisfy the conditions, as indicated in green. These successful instances can be collected to train the smart home agent. Notably, the successful ones share part of their dialogue trajectories with some of the failed ones.

As for this example, MCTS-Flow will collect 2 successful dialogue trajectories, as shown in the following 2 figures. Note that these 2 dialogues meet the user's requirements in different ways, enhancing data diversity.

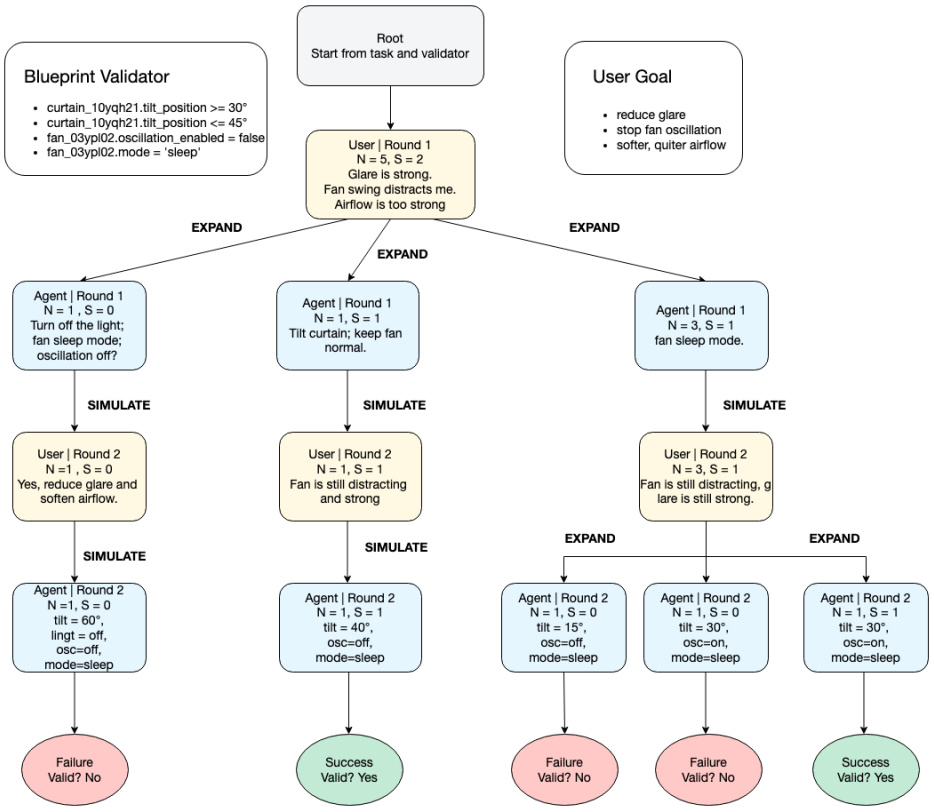

Figure 4: A demonstration of MCTS-Flow for dialogue generation.

Success Dialogue 1

User Query: "The glare is strong, the fan swing distracts me. And the airflow is too strong."

Smart Home Agent:

- `[Query]` → Query the device status of the room.
- `[Device Control]` → Tilt the curtain to 40 degree.

User Query: "The fan is still distracting me. And the airflow is too strong."

Smart Home Agent:

- `[Query]` → Query the device specification of the room.
- `[Device Control]` → Set the oscillation of fan to 'off'. Set the mode of the fan to 'sleep'

Success Dialogue 2

User Query: "The glare is strong. The fan swing distracts me. And the airflow is too strong."

Smart Home Agent:

- `[Query]` → Query the device status of the room.
- `[Device Control]` → Set the mode of the fan to 'sleep'.

User Query: "The fan is still distracting me. And the glare is still strong."

Smart Home Agent:

- `[Query]` → Query the device specification of the room.
- `[Device Control]` → Set the oscillation of fan to 'off'. Tilt the curtain to 40 degree.

D Details of the Two-Stage Training

D.1 Training Data Volume and Composition

The supervised fine-tuning (SFT) dataset consists of three distinct subsets, all derived from a shared foundational pool of 20,000 user profiles. Specifically, we first generated these 20,000 profiles, each encapsulating attributes such as age, occupation, and behavioral habits. Subsequently, HomeMaker was employed to procedurally generate corresponding household configurations for each user. Building upon this foundation, dialogue trajectories were synthesized corresponding to the three data generation schemes evaluated in our ablation study:

Pure-Text Role-Play (w/o Blueprint): Based on the 20,000 user profiles with basic household information, we adopted a conventional pure-text role-playing scheme without strict physical blueprint constraints. This initial process produced 20,000 dialogues. After applying an LLM-based filter to ensure quality, 16,700 dialogues were retained, from which 15,000 samples were randomly selected to form the first subset.

Blueprint-Grounded Linear Generation (w/o MCTS-Flow): To incorporate physical grounding, HomeMaker was utilized to generate explicit household blueprints for the same 20,000 users. Following the linear generation scheme (without search-based exploration), we obtained 17,500 physically grounded dialogue trajectories, from which 15,000 samples were randomly selected for this subset.

Blueprint-Grounded MCTS-Flow (Complete Pipeline): Applying our complete data synthesis pipeline, we coupled the blueprint configurations with MCTS-Flow search. Guided by verification signals, this dynamic exploration phase yielded a highly diverse set of 30,000 verifiable dialogues. From these, 15,000 samples were randomly selected to form the final, highest-quality SFT subset.

For the subsequent reinforcement learning (RL) stage, the training environments are initialized from a distinct set of 10,000 base configurations, each encompassing a user profile, household information, and a physical blueprint. We constructed two corresponding setups to support our online policy optimization:

HomeFlow-RL (Complete Step-wise RLVE): The 10,000 base configurations were utilized directly to initialize the HomeEnv simulator, enabling the agent to engage in complete, multi-turn episodic interactions guided by step-wise dense rewards.

HomeFlow-RL (ST): To construct the single-turn ablation baseline, we first generated complete multi-turn trajectories from the same 10,000 base configurations. We then stripped away all intermediate tool-use and device control turns, retaining only the final single-turn interaction. This limits the agent to perceiving the full context but only receiving sparse feedback based on the ultimate outcome.

D.2 Training Setups

Reward Shaping: We design three rule-based reward components to constrain model behavior.

CodeCheck penalizes the use of restricted Python operations, including introspection and dynamic execution-related keywords such as *getattr*, *setattr*, *eval*, *exec*, and *compile*, among others.

CodeLength enforces a maximum code length of 1000 tokens, assigning a reward of -1 if exceeded and 0 otherwise.

TurnCheck limits the interaction horizon to 3 turns, similarly assigning a penalty of -1 when violated and 0 otherwise.

The detailed hyperparameters for both HomeFlow-RL setups are summarized in Table 9.

Table 9: RLVE training hyperparameters. Bold values mark the key divergence: HomeFlow-RL (ST) employs entropy regularization ($\beta_{\text{entropy}} = 10^{-4}$) to combat collapse, while HomeFlow-RL (Full) removes entropy loss and instead enforces a KL constraint ($\beta_{\text{KL}} = 3 \times 10^{-3}$) against the SFT reference, relying on the dynamic User simulator for natural exploration.

Parameters	**HomeFlow-RL (ST)**	**HomeFlow-RL (Full)**
GPU Type	Nvidia H20 96GB	
Memory	1TB	
Base Model	Qwen3-8B-SFT	
TP	4	
PP	1	
CP	1	
SP	on	
Rollout group size	8	
Rollout temperature	1.0	
LR	1e-6	
LR warmup fraction	0.01	
Minimize LR	5e-7	
TaskFinish weight	1.0	
CodeCheck weight	0.2	
CodeLength weight	0.2	
TurnCheck weight	0.2	
Num of GPUs	**32**	**96**
Training time cost	**38h+**	**95h+**
KL Loss coefficient	**0.0**	**3e-3**
Entropy Loss coefficient	**1e-4**	**0.0**
User LLM	–	**DeepSeek-V3.2**

E SmartHome-Bench

We constructed SmartHome-Bench, a comprehensive benchmark featuring 1,678 test samples systematically categorized into 5 major groups and 18 subcategories (see Table 10). To rigorously test the agent's scalability and spatial reasoning, the environmental difficulty is highly diversified, encompassing a broad distribution of household layouts ranging from simple setups with fewer than 10 devices to highly complex homes with over 100 devices.

Table 10: Template families and generation rules.

Family	Templates	Grounding rule	Surface instruction rule
`TC1: Atomic Control`	`Clear Commands`; `Colloquial Requests`; `Noisy Utterances`	Select one controllable device and one supported service or attribute change. The target must be unique and directly executable.	`clear` is explicit; `colloquial` uses everyday phrasing; `noisy` may include filler words, repetition, ASR-like artifacts, or immediate self-correction.

Continued on next page

Table 10: Template families and generation rules (continued).

Family	Templates	Grounding rule	Surface instruction rule
TC2: Compositional Control	Explicit Multi-device; Batch Operations; State-dependent; Room-dependent; Top-N Selection	Select multiple operations, a device set, a state-filtered subset, a room-conditioned subset, or a ranked top-k target. All selected devices must support the requested operation.	Express the dependency in one household request, e.g., "all lights", "lights above 80%", "rooms where the AC is on", or "the two fastest fans".
TC3: Ambiguous Intent	State-level Ambiguity; Scene-level Ambiguity; Slot-missing	Choose a plausible target but make the initial utterance under-specified, subjective, or scene-like. Missing slots may include room, device, mode, value, duration, or time.	The initial query should require clarification or context-based inference. The hidden target is preserved through `intended_query` or the reference clarification trajectory.
TC4: Context-Aware Multi-turn Interaction	Context Continuation; Topic Shifts; Coreference Resolution; User Corrections	The current target depends on previous turns. The generator may inherit a device, override a previous operation, resolve a pronoun, or switch topics while preserving context.	Use natural dialogue forms such as "also do that", "no, make it 26", "turn that one off", or a topic-shifted follow-up.
TC5: Personalized Memory	Short-term Memory;	Use recent interaction facts or stored user preferences to fill omitted action details. The task should not be solvable in the intended way without the provided memory.	Allow compact preference-triggering instructions such as "sleep time", "as usual", or "set it how I like it".

F Case Study of the Impact of RL Training on Model Ability

In this section, we present several case studies on RL training. Case 1 illustrate the improvement in the model's comprehension ability after RL training. Case 2 illustrate the improvement in the model's prioritization skill ability and reasoning ability after RL training.

Case 1(Before RL): Comprehension Ability

User Query: "Adjust the living room air conditioner to 20 degree and enable vertical swing."

Smart Home Agent: "The living room air conditioner has been set to 20 degree with vertical swing enabled."

User Query: "Do the same for the master bedroom and guest bedroom, and open all the curtains as well."

Smart Home Agent (Failed):

- `[Query]` → Query the device specification of the curtains.
- `[Device Control]` → Open the curtains.
- `[Response]` → The master bedroom curtain and guest bedroom curtain have both been opened.
- *Outcome:* The assistant doesn't set the air conditioners for both the master bedroom and the guest bedroom.

Case 1(After RL): Comprehension Ability

User Query: "Adjust the living room air conditioner to 20 degree and enable vertical swing." **Smart Home Agent:** "The living room air conditioner has been set to 20 degree with vertical swing enabled."

User Query: "Do the same for the master bedroom and guest bedroom, and open all the curtains as well."

Smart Home Agent (Succeed):

- `[Query]` → **Query the device specification of the air conditioners.**
- `[Device Control]` → **Set the air conditioners.**
- `[Query]` → Query the device specification of the curtains.
- `[Device Control]` → Open the curtains.
- `[Response]` → **The master bedroom and guest bedroom air conditioners have both been turned on, set to 20 degree, and vertical swing has been enabled.** The master bedroom curtain and guest bedroom curtain have both been opened.
- *Outcome:* The assistant completed both the air conditioner and curtain actions, correctly understanding "do the same".

Case 2(Before RL): Prioritization Skill Ability and Reasoning Ability

User Query: "It feels a little cool. Change the air conditioner to natural breeze."

Smart Home Agent: "There are air conditioners running in multiple rooms. Which room would you like to adjust?"

User Query: "The master bedroom."

Smart Home Agent (Failed):

- `[Device Control]` → Try to set the mode of the air conditioner to 'natural'.
- *Outcome:* The air conditioner has no "natural" mode thus this command is failed.

Case 2(After RL): Prioritization Skill Ability and Reasoning Ability

User Query: "It feels a little cool. Change the air conditioner to natural breeze."

Smart Home Agent: "There are air conditioners running in multiple rooms. Which room would you like to adjust?"

User Query: "The master bedroom."

Smart Home Agent (Succeed):

- `[Query]` → **Query the device specification of the air conditioner in master bedroom.**
- `[Reason]` → **Find that there is no 'natural' mode, and guess that the user want the 'fanonly' mode.**
- `[Device Control]` → **Set the mode of the air conditioner to 'fanonly'.**
- *Outcome:* Correctly reads the device status and reasonably maps 'natural breeze' to 'fan-only' mode.

G Comparison with Related Embodied / Smart-Home Benchmarks

We compare our SmartHome-Bench against two widely-used household simulation benchmarks: ALFWorld [48] and VirtualHome [49]. Raw success scores cannot be directly cross-compared across these benchmarks due to fundamental differences in observation space, action vocabulary, goal formulation and success criteria.

ALFWorld It adopts a text-based embodied paradigm, whose core tasks center on room navigation and object pick-and-place manipulation. Observations are textual descriptions of rooms and visible objects, and actions are embodied primitives such as `move`, `go to`, `pick-up`, `put-down`. A task is solved when objects reach target spatial locations. ALFWorld does not support conversational multi-turn intent disambiguation nor personalized preference-driven execution.

VirtualHome It builds a 3D household simulation focused on simulating human household activities. Observations combine visual and symbolic object status, and actions correspond to object-manipulation and human activity primitives. Task success is defined as finishing predefined activity scripts. Conversational intent disambiguation is absent, and support for personalized preference execution is very limited.

SmartHome-Bench (Ours) It is designed for LLM-driven smart-home IoT agent orchestration coupled with natural-language conversation. The core evaluation targets conditional multi-device control, ambiguous user intent resolution, multi-turn context tracking, and personalized preference-guided execution. Observations consist of user utterances, partial device-query feedback, and full dialogue history. Available actions include invoking device APIs, querying environment states, and issuing natural-language clarification questions to users. Task completion is judged by verifiable boolean predicates over dynamic physical device states. Multi-turn conversational disambiguation and personalized execution are explicitly evaluated as dedicated task categories (QT3, QT4 and QT5).

Data source note: Properties for ALFWorld and VirtualHome are derived from their original publications. Specifications of SmartHome-Bench are documented in Section 4.1 and Appendix E of this paper, covering the five task categories QT1–QT5.

Although proprietary off-the-shelf LLMs achieve 84–86% *aggregate* success on SmartHome-Bench, this overall metric conceals substantial performance variance across task categories. As reported in the main Table 1, large performance gaps emerge on complex long-tail subsets including compositional multi-device control (QT2), ambiguous intent (QT3), context-aware multi-turn interaction (QT4), and personalized memory (QT5). Therefore, the discriminative capacity of SmartHome-Bench lies primarily in these challenging scenarios instead of trivial atomic device operations (QT1).

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 claims in the abstract and introduction are supported by the experimental results in Sec 4 and accurately reflect the scope and contributions of the work.

   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: The paper explicitly discusses limitations regarding assumptions, data distribution, and computational constraints. Please refer to the "Limitations and future work" part in the "Conclusion" section.

   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: [N/A]

Justification: The paper does not include formal theoretical proofs; contributions are primarily algorithmic and empirical.

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: All hyperparameters, network architectures, training protocols, and dataset splits are described in Section 3 and the appendix, providing sufficient detail for reproduction.

   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: [Yes]

Justification: Code and datasets will be released upon acceptance. Code has been submitted to `https://anonymous.4open.science/r/homeflow-527B/`

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: Training details including base model, learning rate, TP, PP, CP, SP, Rollout group size, Rollout temperature, KL loss coefficient, weights and so on are fully specified in Section 3 and the supplemental appendix.

   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: [N/A]

   Justification: Error bars are not reported due to the high computational cost of running multiple trials. Results are consistent across all tests, supporting the reliability of the conclusions.

   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: Computer resources specifications are reported in appendix.

   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 uses only self-built datasets and does not involve human subjects, sensitive data, or any practice that violates 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: The paper discusses positive impacts (verifiable training pipeline and data flywheel for high-quality training data generation and smart home agent training), and will not cause negative social impacts.

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: [N/A]

    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: We utilize opensouce LLMs for role play and data generation. The creators are properly credited and the license and terms of use are properly respected.

    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: [Yes]

    Justification: The proposed code will be released with documentation covering installation, data preparation, training, and evaluation scripts upon acceptance. Code has been submitted to https://anonymous.4open.science/r/homeflow-527B/

    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: [Yes]

    Justification: The proposed flywheel uses LLMs for role play and data generation.

    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.