
Constrained Reflective Planning for Physically Grounded LLMs

Eduard Anton (Student)¹ Yide Song (Student)² Hsiu-Chin Lin (Mentor)¹

Abstract

Reinforcement learning (RL) has improved the reasoning abilities of large language models (LLMs), but a gap remains between linguistically plausible plans and physically executable actions in embodied settings. We propose Constrained Reflective Planning, a framework that integrates physical constraints directly into the reasoning process through structured, machine-parsable representations. Plans are encoded as directed acyclic graphs (DAGs), and a programmatic validator provides fine-grained feedback used as dense rewards within a step-wise Group Relative Policy Optimization (GRPO) scheme. This enables the model to learn constraint-consistent reasoning policies. By turning planning into a closed-loop process that generates and validates action chunks, our approach reduces error accumulation in long-horizon tasks. Experiments on ALFWorld and CALVIN show improved task success, action validity, and multi-step planning consistency over strong baselines, highlighting the benefits of structured, verifiable reasoning for grounded planning.

1. Introduction

Recent advances in large language models (LLMs) and vision-language-action (VLA) systems have significantly improved robotic capabilities in natural-language understanding, multi-step reasoning, and cross-task generalization. However, a critical gap persists between linguistic plausibility and physical executability. Agentic frameworks such as ReAct (Yao et al., 2023) perform well in digital environments but lack grounding in robot kinematics and dynamics, often producing plans that are logically coherent yet physically infeasible. Other robotics-oriented approaches, such as SayCan (Ahn et al., 2022) and Inner Monologue

(Huang et al., 2022), attempt to mitigate this gap by using external affordance models or feedback prompts. However, because physical constraints are applied as post-processing rather than embedded within the planner’s reasoning, these methods limit the model’s ability to inherently generate executable plans.

Reinforcement learning (RL) methods for LLMs, such as Group Relative Policy Optimization (GRPO), offer a path toward integrating constraints into the reasoning process itself. However, unconstrained reasoning often leads to excessive model “overthinking”, increasing computational cost (Aghajohari et al., 2025) and, in embodied settings, causing accumulated execution errors as long-horizon plans deviate from the true environment state.

To address this, we propose Constrained Reflective Planning, a training framework that transforms the model’s internal reasoning into a structured, machine-parsable representation. Instead of free-form reflection, the model generates logic graphs composed of sub-goals, actions, preconditions, and safety constraints. These structured reasoning traces are evaluated prior to execution using a Programmatic Validator. This design enables a key capability: when violations occur, the validator can localize errors to specific components of the reasoning trace and produce structured diagnostic feedback. We use this feedback as a dense reward signal for GRPO, replacing traditional scalar rewards. By directly optimizing over verifiable reasoning traces, the model learns to produce plans that are consistent with physical constraints, rather than relying on post-hoc correction.

This approach establishes a closed-loop planning process where the model generates and validates action chunks instead of brittle open-loop sequences. By constraining planning at intermediate steps, we can measure and reduce error accumulation and improve reliability over long horizons. We evaluate our method on benchmarks such as ALFWorld (Shridhar et al., 2021) and CALVIN (Mees et al., 2022), where we expect improved task success rates, higher executable action ratios, and more efficient training due to richer feedback signals.

¹Department of Computer Science, McGill University, Montreal, Canada ²Department of Electrical and Computer Engineering, McGill University, Montreal, Canada. Correspondence to: Eduard Anton <eduard.anton@mail.mcgill.ca>, Yide Song <yide.song@mail.mcgill.ca>, Hsiu-Chin Lin <hsiu-chin.lin@cs.mcgill.ca>.

Our main contributions are:

1. **Parsable Reflective Reasoning:** We introduce a structured and decomposable representation of reasoning traces that enables programmatic parsing into a graph data structure and fine-grained verification of intermediate planning steps.
2. **Validator-Driven Dense Reward:** We propose a reward mechanism that converts validation feedback into dense and localized diagnostic signals, enabling GRPO to optimize reasoning using structured, non-scalar rewards.
3. **Constrained Closed-Loop Planning:** We develop a planning framework that generates validated action chunks, reducing accumulated execution errors and improving robustness in long-horizon tasks.

2. Related Work

2.1. Grounding LLMs in Physical Environments

Prior work addresses the gap between high-level reasoning and physical execution through two main approaches.

First, interactive reasoning frameworks such as ReAct (Yao et al., 2023) and more advanced agents like DeepSeek-R1 (Guo et al., 2025) improve multi-step decision-making through iterative reasoning. However, these methods operate primarily in digital environments and do not explicitly model physical constraints.

Second, robotics-focused approaches incorporate grounding through either external affordance models or large-scale training. SayCan (Ahn et al., 2022) uses a learned affordance function to filter actions, while Vision-Language-Action models such as OpenVLA (Kim et al., 2024) and Octo (Team et al., 2024) learn implicit grounding from large robotic datasets. These methods either rely on post-hoc filtering or require significant data and compute, and offer limited interpretability of constraint satisfaction.

In contrast, we integrate constraint verification directly into the reasoning process via structured, machine-parsable representations, enabling fine-grained validation during planning rather than after action selection.

2.2. Feedback and the Overthinking Bottleneck

Robust execution in physical environments requires continuous feedback to handle stochastic failures. Systems such as AutoRT (Ahn et al., 2024) incorporate human intervention during low-confidence decisions, while RT-H (Belkhale et al., 2024) uses language feedback to refine actions during execution. At the same time, reinforcement learning approaches for LLM reasoning introduce a new challenge:

unbounded reasoning. Prior work (Aghajohari et al., 2025) shows that RL-trained reasoning models tend to generate excessively long reasoning traces, increasing computational cost. In embodied settings, this leads to accumulated execution errors, as the environment evolves while the agent continues reasoning.

We address this by constraining reasoning to short, validated action chunks, reducing both computational overhead and error accumulation.

2.3. Programmatic Action Representations

To improve execution reliability, recent work has explored programmatic representations of actions. Methods such as Code as Policies (Liang et al., 2023) and RoboCodeX (Mu et al., 2024) generate executable code (e.g., Python), enabling precise control flow and parameterization beyond natural language descriptions. However, these approaches typically generate full open-loop programs, which are brittle in dynamic environments: failures in early steps can invalidate the entire sequence.

We instead use structured representations but restrict generation to partial plans, enabling validation and execution at intermediate steps. This retains the precision of programmatic representations while improving robustness to environmental changes.

2.4. Structured Reasoning Representations via Graphs

Recent work has explored representing model reasoning as structured graphs rather than free-form text. In particular, (Zhang et al., 2025) introduces a framework where natural language reasoning traces are parsed into directed acyclic graphs (DAGs) capturing variables and their dependencies, and can be rendered back into natural language. This representation enables explicit separation between reasoning structure and surface form, allowing reasoning processes to be programmatically analyzed and compared against a ground-truth graph. Such graph-based designs provide fine-grained interpretability and support process-level evaluation by identifying incorrect operations or dependencies within a reasoning trace. However, these methods are primarily used for controlled evaluation and analysis rather than for optimizing reasoning policies.

In contrast, we leverage structured, parsable reasoning representations not only for verification but as a training signal, enabling direct optimization of reasoning through validator-derived feedback in an embodied setting.

3. Datasets

3.1. Benchmarks

We evaluate our approach on two complementary benchmarks: ALFWorld (Shridhar et al., 2021) and CALVIN (Mees et al., 2022), which together capture both grounded execution and structured planning.

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based household environment grounded in PDDL dynamics, where agents execute multi-step tasks via language-conditioned actions. Each episode provides a goal, textual observations, and admissible commands. The environment exposes structured state information through `game_state.facts`, enabling programmatic verification of preconditions and effects. This makes ALFWorld suitable for evaluating long-horizon planning under partial observability, where errors can accumulate over time.

CALVIN. To isolate planning ability, we also evaluate on the CALVIN (Mees et al., 2022) symbolic task domain. We use its 34 primitive action definitions to construct text-based planning problems without vision or low-level control. Given a symbolic state and goal, the model must predict valid actions consistent with precondition-effect logic. This setting removes perception and control noise, allowing precise evaluation of reasoning correctness.

3.2. Training Data

We use the ALFWorld expert trajectory dataset (`alfworld.expert_trajectories.json`), which contains 2,817 trajectories generated by a PDDL planner. Each trajectory is converted into supervised fine-tuning (SFT) examples by transforming every step into a structured `(system, user, assistant)` tuple.

The assistant output corresponds to the expert action (`action_chunks.command`). This yields 42,234 training examples across 816 unique goals, summarized in Table 1.

Table 1. Statistics of the ALFWorld expert trajectory dataset.

Stat	Value
Trajectories / SFT examples / unique goals	2,817 / 42,234 / 816
Top task verbs	put (22,191), clean (5,333), cool (4,000), heat (3,976), find (2,974), examine (1,912), look (1,848)
Expert trajectory length (min / mean / max)	3 / 15.0 / 49
User / assistant content length (mean words)	41 / 109

3.3. Evaluation Protocol

ALFWorld. All results are reported on the `valid_unseen` split, where environments are not observed during training.

CALVIN (symbolic). We construct evaluation tasks using CALVIN’s symbolic action space. Each instance consists of a state and goal, and the model must predict valid actions. We evaluate on two tasks, each containing 50 randomly generated 5-step task chains.

3.4. Evaluation Metrics

ALFWorld. We evaluate both task completion and planning efficiency using a curriculum-based setup:

- **Curriculum Success Rate (SR):** The model is initialized r steps away from the goal by replaying expert actions, with $r \in \{1, 3, 5\}$. This measures the ability to complete tasks from varying distances.
- **Average Model-Steps:** The number of steps required to reach the goal from the r -step initialization, measuring planning efficiency.

CALVIN. We evaluate reasoning accuracy over symbolic state transitions:

- **Next-Action Accuracy:** Evaluated over 200 prompts, this measures the exact-match percentage of the model predicting the single valid next action given a specific symbolic state and goal.
- **Valid Prefix Length (5-step):** When generating a full 5-step plan sequentially, this metric counts the average number of continuous correct steps starting from the beginning of the sequence. It strictly penalizes if there is any logical chain break.
- **Step Validity:** When generating a full 5-step plan sequentially, this metric counts how many steps are valid under the current symbolic preconditions.

4. Approach

4.1. Hypothesis

We hypothesize that constraining reasoning traces to a structured, machine-parsable Directed Acyclic Graph (DAG) improves both learning efficiency and execution reliability in embodied planning. Specifically, we test three effects: (i) a more stable supervised signal due to a fixed output schema, (ii) denser and more informative reinforcement learning rewards via structured state comparison, and (iii) improved long-horizon execution through explicit subgoal

decomposition and cycle prevention. To isolate these effects, we compare against both unstructured baselines and a DAG-SFT-only model without reinforcement learning.

4.2. Baselines

Qwen3-8B (base). We use the pretrained Qwen3-8B model with the same prompting format (DAG + action chunk) and a fixed 30-step interaction budget. This baseline controls for improvements due to training rather than model capacity.

GPT-4o-mini. We include GPT-4o-mini as a strong closed-source baseline, using identical prompts. This reflects common practice in ALFWorld and provides a high-performance reference.

Qwen3-8B + DAG-SFT (ours). We train Qwen3-8B using supervised fine-tuning on 42,234 DAG-formatted examples. This baseline isolates the effect of structured supervision without reinforcement learning, allowing us to quantify the contribution of GRPO.

4.3. Proposed Method

We train our model in two stages: supervised fine-tuning (SFT) followed by reinforcement learning via GRPO.

4.3.1. STAGE 1: DAG-SUPERVISED FINE-TUNING

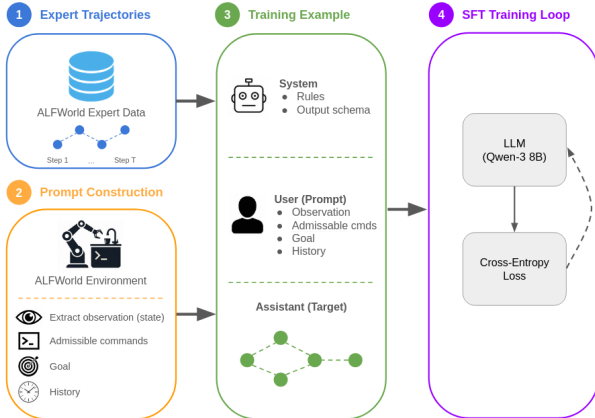


Figure 1. The Training Pipeline for the SFT DAG stage

We fine-tune a pretrained Qwen3-8B model to produce structured, machine-parsable reasoning traces in the form of directed acyclic graphs (DAGs). Starting from expert trajectories in ALFWorld, we convert each step into a supervised training example consisting of the current state (observation, admissible actions, goal, and history) and a target output. The target contains a DAG representing the remaining plan from the current step to task completion, and the next executable action. The DAG encodes subgoals and their dependencies, enforcing an explicit structure over the reasoning

process rather than free-form text.

We perform supervised fine-tuning using LoRA, optimizing the model to generate the target DAG and action given the input state. The training objective is standard cross-entropy loss applied only to the assistant output, ensuring the model learns to produce valid structured plans.

This stage serves two purposes: (i) it constrains the output space to a fixed schema, improving training stability, and (ii) it teaches the model to decompose long-horizon tasks into structured intermediate steps. The resulting model provides a baseline and a strong initialization for the subsequent GRPO stage.

4.3.2. STAGE 2: GRPO WITH STEP-WISE STRUCTURED REWARDS

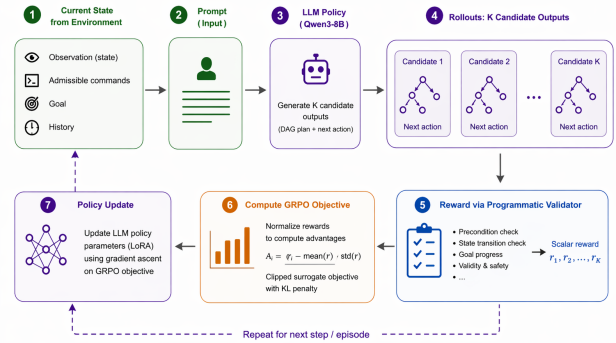


Figure 2. The Training Pipeline for the GRPO stage

We optimize the DAG-SFT model with Group Relative Policy Optimization (GRPO). The key design choice is that GRPO is applied *per environment step*, not per full episode. This is necessary because GRPO requires all completions within a group to be sampled from the same prompt so that the group mean defines a valid relative baseline. In long-horizon environments such as ALFWorld, grouping by episode would violate this almost immediately: multiple rollouts diverge after the first action and no longer share the same context, while the episode reward remains sparse and delayed.

We therefore form one GRPO group at each decision step. Given the current observation, admissible commands, action history, and previous step context, we construct a single prompt and sample $N = 8$ completions from it. Each completion contains a DAG plan and a candidate next action. Every candidate is evaluated independently in a shadow TextWorld instance reset to the current state, and only that candidate’s action is executed in the shadow environment to compute its reward. Thus, all N rewards correspond to different one-step decisions from the same state. The highest-reward valid action is then executed in the real en-

vironment to advance to the next step. This preserves the GRPO grouping assumption and provides dense training signals throughout long trajectories.

To improve stability on long-horizon tasks, we use a level-gated reverse curriculum. Instead of always starting from the beginning of an episode, training begins from partially completed expert trajectories. At curriculum level N , the environment is advanced with expert actions up to the point where N steps remain, and the model is responsible only for the suffix. The level is increased when the model’s win rate over a sliding window exceeds a fixed threshold, with occasional replay of earlier levels to reduce forgetting. This curriculum allows the policy to first learn short valid suffixes before solving full trajectories.

The reward is designed to be dense, verifiable, and grounded in simulator state changes. The first component is a potential-based reward computed from the PDDL fact set $F(s)$ exposed by the simulator. We define milestone values over goal-relevant predicates, e.g., `holds` for pickup, property predicates such as `isHot` or `isClean` for object transformation, and `inReceptacle` for final placement. Let $\Delta F(s \rightarrow s') = F(s') \setminus F(s)$ denote the fact difference after one action. We compute the potential only on newly introduced facts and only for entities appearing in the current goal, preventing the model from collecting reward by manipulating irrelevant objects. To avoid repeatedly rewarding the same milestone, we maintain the maximum potential reached so far in the episode, $\Phi_{\max}$, and define

$$R_{\text{potential}} = \max(0, \Phi_{\text{step}} - \Phi_{\max}),$$

so that each milestone is rewarded at most once per episode.

Potential reward alone does not cover necessary actions such as navigation, which may change location but not trigger a milestone predicate. We therefore add a second term, *expert state-match reward*, based on agreement between the model’s state transition and the expert’s state transition. Let ΔF_{model} be the fact diff induced by the model’s candidate action from the current state, and let ΔF_t^* be the fact diff induced by the expert’s next action from that same state. We define

$$r_{\text{state_match}} = 0.3 + 0.7 \cdot \frac{|\Delta F_{\text{model}} \cap \Delta F_t^*|}{|\Delta F_t^*|},$$

with an additional bonus when both diffs are empty, which correctly rewards observation-only actions such as `look`. This term gives credit to intermediate transitions that are useful but not directly captured by milestone rewards.

The final reward combines structural validity, action feasibility, task progress, and task completion:

$$R_{\text{total}} = R_{\text{format}} + R_{\text{admiss}} + R_{\text{potential}} + R_{\text{state_match}} + R_{\text{win}} + P_{\text{noop}} + P_{\text{repeat}} + P_{\text{length}}. \quad (1)$$

Table 2. The reward terms, their ranges, and corresponding roles in the environment.

Term	Range / Value	Role
r_{format}	−1.0	Hard termination signal for invalid JSON formats.
$r_{\text{admissible}}$	−0.5	Penalty for commands not in the current admissible set.
$r_{\text{state_match}}$	[0.3, 1.0]	Reward for aligning the model’s state change with the expert’s path.
$r_{\text{potential}}$	[0.5, 1.5]	Reward for achieving task milestones (e.g., picking up, heating).
r_{win}	+1.5	Substantial bonus for completing the overall task.
p_{noop}	−0.1	Penalty for observation-only actions (e.g., <code>look</code>).
p_{repeat}	−0.2	Penalty for repeating recently executed actions (spinning).
p_{length}	Dynamic	Token bloat penalty (−0.001 per token exceeding 500).

Here, R_{format} penalizes invalid DAG/JSON outputs, R_{admiss} penalizes actions outside the admissible set, R_{win} rewards successful completion, and the penalty terms discourage no-op behavior, repeated spinning, and unnecessarily long outputs. The reward terms, their specific ranges (or values), and their corresponding roles are demonstrated in Table 2. Together, these rewards let GRPO optimize reasoning traces that are not only well-formed, but also physically valid and task-progressing.

5. Experiment

Hardware Single NVIDIA H100 80GB

Software torch, vllm, trl, transformers, peft, accelerate, datasets, bitsandbytes, alfworld, textworld, comet_ml, pyyaml

The hypermeter is setted as following table 3

Regarding the training duration, the SFT phase was executed for 160 steps, while the GRPO phase was trained for 4500 steps. In here, a ”step” refers to a single gradient update to model’s weight. In SFT context, it corresponds to processing one effective batch. In GRPO, it corresponds to one batch of generated rollouts.

6. Result

6.1. Performance on ALFworld

The result in table 4 demonstrates that the GRPO significantly outperforms the base model and the SFT model. SFT

Table 3. Hyperparameters for SFT and GRPO phases.

Hyperparameter	SFT	GRPO
Base Model	Qwen3-8B	SFT Merged
Learning Rate	2e-4	1e-5
Batch Size	2	1
Gradient Accum. Steps	8	10
Max Sequence Length	1536	1024
LoRA (r , α , dropout)	16, 32, 0.05	16, 16, 0.05
Num Generations (K)	-	8
KL Penalty (β)	-	0.04
Temperature	-	1.5
Top- p	-	0.95

model is struggling even for a short-range task ($r=1$), while the GRPO-fine-tuned model achieves a 90% success rate. This indicates that reinforcement learning with our reward setting could successfully teach the model how to recover from the current state and follow logical constraints to finish the task. However, the success rate declines as the distance increases to 5. This is due to the following reasons:

- **Redundant Action Loops:** the log indicates models are easy to fall into a redundant action loop, although our reward is designed to overcome such behavior. However, when it becomes a long-term horizon reasoning chain, it may still repeat the same action after it has already achieved a certain goal. For instance, in game 1, the model repeatedly executes examine alarmclock 1 three times after successfully taking the object.
- **Goal Drift in Long-Range Planning:** The log points out that the model starts to lose attention to its goal in long-range planning. Models sometimes fail to understand the environment state, for example, when it needs to pick up some object, it begins traveling different places (e.g., moving between shelf 1 and shelf 4 in Game 8) instead of performing an interaction with the object

6.2. Performance on CALVIN

6.2.1. THE DIFFERENCE BETWEEN INTUITION AND PLANNING

A key finding is the difference between single-step accuracy and planning capability in base mode. The Qwen3-8B base model achieves a high next-action accuracy of 90.8%. However, the average valid prefix is nearly zero, indicating that the model can guess a plausible next action, but is unable to maintain a coherent sequence over multiple steps. The accuracy drop in the SFT model and the GRPO model indicates that the SFT state may have caused the model to “overfit” the action in ALFworld, which may lose some ability to generalize

6.2.2. GRPO PLANNING CAPABILITY

Our GRPO model could improve the average valid prefix from 0.06 to 0.56 and the Step validity from 1.2% to 11.2%. It shows that the DAG-based reasoning learned in ALFworld has been successfully transferred as a generalized planning skill, allowing the model to respect the condition restriction, but current step validation in the 5-step plan indicates that the model may still fail to really “understand” symbolic constraints in CALVIN. Also, the root cause of the low average valid prefix is similar to long step planning in ALFworld, where the model is currently struggling to produce a long coherent plan

7. Conclusion

We presented Constrained Reflective Planning, a framework for improving the physical grounding of LLM-based planners through structured reasoning and validator-driven reinforcement learning. Our approach represents reasoning traces as machine-parsable DAGs, enabling fine-grained verification and dense reward design. By combining DAG-supervised fine-tuning with step-wise GRPO, we optimize not only for task completion, but also for action validity and intermediate state consistency.

Experiments on ALFWorld show that the proposed method substantially improves short- and mid-horizon task completion over both base and SFT-only baselines. On CALVIN’s symbolic planning domain, the model also achieves better multi-step planning consistency, suggesting that the learned structured reasoning transfers beyond the training environment. At the same time, performance degradation on longer horizons highlights that maintaining coherent plans over many steps remains challenging.

Overall, our results suggest that structured, verifiable reasoning is a promising direction for bridging the gap between language-level plausibility and physical executability. We hope this work motivates further research on grounded

Table 4. ALFWorld (valid_unseen, curriculum-placed evaluation). We evaluate each model at three curriculum distances-from-goal (remaining=1, 3, 5). Configuration: 10 games per level $\times$ 3 levels = 30 episodes per model.

Model	r=1 SR $\uparrow$	r=3 SR $\uparrow$	r=5 SR $\uparrow$	r=1 model-steps $\downarrow$	r=3 model-steps $\downarrow$	r=5 model-steps $\downarrow$
B1: Qwen3-8B base	0.10 (1/10)	0.10 (1/10)	0.10 (1/10)	1.6	3.7	3.3
B2: GPT-4o-mini	0.40 (4/10)	0.20 (2/10)	0.00 (0/10)	2.0	3.8	3.8
B3: Qwen3-8B + DAG-SFT	0.00 (0/10)	0.10 (1/10)	0.00 (0/10)	2.0	4.8	5.0
Ours: DAG-SFT + GRPO	0.90 (9/10)	0.80 (8/10)	0.30 (3/10)	1.1	3.7	4.7

Table 5. CALVIN planning transfer. We evaluate planning on CALVIN’s 34-task symbolic domain (text-based, no rollout). The best performance across two runs is reported.

Model	Next-action acc.	5-step plan: avg valid prefix	5-step plan: step validity
Qwen3-8B base	90.8%	0.06	1.2%
Qwen3-8B + DAG-SFT	60.4%	0.36	7.2%
Ours: DAG-SFT + GRPO	60.4%	0.56	11.2%

reasoning representations, dense environment-aware rewards, and robust long-horizon planning for embodied LLM agents.

8. Code

Our Code is published on GitHub: <https://github.com/PersoSirEduard/constrained-reflective-thinking-planner>

9. Contributions

- Eduard Anton: Responsible for SFT training stage and Evaluation of model in ALFworld
- Yide Song: Responsible for GRPO training stage and Evaluation of model in CALVIN
- Hsiu-Chin Lin (mentor): Provided guidance and feedback.

References

- Aghajohari, M., Chitsaz, K., Kazemnejad, A., Chandar, S., Sordoni, A., Courville, A., and Reddy, S. The markovian thinker: Architecture-agnostic linear scaling of reasoning, 2025. URL <https://arxiv.org/abs/2510.06557>.
- Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J.,

Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL <https://arxiv.org/abs/2204.01691>.

Ahn, M., Dwibedi, D., Finn, C., Arenas, M. G., Gopalakrishnan, K., Hausman, K., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kirmani, S., Leal, I., Lee, E., Levine, S., Lu, Y., Leal, I., Maddineni, S., Rao, K., Sadigh, D., Sanketi, P., Sermanet, P., Vuong, Q., Welker, S., Xia, F., Xiao, T., Xu, P., Xu, S., and Xu, Z. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024. URL <https://arxiv.org/abs/2401.12963>.

Belkhale, S., Ding, T., Xiao, T., Sermanet, P., Vuong, Q., Tompson, J., Chebotar, Y., Dwibedi, D., and Sadigh, D. Rt-h: Action hierarchies using language, 2024. URL <https://arxiv.org/abs/2403.01823>.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun,

- Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning, *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL <http://dx.doi.org/10.1038/s41586-025-09422-z>.
- Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner monologue: Embodied reasoning through planning with language models, 2022. URL <https://arxiv.org/abs/2207.05608>.
- Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., and Finn, C. Openvla: An open-source vision-language-action model, 2024. URL <https://arxiv.org/abs/2406.09246>.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control, 2023. URL <https://arxiv.org/abs/2209.07753>.
- Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022. URL <https://arxiv.org/abs/2112.03227>.
- Mu, Y., Chen, J., Zhang, Q., Chen, S., Yu, Q., Ge, C., Chen, R., Liang, Z., Hu, M., Tao, C., Sun, P., Yu, H., Yang, C., Shao, W., Wang, W., Dai, J., Qiao, Y., Ding, M., and Luo, P. Robocodex: Multimodal code generation for robotic behavior synthesis, 2024. URL <https://arxiv.org/abs/2402.16117>.
- Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL <https://arxiv.org/abs/2010.03768>.
- Team, O. M., Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y. L., Chen, L. Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. Octo: An open-source generalist robot policy, 2024. URL <https://arxiv.org/abs/2405.12213>.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models, 2023. URL <https://arxiv.org/abs/2210.03629>.
- Zhang, C., Neubig, G., and Yue, X. On the interplay of pre-training, mid-training, and rl on reasoning language models, 2025. URL <https://arxiv.org/abs/2512.07783>.