TRAIN AN IMAGINATIVE AGENT

Jiaxin Ge[1,*] **Junyi Zhang**[1,*] **Tianning Chai**[1,*] **Hanjun Yoo**[1]
Chenyang Wang[2] **Xiuyu Li**[2] **Weijia Shi**[3] **XuDong Wang**[1,4]
Sewon Min[1] **Luke Zettlemoyer**[3] **Trevor Darrell**[1,†] **Haiwen Feng**[1,2,†]
[1]University of California, Berkeley [2]Impossible Research
[3]University of Washington [4]Duke University

ABSTRACT

A seemingly reasonable push in Sokoban can make the puzzle unsolvable. To avoid such mistakes, an agent has to anticipate the consequences of its actions before execution, especially when feedback from the real environment is sparse or actions are irreversible. We introduce Agentic Imagination (Agima), a training framework that trains an agent to imagine before acting, i.e. to be imaginative. Agima trains a single omni model to propose actions, generate images of their outcomes, and then reason over these imagined futures to decide what to execute in the real environment. Rather than committing to a single plan, the agent can simulate a sequence of actions, explore alternatives, and compare possible futures. We first use supervised fine-tuning (SFT) to unify policy and world modeling within a single omni model. This enables the agent to predict future outcomes and perform basic forms of imagination, such as forward thinking and counterfactual branching. We then use agentic reinforcement learning (RL) to teach the agent to reason with its imagined futures and use them effectively for planning. Across four environments spanning Maze2D, Sokoban, Pacman, and PushT, Agima generalizes to harder settings unseen during training. Trained with SFT alone on two-box Sokoban, Agima solves 40% of four-box puzzles while seeing the real board only once every five moves, compared with an action-only planner that only solves 2%. After RL, new behaviors emerge in how Agima uses its imagination: it learns a new unified format that was unseen during the SFT stage, learns to conduct forward thinking only when necessary, and branches more frequently at critical decision points. An agent that can picture where its actions lead can learn when and where to look ahead, that is, to plan. Project page: https://agima.net/.

1 INTRODUCTION

Agents take actions and receive observations from the environment (Sutton & Barto, 2018; Yao et al., 2023b), but making good decisions often requires anticipating the consequences of actions several steps ahead. When environment interactions are expensive or actions are irreversible, directly testing alternative actions through execution may not be feasible. For example, in Sokoban, a seemingly reasonable push can block another box and make the puzzle unsolvable. To avoid such failures, agents can predict and evaluate the consequences of possible actions before executing them (Racanière et al., 2017).

We introduce *Agentic Imagination* (Agima), a training framework that teaches an agent to imagine future observations of its actions and reason over them to plan before acting in the real environment. We build Agima on an omni model that unifies visual generation and language reasoning within the same multimodal context (Zhou et al., 2025; Deng et al., 2025). The same model serves as both the policy model that proposes actions and as the world model that imagines their visual states. These predictions become observations that the agent can use for further planning: it can extend a rollout, explore counterfactual actions, or compare longer candidate plans before execution (Figure 1). Importantly, imagined and real observations share the same interface and can be used interchangeably as inputs to the model. This allows the agent to use imagined observations in place of real

*, †: Equal contribution.

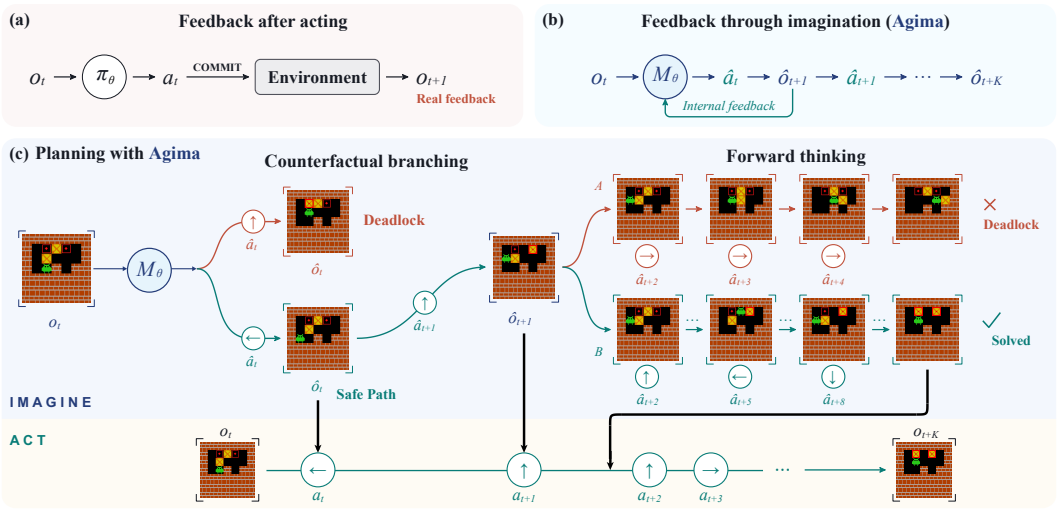


Figure 1: **Planning with agentic imagination.** **(a)** A conventional agent receives feedback only after acting in the environment. **(b)** Agima uses a single model to propose actions and predict the consequences of them. Each prediction serves as an internal observation for the next decision. **(c)** Planning with Agima in Sokoban: counterfactual branching rejects a deadlocking action, and forward thinking compares candidate plans before execution. The executed actions reach o_{t+K}, which grounds the next round of imagination. In (c), $\hat{o}_j$ is the predicted outcome of $\hat{a}_j$.

environment feedback to continue planning, while real observations can replace imagined ones after execution to ground the next round of planning.

To teach the agent to imagine futures and plan with them, we construct training trajectories that interleave action proposals, visual predictions, and outcome evaluation. This allows the agent to unify policy, world modeling, and reward modeling within a single omni model. We also teach the agent basic ways to reason over its imagined futures for planning. A *linear rollout* follows a single action sequence, predicting the state after each step. *Counterfactual branching* explores alternative actions at individual steps and rejects those that lead to failures. *Forward thinking* imagines and compares longer candidate plans before deciding which one to execute. Together, these trajectories teach the agent to generate imagined futures and use them to evaluate different actions before execution. We use these trajectories for supervised fine-tuning (SFT).

Once the agent can imagine futures, we want it to learn how to reason over them more effectively, rather than simply following predefined reasoning formats. Recent progress in reinforcement learning (RL) for reasoning and tool use in language models (Kumar et al., 2025; Jin et al., 2025) shows that models can learn new reasoning behaviors from outcome rewards. We adapt this idea to Agima using GRPO (Shao et al., 2024). During RL, the agent imagines and evaluates candidate plans, executes its selected actions in the real environment, and receives a binary reward based on task success. We optimize the agent's reasoning in language space and keeps its generated visual observations as context for planning. Unlike SFT, we do not provide additional expert demonstrations or supervise how the agent should organize its imagination. This allows the agent to internalize adaptive reasoning strategies. It learns from task success when to explore another plan, which alternatives to consider, and when to proceed with its current plan.

We evaluate Agima in Maze2D (Wang et al., 2026), Sokoban and Pacman (Racanière et al., 2017), and PushT (Florence et al., 2022; Chi et al., 2024), training on simpler instances and testing on harder settings excluded from training. With SFT alone, Agima directly generalizes from two-box to four-box Sokoban without further training. With real observations every five actions, it achieves 40% success, compared with 2% for an action-only planner fine-tuned from the same backbone. Forward thinking further improves long-horizon planning in Pacman using the same training instances as linear rollouts. Agentic RL further improves task success and leads to new behaviors in how Agima uses its imagination. For forward thinking, the agent learns a new reasoning format unseen during SFT: it first imagines a single plan and only explores another candidate when the first is predicted to fail. For counterfactual branching, the agent explores more alternatives and increasingly focuses

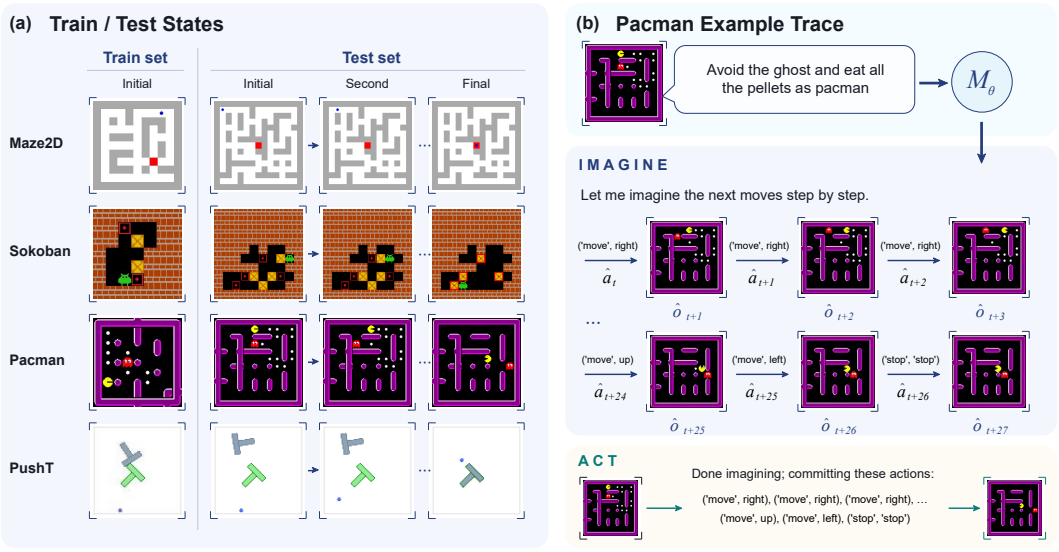


Figure 2: **Evaluation environments and imagined visual rollouts.** (a) Training and test states in four environments. (b) Agima interleaves actions and predicted states, then executes the actions.

its branches on critical decisions where a wrong action could make the puzzle unsolvable. These behaviors emerge from task-level rewards without additional supervision on how to organize its imagination.

Our contributions are threefold. First, we introduce Agima, a training framework for agentic imagination that teaches agents to explore, evaluate, and revise plans using predicted visual observations before acting. Second, we show that agentic imagination generalizes to out-of-domain settings across four environments, with further gains from learning to compare imagined futures using the same underlying training instances. Third, we show that agentic RL teaches Agima to organize its own imagination through task-success rewards, eliciting adaptive reasoning strategies that determine when to explore another plan and where to examine alternative actions.

2 Learning Agentic Imagination

Agentic imagination (Agima) trains an agent to imagine the future observations of its actions and reason over these imagined futures before acting in the environment. Given the current observation and a goal, the agent proposes an action, generates an image of its predicted outcome, and uses this imagined observation to continue planning. We build Agima on a single omni model that unifies language reasoning and visual generation. This allows the same model to both propose actions and predict their consequences, serving as both the agent policy and the world model. Our training framework has two stages. First, supervised fine-tuning (SFT) teaches the agent to unify policy and world modeling within a single model, allowing the agent to imagine future observations and perform basic reasoning over its imagined futures through different planning traces. Second, reinforcement learning (RL) teaches the agent to use its imagination more effectively for planning, learning when to explore additional futures and which alternatives to consider using only task-level rewards.

2.1 Planning with Imagined Observations

Consider an agent at time t with visual observation o_t and goal g. Under action chunking (Zhao et al., 2023), an action-only agent proposes a sequence of K actions:

$$A_t^{(K)} = (\hat{a}_t, \hat{a}_{t+1}, \ldots, \hat{a}_{t+K-1}) .$$

Agentic imagination interleaves proposed actions with predicted future observations:

$$\tau_t^{(K)} = (\hat{a}_t, \hat{o}_{t+1}, \hat{a}_{t+1}, \hat{o}_{t+2}, \ldots, \hat{a}_{t+K-1}, \hat{o}_{t+K}) .$$

Each predicted state becomes an *internal observation* for subsequent planning. The agent can therefore propose its next action from imagined consequences without waiting for real feedback. Once

Table 1: **Performance comparison across Pacman, Maze, Sokoban, and PushT environments.** K indicates the number of actions generated before feedback. Reactive ($K = 1$) receives ground-truth feedback after every action. Pacman, Maze, and Sokoban use the hard splits. For PushT, we report success accuracy and final coverage on the *far* split. A7B denotes 7B active parameters (14B total); No Imagine is trained to predict actions without intermediate visual predictions. Bold indicates the best result at each K (including ties); gains are relative to Agima-A7B (No Imagine).

K	Group	Method	Pacman	Maze	Sokoban	PushT	
						Acc.	Final Cov.
1	Closed	Gemini 3.1 Pro	0%	38%	3%	0%	0.2%
	Closed	GPT-5.6 Sol	0%	48%	32%	0%	1.8%
	SFT	Agima-A7B	**84%**	**62%**	**47%**	**42%**	**45.9%**
3	Closed	Gemini 3.1 Pro	0%	21%	4%	0%	2.2%
	Closed	GPT-5.6 Sol	0%	39%	26%	0%	3.0%
	SFT	Qwen2.5-VL-7B	0%	40%	4%	4%	8.48%
	SFT	Qwen3-VL-8B	0%	**85%**	10%	6%	11.77%
	SFT	Agima-A7B (No Imagine)	25%	54%	10%	12%	18.6%
	SFT	Agima-A7B	**38%** (+13%)	59% (+5%)	**53%** (+43%)	**24%** (+12%)	**31.7%** (+13.1%)
5	Closed	Gemini 3.1 Pro	0%	20%	6%	0%	1.8%
	Closed	GPT-5.6 Sol	1%	34%	20%	1%	8.3%
	SFT	Qwen2.5-VL-7B	0%	28%	0%	2%	4.07%
	SFT	Qwen3-VL-8B	0%	30%	3%	1%	3.41%
	SFT	Agima-A7B (No Imagine)	1%	23%	2%	1%	6.7%
	SFT	Agima-A7B	**37%** (+36%)	**40%** (+17%)	**40%** (+38%)	**18%** (+17%)	**28.5%** (+21.8%)
10	Closed	Gemini 3.1 Pro	0%	15%	4%	0%	2.0%
	Closed	GPT-5.6 Sol	1%	**34%**	**23%**	0%	8.7%
	SFT	Qwen2.5-VL-7B	0%	0%	0%	2%	5.36%
	SFT	Qwen3-VL-8B	0%	0%	0%	0%	5.55%
	SFT	Agima-A7B (No Imagine)	0%	2%	0%	1%	5.3%
	SFT	Agima-A7B	**7%** (+7%)	21% (+19%)	15% (+15%)	**9%** (+8%)	**22.2%** (+16.9%)
∞	Closed	Gemini 3.1 Pro	0%	9%	13%	0%	0.8%
	Closed	GPT-5.6 Sol	2%	**36%**	**21%**	0%	5.5%
	SFT	Qwen2.5-VL-7B	0%	0%	0%	2%	5.60%
	SFT	Qwen3-VL-8B	0%	0%	0%	**3%**	5.87%
	SFT	Agima-A7B (No Imagine)	0%	0%	0%	**3%**	6.2%
	SFT	Agima-A7B	**6%** (+6%)	3% (+3%)	19% (+19%)	**3%**	**8.3%** (+2.1%)

a plan is selected, its proposed actions are executed as $a_{t+j} = \hat{a}_{t+j}$. The resulting real observation then serves as the starting point for the next round of imagination. Beyond a single rollout, the agent can also explore alternative actions, return to previous plans, and compare different imagined futures before deciding what to execute. We next describe how we train the agent to perform these different forms of imagination through supervised fine-tuning.

2.2 Supervised Learning from Imagination Traces

We build Agima on the BAGEL backbone (Deng et al., 2025), an omni model that can understand and generate both images and text. The backbone has 14B total parameters and 7B active parameters, denoted by A7B in our model names. During SFT, we train on interleaved sequences of text and images such as the imagined rollout in Section 2.1. The current observation and goal are provided as input, and the remaining text and images are generated sequentially under teacher forcing. We apply next-token cross-entropy to text outputs and the image-generation model's MSE objective to image outputs, combining them in the SFT loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{i \in \text{text}} \log p_\theta(x_i \mid x_{<i}), \quad \mathcal{L}_{\text{MSE}} = \sum_{j \in \text{image}} \|\hat{v}_j - v_j\|_2^2, \quad \mathcal{L}_{\text{SFT}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{img}} \mathcal{L}_{\text{MSE}}.$$

Training on these interleaved sequences teaches the model to both predict future observations and reason over them to decide subsequent actions. We design three types of imagination traces that differ in how the agent explores alternative futures.

Linear Rollout. A linear rollout alternates proposed actions and predicted observations:

$$[o_t, g] \to \hat{a}_t \to \hat{o}_{t+1} \to \hat{a}_{t+1} \to \hat{o}_{t+2} \to \cdots.$$

Table 2: **Replacing imagined observations at inference.** No Imagine is a separately trained action-only baseline; the four Agima-A7B conditions share checkpoint, episodes, and decoding within each environment. Blank replaces predictions with gray images, Stale with the latest real observation, and Oracle GT with ground-truth frames for the proposed actions.

Model	Condition	Sokoban	Pacman	Maze2D	PushT
No Imagine	—	2%	1%	23%	1%
Agima-A7B	Own predictions	40%	37%	40%	18%
	Blank	1%	0%	8%	1%
	Stale	0%	0%	8%	0%
	Oracle GT	43%	81%	69%	47%

Each imagined observation is fed back into the sequence before the next action is generated, allowing the model to continue planning without a new observation from the environment.

Counterfactual Branching. A linear rollout commits to each action as it develops a plan. Counterfactual branching instead checks local alternatives while maintaining a primary rollout. At selected imagined states, the model tries a different action, predicts its immediate consequence, rejects the alternative if it is undesirable, and then continues along the main trajectory:

$$[o_t, g] \to \hat{a}_t \to \hat{o}_{t+1} \to \begin{cases} \hat{a}'_{t+1} \to \hat{o}'_{t+2} \to \text{reject}, \\ \hat{a}_{t+1} \to \hat{o}_{t+2} \to \cdots \to \hat{o}_j \to \begin{cases} \hat{a}'_j \to \hat{o}'_{j+1} \to \text{reject}, \\ \hat{a}_j \to \hat{o}_{j+1} \to \cdots \end{cases} \end{cases}$$

A single rollout may contain multiple such checks. The main trajectory follows the current plan, while the counterfactual branches explore alternative actions that may lead to failure.

Forward Thinking. While counterfactual branching explores local alternative actions at individual steps along a plan, forward thinking allows the agent to conduct long-horizontal imagination and compare different complete plans before deciding what to execute. Each candidate is rolled forward over a longer horizon:

$$[o_t, g] \to \begin{cases} \hat{a}_t^{(1)} \to \hat{o}_{t+1}^{(1)} \to \hat{a}_{t+1}^{(1)} \to \hat{o}_{t+2}^{(1)} \to \cdots, \\ \hat{a}_t^{(2)} \to \hat{o}_{t+1}^{(2)} \to \hat{a}_{t+1}^{(2)} \to \hat{o}_{t+2}^{(2)} \to \cdots. \end{cases}$$

The model can therefore compare how different plans unfold before choosing which one to execute.

2.3 Reinforcement Learning for Adaptive Imagination

SFT teaches the model how to generate and reason with imagination traces, but the demonstrations also specify which alternatives to explore and compare. We use RL to optimize these choices directly from task success. Starting from the SFT model, we apply GRPO (Shao et al., 2024). For each problem, the model generates a complete planning trace and final executable actions without receiving additional observations from the real environment. The selected actions are then executed, producing a binary task-success reward. We obtain group-relative advantages by standardizing rewards within each group of sampled traces. GRPO updates the language tokens in the planning trace; generated images remain in the multimodal context but receive no direct RL loss. We use a group size of 8 and batch size of 16. Additional optimization details are provided in Appendix B.

3 Generalization and Planning with Imagination

We evaluate two capabilities enabled by agentic imagination. First, we ask whether imagined observations can support multi-step planning without frequent feedback from the environment, and whether this ability generalizes beyond the configurations seen during training. Second, we ask whether the agent can use imagined futures not only to extend a rollout, but also to evaluate alternatives before acting. We study these questions across four environments with qualitatively different dynamics: Maze2D (Wang et al., 2026) for spatial navigation, Sokoban (Racanière et al., 2017) for irreversible object interactions, Pacman (Racanière et al., 2017) for reactive dynamics,

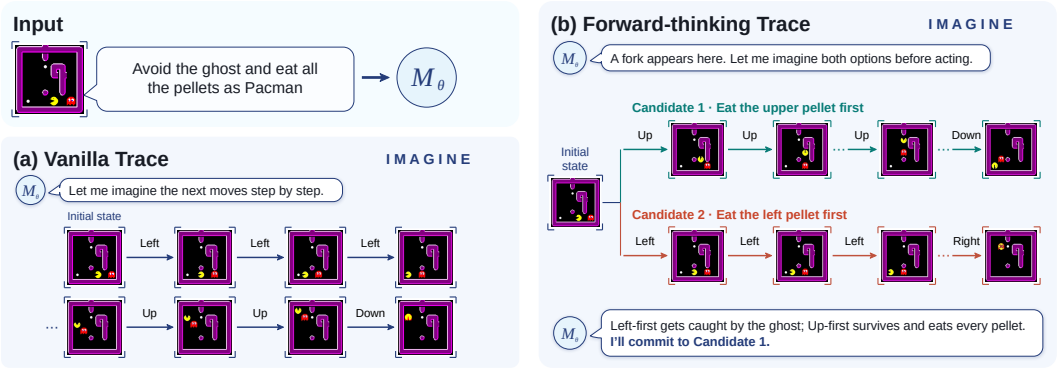


Figure 3: **Forward thinking compares alternative futures before execution.** A Pacman example (Table 3(b)): (a) Agima-A7B is caught after executing a single rollout; (b) Agima-A7B (Forward) compares two imagined futures and selects Candidate 1, which succeeds.

and PushT (Florence et al., 2022; Chi et al., 2024) for continuous contact dynamics. Figure 2 illustrates the environments and an example imagined rollout. Full environment details are provided in Appendix A.

3.1 Agima Enables Easy-to-Hard Generalization

Setup. We first evaluate linear imagination rollouts. For each environment, we train on simpler task configurations and evaluate on harder or unseen configurations excluded from training (Figure 2(a)). Let K denote the number of actions planned and executed before the next ground-truth observation is provided. The fully reactive setting $K = 1$ provides a reference with feedback after every action. At $K = \infty$, no intermediate environment observation is provided during the rollout. We denote the model trained with linear imagination rollouts as Agima-A7B. Our primary controlled baseline is Agima-A7B (No Imagine), which is separately fine-tuned from the same BAGEL backbone on the same task data, but predicts action chunks without intermediate visual states. We also supervised fine-tune Qwen2.5-VL-7B (Bai et al., 2025b) and Qwen3-VL-8B (Bai et al., 2025a) on the same task data as additional action-only baselines. Finally, we evaluate Gemini 3.1 Pro (Google DeepMind, 2026) and GPT-5.6 Sol (OpenAI, 2026) as proprietary baselines.

Imagination Transfers to Harder Settings and Improves Planning. Table 1 shows that Agima transfers from simpler training instances to harder and unseen settings across all four environments, especially when planning multiple actions without real feedback. At $K = 5$, Agima improves over the action-only model by 36 points in Pacman, 17 points in Maze2D, 38 points in Sokoban, and 17 points in PushT. For example, after training on two-box Sokoban, Agima reaches 40% success on four-box puzzles at $K = 5$, compared with 2% for the action-only model. Performance eventually degrades for very long open-loop planning: at $K = \infty$, both absolute performance and the advantage over action-only planning shrink in several environments.

Agima Relies on the Content of Its Imagined Observations. Replacing imagined observations with blank or stale frames sharply reduces success across all four environments (Table 2). Replacing predicted frames with ground-truth frames improves success by 29-44 percentage points in Pacman, Maze2D, and PushT, and by 3 points in Sokoban. These results show that Agima actually uses the content of its imagined observations for planning and benefits from more accurate predictions.

3.2 Reasoning over Alternative Imagined Futures

Linear rollouts use imagined observations to extend a single future. We next test whether the same learned world model can support richer planning operations: probing local counterfactual actions with branching and comparing long-horizon candidate futures with forward thinking.

Setup. All experiments use $K = \infty$, without intermediate observations from the real environment. Agima-A7B (Branching) and Agima-A7B (Forward) use the same training instances as the linear-rollout model, differing only in the structure of their imagination traces. The underlying training distribution remains unchanged. In Sokoban, models trained on standard two-box puzzles are evaluated on two-, three-, and four-box *Deadlock* and *Structural Fork* sets, testing delayed consequences

Table 3: **Success rates on Sokoban and Pacman reasoning tasks.** (a) Sokoban Deadlock and Structural Fork configurations with 2, 3, and 4 boxes. (b) Pacman route-order traps, where G denotes grid size, and F denotes the number of food items. Bold indicates the best result in each column; changes in parentheses are absolute percentage-point differences relative to Agima-A7B.

(a) Sokoban	**Deadlock**			**Structural Fork**		
Method	**2-box**	**3-box**	**4-box**	**2-box**	**3-box**	**4-box**
Gemini 3.1 Pro	**74%**	42%	19%	84%	34%	7%
GPT-5.6 Sol	44%	23%	17%	60%	20%	12%
Qwen2.5-VL-7B	1%	2%	0%	12%	2%	0%
Qwen3-VL-8B	6%	1%	1%	19%	2%	0%
Agima-A7B (No Imagine)	12%	5%	1%	15%	2%	1%
Agima-A7B	22%	39%	27%	87%	34%	22%
Agima-A7B (Branching)	30% (+8%)	39% (+0%)	**32%** (+5%)	84% (-3%)	39% (+5%)	**32%** (+10%)
Agima-A7B (Forward)	60% (+38%)	**48%** (+9%)	24% (-3%)	**93%** (+6%)	**46%** (+12%)	22% (+0%)

(b) Pacman	**G8**				**G10**
Method	**F2**	**F4**	**F6**	**F8**	**F8**
Gemini 3.1 Pro	1%	1%	0%	1%	0%
GPT-5.6 Sol	16%	19%	6%	8%	0%
Qwen2.5-VL-7B	0%	0%	0%	0%	0%
Qwen3-VL-8B	0%	0%	0%	1%	0%
Agima-A7B (No Imagine)	3%	0%	2%	0%	0%
Agima-A7B	7%	22%	16%	16%	5%
Agima-A7B (Forward)	**40%** (+33%)	**54%** (+32%)	**41%** (+25%)	**37%** (+21%)	**12%** (+7%)

and critical action choices, respectively. In Pacman, we evaluate route-order traps requiring long-horizon plan comparison on out-of-domain grid sizes. Training traces and evaluation details are provided in Appendices B and A.1.

Counterfactual Branching Helps at Critical Decision Forks. Counterfactual branching allows the agent to explore alternative actions at each step along the main trajectory. It is particularly helpful in the harder Structural Fork settings, where the agent encounters locally states that a seemingly reasonable action can lead to failure. Compared with the linear-rollout model, branching improves three-box success from 34% to 39% and four-box success from 22% to 32% (Table 3(a)). These results show that Agima can use imagined states to probe local alternatives that may lead to failure before actually committing to the next step.

Forward Thinking Improves Long-Horizon Plan Selection. Forward thinking allows the agent to imagine and compare different futures before execution, helping it avoid failures that only become apparent several steps later. On Sokoban Deadlock, forward thinking improves success from 22% to 60% on two-box puzzles and from 39% to 48% on three-box puzzles. The effect is particularly consistent on Pacman route-order traps: forward-thinking supervision improves success across every evaluated configuration, including from 22% to 54% on G8/F4 and from 5% to 12% on G10/F8 (Table 3(b)). Figure 3 illustrates why long-horizon comparison matters. With linear rollout, the model directly chooses the more tempting option: collecting the nearer food first, which appears safe because the ghost is still far away. However, when Pacman turns toward the remaining food, it can no longer escape the ghost and eventually dies. With forward thinking, Agima imagines alternative futures that collect the food items in different orders and discovers that collecting the farther food first leads to a successful outcome.

4 Adaptive Reasoning through Agentic RL

SFT teaches Agima to construct and reason over imagined futures, but the training traces follow predefined heuristics. Since Agima combines language-space reasoning with visual prediction, we can apply agentic RL to improve how the agent uses its imagination for planning. Specifically, we ask whether task-level rewards can teach the agent when to think ahead, when to reconsider a plan, and where to explore counterfactual alternatives.

Setup. We initialize RL from four SFT checkpoints trained on two-box Sokoban: Agima-A7B (No Imagine), Agima-A7B, Agima-A7B (Forward), and Agima-A7B (Branching). Agima-A7B

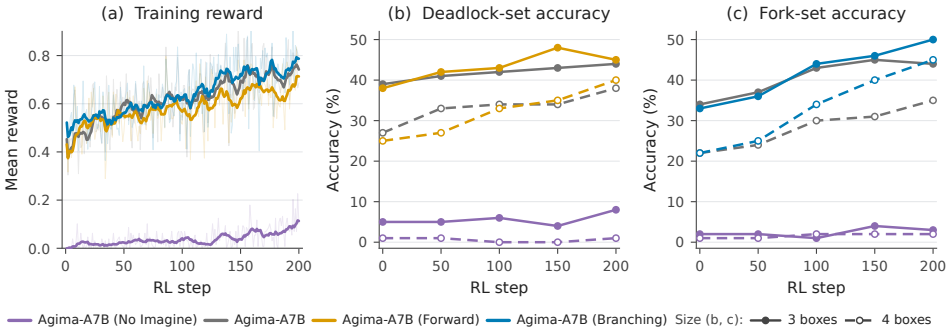


Figure 4: **RL improves training reward and held-out planning performance.** (a) Training reward during RL. (b–c) Accuracy on the Deadlock and Structural Fork evaluation sets. Solid and dashed lines denote three-box and four-box tasks, respectively.

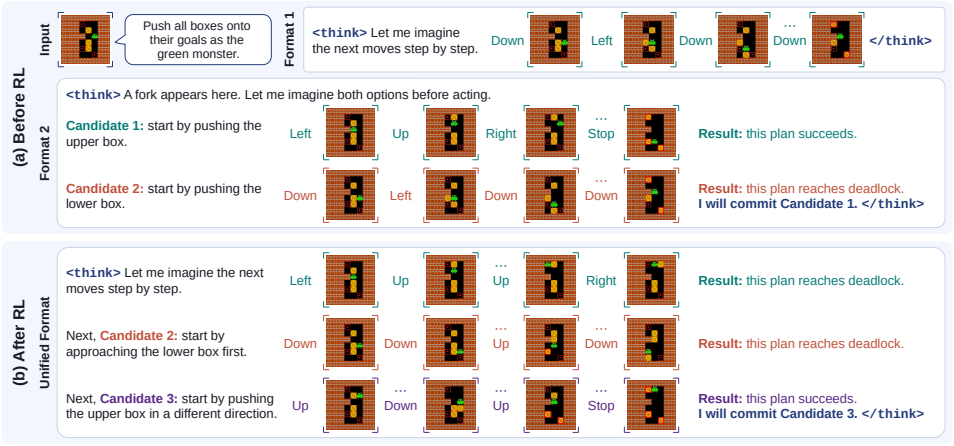


Figure 5: **RL induces a unified, adaptive planning format.** (a) SFT produces a linear rollout or a fixed two-candidate comparison. (b) After RL, the model evaluates each imagined plan and generates alternatives as needed, here rejecting two failed plans before finding a successful third.

(No Imagine) provides a no-imagination baseline, generating actions without intermediate visual predictions. For the latter two, we use mixed-format checkpoints trained on random mixtures of $\sim$50% linear rollouts and $\sim$50% of the corresponding reasoning traces. This keeps their pre-RL performance close to the linear-rollout checkpoint while letting the model being able to generate multiple reasoning formats, leaving RL to learn when and how to use them. Because ordinary two-box Sokoban is nearly saturated after SFT, we perform RL on ordinary three-box puzzles. The agent receives only a binary task-success reward, with no additional expert demonstrations or trace-level supervision. We train for 200 RL steps using GRPO, and evaluate on held-out three- and four-box Deadlock and Structural Fork sets. Additional details are provided in Appendix B.4.

RL Improves Planning Beyond SFT. Figure 4 shows that RL improves training reward for all four initializations. Even Agima-A7B (No Imagine) rises from near-zero reward to roughly 15% by the end of training, showing that task reward can improve the action policy. However, this gain transfers little to the held-out Deadlock and Structural Fork sets, where its accuracy remains near zero. In contrast, all three imagination-based initializations improve substantially on both training reward and held-out planning performance, including the four-box settings never seen during RL. The gain is particularly strong for Agima-A7B (Branching): despite starting near the linear-rollout Agima-A7B checkpoint, RL eventually pushes it beyond Agima-A7B on Structural Fork.

Forward Thinking Becomes Adaptive. RL teaches Agima when to explore alternative plans in forward thinking. During SFT, the training data only contains two predefined reasoning formats: (1) a linear rollout, or (2) a forward thinking trace that decides from the beginning to compare two candidates. After 100 steps of RL training, a new format emerges that combines these two behaviors: the agent first rolls out a plan and evaluates its predicted outcome. If the plan is predicted to succeed, it ends the reasoning trace and proceeds with the plan; otherwise, it generates another

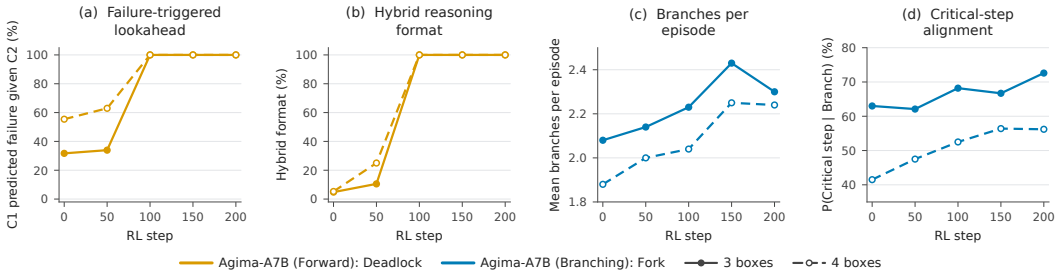


Figure 6: **RL induces adaptive use of imagination.** Forward thinking becomes increasingly failure-triggered, while branching explores more alternatives and shifts toward critical decision points.

candidate (Figure 5, Appendix C.1). Meanwhile, the model also learns when additional exploration is necessary. When a second candidate is generated, the first plan fails in 100% of cases, compared to 32% before RL. This suggests that the agent learns to reason over its predicted outcomes and only uses forward thinking when needed, rather than simply following the predefined formats seen during SFT. The agent also learns to explore beyond the two-candidate format seen during SFT which is trained on two-box puzzles. As shown in Figure 5, in the three-box setting, it learns to explore a third candidate after the first two plans fail.

Branching Becomes More Exploratory and Targeted. RL also changes the branching behavior of Agima. The model explores more alternatives once branching begins, with the average number of branches per trace increasing from 1.70 before RL to 1.95 after 150 steps of RL training. We define *Critical Branch Rate* as the fraction of branches opened at critical states, where at least one alternative action would make the puzzle unsolvable. As shown in Figure 6, this rate increases after RL, indicating that counterfactual reasoning becomes more concentrated on decisions where the choice of action matters. Meanwhile, the total number of imagined frames does not increase (Appendix C.3), suggesting that RL teaches the agent to use branching more effectively to avoid potential failures without increasing its overall imagination budget.

5 Related Work

Model-based planning. Model-based reinforcement learning learns a model of the environment dynamics and uses it to plan or to train a policy (Sutton, 1991; Ha & Schmidhuber, 2018; Hafner et al., 2019b;a; Schrittwieser et al., 2020). I2A (Racanière et al., 2017) is particularly relevant here: it uses learned environment rollouts to inform a model-free policy on Sokoban and MiniPacman. These tasks share the irreversible decisions and long planning horizons of our setting. Language models have also been combined with tree search, where they act as the policy, the world model, or the evaluator (Hao et al., 2023; Yao et al., 2023a), and recent video world models generate high-fidelity future frames conditioned on actions (Bruce et al., 2024; Parker-Holder & Fruchter, 2025; Valevski et al., 2025; Alonso et al., 2024; Yang et al., 2024; NVIDIA et al., 2025; Bar et al., 2025). I2A leaves the choice of imagined actions to a rollout policy, while the tree-search methods use an external search procedure. In Agima, action proposals and visual predictions come from the same model. Each prediction stays in the context for subsequent reasoning, so choosing whether to extend a rollout, try an alternative, or reject a plan becomes part of the generated trace.

Reasoning with generated images. Several recent methods include images in the reasoning process of multimodal models. Visual Sketchpad (Hu et al., 2024) calls tools to draw auxiliary lines and boxes, MVoT (Li et al., 2025) generates images of intermediate states in spatial reasoning, and ThinkMorph (Gu et al., 2026) alternates between text and image thoughts. Other work keeps the visual thought in a latent space (Zhang et al., 2025), and Su et al. (2025) provide a survey. In these methods, the image helps the model reach a final answer, and there is usually no single correct image to generate. UniWM (Dong et al., 2026) and WorldVLA (Cen et al., 2025) jointly predict actions and future images for navigation and manipulation, but they follow a single predicted trajectory and do not use the predicted images to compare alternative actions. In our work, each generated image is the predicted result of a specific action, and the model learns to carry out the planning steps itself, including counterfactual branching and forward thinking (Section 2).

Reinforcement learning for VLM agents. Reinforcement learning with outcome rewards has improved the reasoning of language models (Shao et al., 2024; Guo et al., 2025) and has been applied to multi-turn LLM agents (Wang et al., 2025b; Feng et al., 2025) and to VLM agents that act from visual observations (Zhai et al., 2024). Two works are closest to ours. VAGEN (Wang et al., 2025a) trains a VLM agent to describe the current state and predict the next state in its textual reasoning, and rewards the accuracy of these predictions; Sokoban is among its tasks. Visual Planning (Xu et al., 2026) represents a plan as a sequence of generated images and rewards each generated state. Neither method trains an agent to reason in language about its own visual predictions. We apply RL with a binary task-success reward to an agent of this kind. Only the text tokens are optimized. These tokens carry the agent's planning decisions, such as whether to imagine another plan and where to branch, while the generated images remain in the context as observations.

6 Conclusion

We present Agima, a framework that teaches agents to plan with predicted visual observations. Across four environments, Agima generalizes beyond training configurations and supports multi-step planning without intermediate feedback, counterfactual branching, and comparison of candidate futures. With agentic RL, Agima learns new behaviors, including a new reasoning format, conditioning forward thinking only on failure, and focusing branching on critical decisions. Our results show that a single omni model can be trained as an imaginative agent that can imagine possible futures and reason over them to guide its own actions.

Acknowledgment

We thank Angjoo Kanazawa, Ranjay Krishna, Joey Gonzalez, Rulin Shao, Zixian Ma, Xiaochuang Han, and Baifeng Shi for feedback on our manuscript.

References

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for World Modeling: Visual Details Matter in Atari. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 58757–58791. Curran Associates, Inc., 2024. doi: 10.52202/079017-1873. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6bdde0373d53d4a501249547084bed43-Paper-Conference.pdf.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025b. URL https://arxiv.org/abs/2502.13923.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation World Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801, June 2025.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative Interactive Environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards Autoregressive Action World Model, 2025. URL https://arxiv.org/abs/2506.21539.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, October 2024. ISSN 1741-3176. doi: 10.1177/02783649241273668. URL http://dx.doi.org/10.1177/02783649241273668.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging Properties in Unified Multimodal Pretraining, 2025. URL https://arxiv.org/abs/2505.14683.

Yifei Dong, Fengyi Wu, Guangyu Chen, Lingdong Kong, Xu Zhu, Qiyu Hu, Yuxuan Zhou, Jingdong Sun, Jun-Yan He, Qi Dai, Alexander G. Hauptmann, and Zhi-Qi Cheng. Towards Unified World Models for Visual Navigation via Memory-Augmented Planning and Foresight, 2026. URL https://arxiv.org/abs/2510.08713.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-Group Policy Optimization for LLM Agent Training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46375–46408. Curran Associates, Inc., 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/florence22a.html.

Google DeepMind. Gemini 3.1 Pro Model Card. Google DeepMind Model Cards, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Jiawei Gu, Yunzhuo Hao, Huichen Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. ThinkMorph: Emergent Properties in Multimodal Interleaved Chain-of-Thought Reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 141405–141447, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e5095602ad1c6a835e2b643ec4ed97d0-Paper-Conference.pdf.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu,

Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 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.

David Ha and Jürgen Schmidhuber. World Models, 2018. URL https://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019b. URL https://proceedings.mlr.press/v97/hafner19a.html.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with Language Model is Planning with World Model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah Smith, and Ranjay Krishna. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 139348–139379. Curran Associates, Inc., 2024. doi: 10.52202/079017-4423. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fb82011040977c7712409fbdb5456647-Paper-Conference.pdf.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning, 2025. URL https://arxiv.org/abs/2503.09516.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training Language Models to Self-Correct via Reinforcement Learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 54523–54549, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/871ac99fdc5282d0301934d23945ebaa-Paper-Conference.pdf.

Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. Imagine While Reasoning in Space: Multimodal Visualization-of-Thought. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36340–36364. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/li25cz.html.

NVIDIA et al. Cosmos World Foundation Model Platform for Physical AI. *arXiv preprint arXiv:2501.03575*, 2025.

OpenAI. GPT-5.6 Sol. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models. Google DeepMind, 2025. URL https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/.

Sébastien Racanière, Theophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-Augmented Agents for Deep Reinforcement Learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/9e82757e9a1c12cb710ad680db11f6f1-Paper.pdf.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, Linjie Li, Yu Cheng, Heng Ji, Junxian He, and Yi R. Fung. Thinking with Images for Multimodal Reasoning: Foundations, Methods, and Future Frontiers, 2025. URL https://arxiv.org/abs/2506.23918.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL http://dx.doi.org/10.1145/122344.122377.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. ISBN 9780262039246. URL https://mitpress.mit.edu/9780262039246/reinforcement-learning/.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion Models Are Real-Time Game Engines. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 73754–73776, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/b71ecea210f7159f31e46631fe5c838f-Paper-Conference.pdf.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. VAGEN: Reinforcing World Model Reasoning for Multi-Turn VLM Agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://arxiv.org/abs/2510.16907.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning, 2025b. URL https://arxiv.org/abs/2504.20073.

Zirui Wang, Junyi Zhang, Jiaxin Ge, Long Lian, Letian Fu, Lisa Dunlap, Ken Goldberg, XuDong Wang, Ion Stoica, David M. Chan, Sewon Min, and Joseph E. Gonzalez. VisGym: Diverse, Customizable, Scalable Environments for Multimodal Agents, 2026. URL https://arxiv.org/abs/2601.16973.

Yi Xu, Chengzu Li, Han Zhou, Xingchen Wan, Caiqi Zhang, Anna Korhonen, and Ivan Vulić. Visual Planning: Let's Think Only with Images. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 36706–36739, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3de60dc2ff7ac8866e805bbf457e2291-Paper-Conference.pdf.

Sherry Yang, Yilun Du, Seyed Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning Interactive Real-World Simulators. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 45210–45234, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c4d66eae503694424123b93ac0fbaf17-Paper-Conference.pdf.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-Tuning Large Vision-Language Models as Decision-Making Agents via Reinforcement Learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 110935–110971. Curran Associates, Inc., 2024. doi: 10.52202/079017-3522. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c848b7d3adc08fcd0bf1df3101ba6728-Paper-Conference.pdf.

Huanyu Zhang, Wenshan Wu, Chengzu Li, Ning Shang, Yan Xia, Yangyu Huang, Yifan Zhang, Li Dong, Zhang Zhang, Liang Wang, Tieniu Tan, and Furu Wei. Latent Sketchpad: Sketching Visual Thoughts to Elicit Multimodal Reasoning in MLLMs, 2025. URL https://arxiv.org/abs/2510.24514.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the Next Token and Diffuse Images with One Multi-Modal Model. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 6446–6469, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/12678c3948153f4bc391f51e2082bd6e-Paper-Conference.pdf.

Appendix Contents

A Environments and Evaluation Sets **16**

A.1 Planning Evaluation Sets 16

B Training Data and Optimization **17**

B.1 Action Chunking and Linear Rollout Data 17

B.2 Constructing Alternative-Future Traces 17

B.3 Training Data Examples and Statistics 17

B.4 Model and Training Setup 21

C Analysis of Adaptive Imagination **22**

C.1 Combining Linear Rollouts and Forward Thinking 22

C.2 Counterfactual Branching 22

C.3 Task Success and Rollout Length 23

D Qualitative Dynamics Predictions and Rollouts **24**

D.1 Maze2D Rollouts 24

D.2 Sokoban Rollouts 30

D.3 Pacman Rollouts 35

D.4 PushT Rollouts 44

D.5 Rollouts for different reasoning formats 51

D.6 RL Trajectories across Checkpoints 52

D.7 Failure Cases 63

A Environments and Evaluation Sets

The training and test configurations below are used for the easy-to-hard evaluation in Table 1. All grid sizes include the outer boundary walls. Maze2D, Sokoban, and Pacman use four movement actions: up, down, left, and right. PushT uses continuous (x, y) pusher targets. Training traces end with `STOP`. We report episode success rates for the three discrete tasks, and both success rate and mean final coverage for PushT.

Maze2D: Constrained Navigation. Maze2D is a grid navigation task from VisGym (Wang et al., 2026). The input image shows the agent, walls, and goal. The model must navigate to the goal while predicting its position after each move; an episode succeeds when the agent reaches the goal. We train on 11×11 mazes with 9×9 interiors and test on larger mazes.

Sokoban: Irreversible Object Dynamics. Sokoban is a box-pushing puzzle (Racanière et al., 2017). The input image shows the agent, boxes, walls, and targets. Movement into a box pushes it when the next cell is free. A wrong push can permanently trap a box or create a deadlock, so the model must predict how actions change the box configuration. Success requires placing all boxes on targets. For Table 1, we train on 7×7 two-box puzzles and test on four-box puzzles.

Pacman: Reactive Dynamics. Pacman adds a moving opponent, as in prior work on MiniPacman (Racanière et al., 2017). The input image shows Pacman, the red ghost, walls, and food pellets. After each Pacman move, the ghost moves by greedily chasing Pacman. Success requires collecting all food without being caught. We train on 9×9 grids with 8 food pellets. This training configuration is shared by the linear-rollout and forward-thinking data. We test on 12×12 grids with 12 food pellets and a closer initial ghost position.

PushT: Continuous Contact Dynamics. PushT is a continuous manipulation task (Florence et al., 2022; Chi et al., 2024). The input image shows the pusher, a T-shaped object, and its target pose. The model predicts pusher targets and the resulting object motion. Training uses initial object positions nearer the center; evaluation uses unseen positions farther from the center. Final coverage measures the fraction of the target covered by the object at the end of execution. We report its mean across episodes and the success rate, defined as the fraction of episodes with final coverage greater than 0.80.

A.1 Planning Evaluation Sets

Table 3 evaluates counterfactual branching and forward thinking on the Sokoban and Pacman sets below. These sets test decisions whose consequences become apparent only after several actions, separately from the easy-to-hard evaluation in Table 1. We report task success rates with $K = \infty$, without intermediate observations from the real environment.

Sokoban Deadlock and Structural Fork. Each family contains 2-box, 3-box, and 4-box evaluation sets, with 100 episodes per set. Deadlock instances contain locally legal pushes that can make the puzzle unsolvable, with harmful consequences becoming apparent later in the trajectory. Structural Fork instances offer multiple legal choices, some of which lead to dead ends. The supervised checkpoints are trained on standard 2-box puzzles; the 3-box and 4-box sets test transfer to larger object configurations.

Pacman route-order traps. We evaluate on out-of-domain grids of size 8×8 with 2, 4, 6, or 8 food items and 10×10 with 8 food items. Both grid sizes differ from the 9×9 training grids. In these instances, pursuing the nearer food first leaves no successful continuation, while a farther-first route has a replay-verified solution that clears the food, avoids the ghost, and ends with `STOP`.

We denote a configuration as Gx/Fy, where x is the grid size and y is the number of food items; for example, G8/F4 denotes an 8×8 grid with four food items.

B Training Data and Optimization

We first describe the supervised training traces and their statistics, then give the model and optimization settings separately.

B.1 Action Chunking and Linear Rollout Data

We segment expert trajectories into action chunks with $K \in \{1, 3, 5, 10, \infty\}$. Each example starts from an observation o_t and goal g. The action-only target contains the next K expert actions; the linear-imagination target interleaves each action with its successor image. The images used as training targets are observations from the expert trajectory. At inference time, the model generates these images as predicted observations and uses them to plan subsequent actions.

For example, a $K = 3$ training segment has the following schematic form:

$$\begin{aligned} \text{Input:} \quad & [o_t, g], \\ \text{Action-only target:} \quad & [a_t, a_{t+1}, a_{t+2}], \\ \text{Linear-imagination target:} \quad & [a_t, o_{t+1}, a_{t+1}, o_{t+2}, a_{t+2}, o_{t+3}]. \end{aligned}$$

After these three actions are executed, the next chunk starts from the new real observation. $K = 1$ gives feedback after every action, while $K = \infty$ uses the full remaining plan without intermediate real observations. The final segment ends with `STOP` when the expert task is complete. Figure 7 shows a qualitative example of different action chunk sizes.

B.2 Constructing Alternative-Future Traces

Counterfactual-branching and forward-thinking traces use the same underlying expert episodes as linear rollouts. We do not select additional training instances specifically for failures or difficult forks; only the imagination trace structure changes. The additional supervision teaches the model how to explore and compare imagined futures. Figures 8, 9, and 10 show examples of the three constructions below.

Counterfactual branching. For Sokoban, expert trajectories are scanned for critical states where the expert push keeps the level solvable and another legal push is certified unsolvable by moving a box against a wall. At up to three earliest such states, a trace inserts an alternative push, its successor image, a rejection, and a return to the expert continuation. The executable sequence after `</think>` remains the original expert solution. Episodes without a certified harmful alternative remain single-plan rollouts.

Forward thinking in Sokoban. Each trace compares two candidate box-pushing plans from the same initial state. One follows the expert solution; the other starts with a different first push and uses the best continuation found by search. Breadth-first search (BFS) labels each candidate as solved, unfinished at the action limit, or deadlocked. The target demonstrates both futures and selects a successful plan, preferring the shorter plan when both succeed. Candidate order is randomized.

Forward thinking in Pacman. Each trace begins with a shared prefix of the expert trajectory, reaching a decision state 2 to 10 moves into the episode. It then compares two continuations from that state: the expert continuation and an alternative that takes a different first move toward a nearer food item. The trace returns to the same decision state between candidates, compares their outcomes, and commits a complete plan. Candidate order is randomized. Figure 10 shows an example in which the alternative is caught by the ghost. The outcome proportions across all training traces are given in Table 5.

B.3 Training Data Examples and Statistics

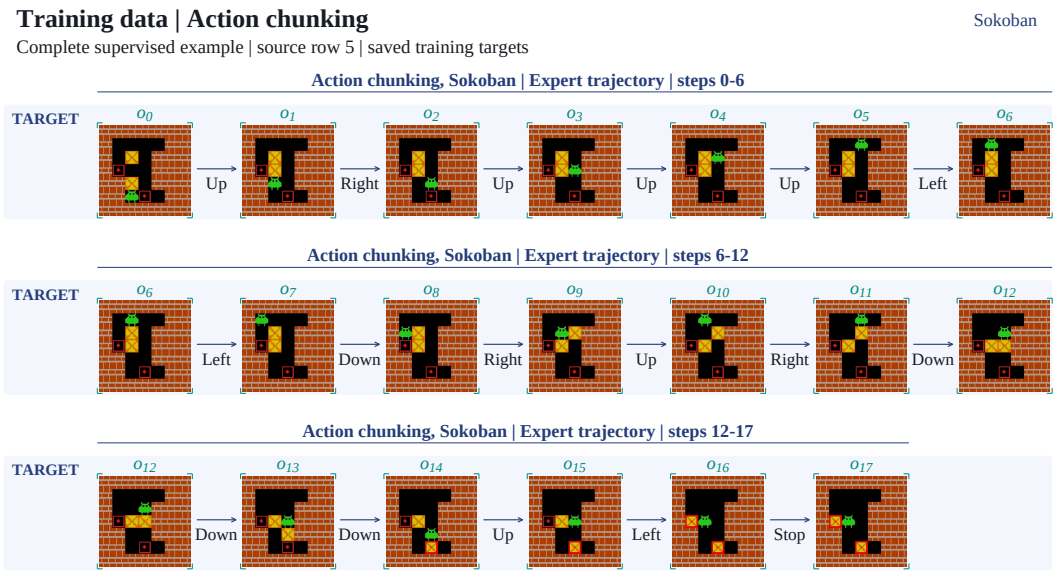
Table 4 summarizes the expert rollouts used to construct action chunks. Table 5 reports the alternative-future trace statistics. The Sokoban forward-thinking traces used for training contain two candidates; their outcome proportions are reported within that population. PushT expert trajectories are generated with PPO (Schulman et al., 2017). Training budgets are given separately below.

Table 4: **Linear-rollout training data per task.** Plan length counts the expert's actions excluding the final `STOP` and is reported as median (mean) [range]. Grid sizes include the outer wall.

	Maze2D	**Sokoban**	**Pacman**	**PushT**
Grid size	11×11	7×7	9×9	N/A
Training setting	maze navigation	2 boxes	8 food	diverse T start
Expert	shortest-path oracle	Sokoban solver	oracle planner	PPO policy
Action space	4 moves	4 moves	4 moves	continuous (x, y)
Action budget	40	20	80	30
Plan length	9 (9.9) [5–33]	10 (9.9) [2–19]	23 (22.5) [9–27]	20 (20.0) [4–29]

Table 5: **Forward-Thinking and Counterfactual-Branching trace statistics.** Forward-thinking outcome percentages are over two-candidate traces.

Sokoban Forward Thinking	
Both solve	79%
At least one candidate deadlocks	21%
Sokoban Counterfactual Branching	
Branches per episode: 0 / 1 / 2 / 3	27.1 / 35.3 / 20.6 / 17.1%
Pacman Forward Thinking	
Both candidates succeed	96.4%
A candidate dies	3.6%



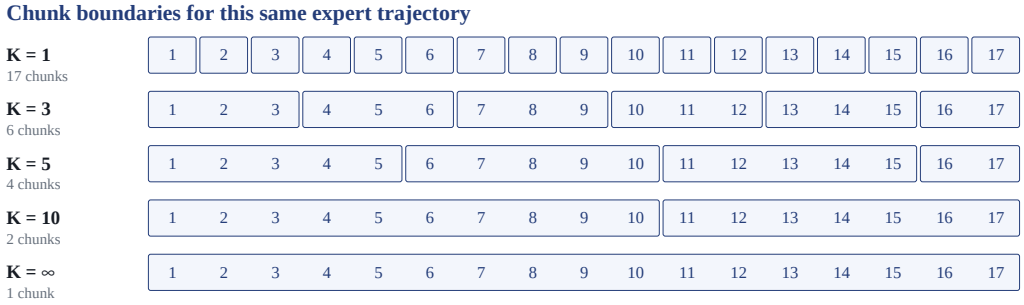


Figure 7: **A complete expert trajectory and its action chunks.** The same 17-action training trajectory, including `STOP`, is segmented with $K \in \{1, 3, 5, 10, \infty\}$. Numbers in the lower panel refer to the actions above. Images are saved environment observations used as supervision.

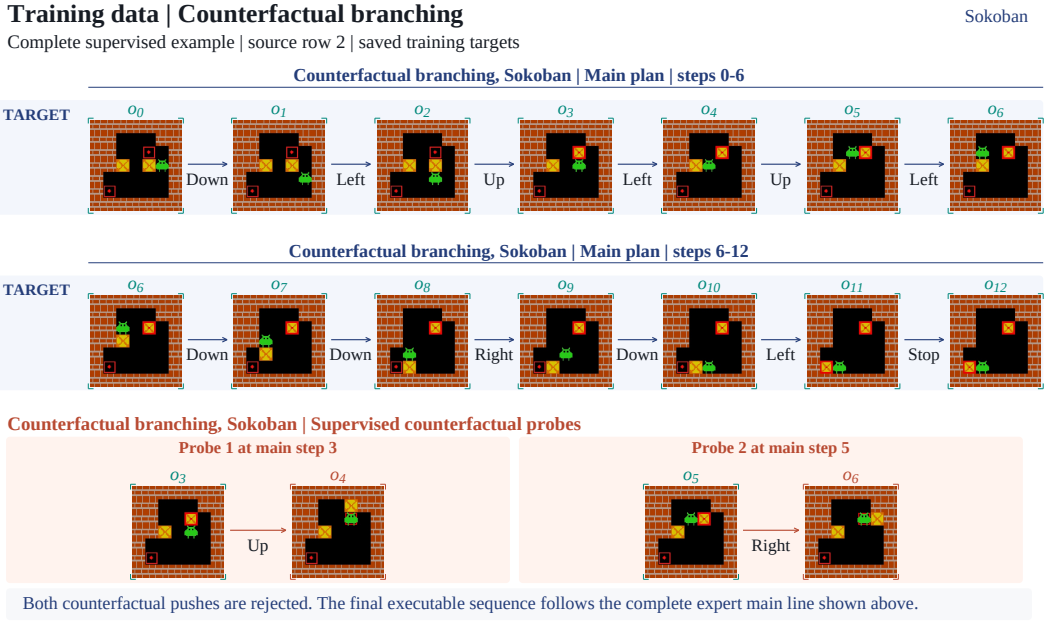
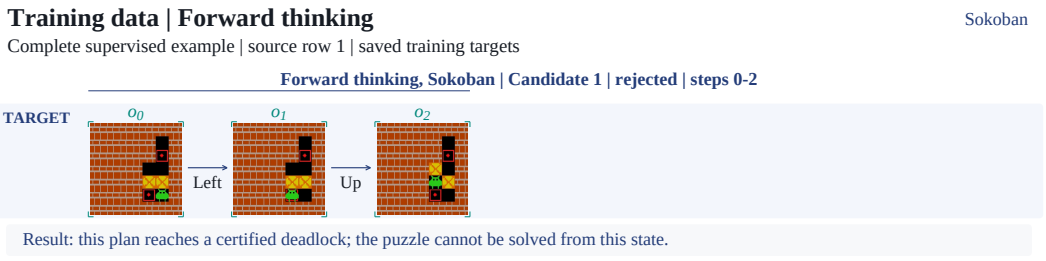


Figure 8: **Complete counterfactual-branching training example.** The main line contains 12 actions including `STOP`. The two alternative pushes, inserted after main-line actions 3 and 5, lead to certified deadlocks and are rejected.



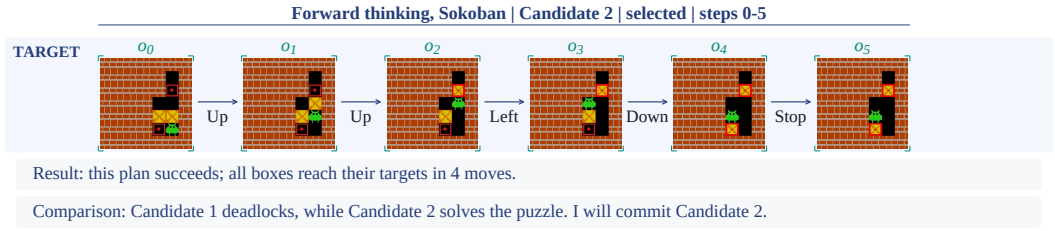


Figure 9: **Complete forward-thinking training example in Sokoban.** Candidate 1 deadlocks after two moves. Candidate 2 solves the puzzle in four moves and ends with `STOP`.



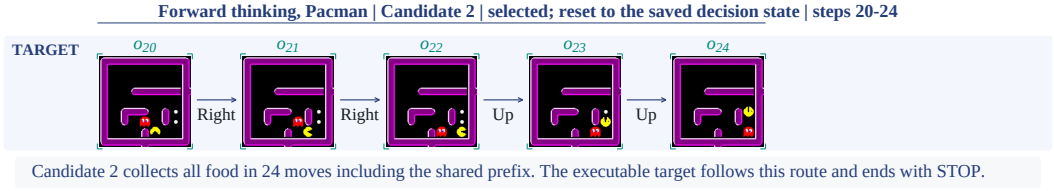


Figure 10: **Complete forward-thinking training example in Pacman.** After a shared five-move prefix, Candidate 1 is caught by the ghost after seven further moves. Candidate 2 collects all food after 19 further moves; its executable target ends with `STOP`. All saved training images are shown.

B.4 Model and Training Setup

Backbone and SFT. We use BAGEL (Deng et al., 2025), with 14B total parameters and 7B active parameters. SFT uses teacher forcing, with next-token cross-entropy for text outputs and the image-generation MSE objective described in Section 2.2.

For PushT world-model training, we use a keep-last-VAE attention mask that restricts VAE conditioning to the most recent block while retaining all preceding text and ViT tokens. At inference, we enforce the same policy by evicting the previous conditioning VAE block from the KV cache whenever a new image is added.

Table 6: **SFT settings.**

Setting	Value
Standard SFT steps	5,000
SFT batch size	8
Training-example presentations	40,000
Released Sokoban checkpoints	BAGEL-7B-MoT, exponential-moving-average (EMA) weights
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)
Learning rate	2×10^{-5}
Learning-rate schedule	Cosine
Warmup steps	100
EMA decay	0.993

Reinforcement learning. During RL, the model samples a reasoning trace containing textual decisions, imagined visual states, and a final executable action sequence. It receives no real environment observations during this deliberation. Only the final sequence is executed, and the reward is binary task success. We use GRPO (Shao et al., 2024) with group size $N = 8$, batch size 16, and 200 training steps on ordinary 8×8 three-box Sokoban puzzles, without additional expert demonstrations. The learning rate is 2×10^{-6}. The Sokoban RL recipe uses sampling temperature 1.0. Evaluation uses held-out three- and four-box Deadlock and Structural Fork sets. The policy-gradient objective uses the language-token log probabilities. Generated images are not directly supervised with an image reconstruction loss during this stage.

C Analysis of Adaptive Imagination

We examine how RL changes the plans generated by the model. All results use the four 3/4-box Deadlock and Structural Fork sets, with 100 episodes per set. Pooled results therefore contain 400 episodes at each checkpoint. Pre-RL denotes the SFT checkpoint; RL50, for example, denotes 50 RL training steps.

C.1 Combining Linear Rollouts and Forward Thinking

SFT teaches two separate formats: a linear rollout that produces one plan, and a fixed comparison that generates two candidate plans before choosing one. RL combines them into a *hybrid format*: the model begins with a linear rollout, evaluates its predicted outcome, and adds another candidate when it judges the first plan unsuccessful. Figure 5 shows this process continuing to a third candidate. The model can therefore decide to compare plans after inspecting its first imagined outcome. Related embodied agents use execution feedback to diagnose failures and revise plans. Here, imagined outcomes trigger further planning before execution. Table 7 separates the format change from the decision to generate another candidate. From RL100 onward, almost every comparison uses the hybrid format, and the model always judges the first candidate unsuccessful before adding a second. We check that judgment by executing the first candidate's actions from the original state. At RL200, these actions fail to solve the puzzle in 95.5–100% of the comparisons across the four sets.

Table 7: **Forward-Thinking formats and first-plan failures.** C1 and C2 denote the first and second candidate plans. Generates C2, Fixed comparison, and Hybrid use all 100 episodes per set as their denominator. The last two rows use comparisons with readable candidate actions and verdicts: Predicts C1 failure records the model's judgment; Actual C1 failure records whether C1 fails when executed independently. All values are percentages.

Set	Metric	Pre-RL	RL50	RL100	RL150	RL200
3-box Deadlock	Generates C2	83%	57%	18%	20%	22%
	Fixed comparison	79%	51%	0%	0%	0%
	Hybrid	4%	6%	18%	20%	22%
	Predicts C1 failure	31.7%	33.9%	100%	100%	100%
	Actual C1 failure	61.0%	64.3%	94.4%	100%	95.5%
4-box Deadlock	Generates C2	76%	56%	34%	36%	37%
	Fixed comparison	72%	42%	0%	0%	0%
	Hybrid	4%	14%	34%	36%	37%
	Predicts C1 failure	55.4%	63.0%	100%	100%	100%
	Actual C1 failure	81.1%	83.3%	97.1%	97.2%	97.3%
3-box Structural Fork	Generates C2	80%	59%	20%	29%	30%
	Fixed comparison	75%	47%	0%	0%	0%
	Hybrid	5%	12%	20%	29%	30%
	Predicts C1 failure	53.5%	61.1%	100%	100%	100%
	Actual C1 failure	73.2%	72.2%	100%	96.6%	96.7%
4-box Structural Fork	Generates C2	77%	60%	31%	34%	34%
	Fixed comparison	70%	42%	1%	0%	0%
	Hybrid	7%	18%	30%	34%	34%
	Predicts C1 failure	57.6%	60.0%	100%	100%	100%
	Actual C1 failure	84.8%	83.6%	100%	100%	100%

C.2 Counterfactual Branching

RL mainly changes how many alternatives the model examines within a plan. Across the four sets, the percentage of episodes with any branch changes from 81.8% before RL to 83.5% at RL200. The mean number of branches per episode increases from 1.70 to 1.89, reaching 1.95 at RL150.

Table 8 shows the results for each set. On 3-box Structural Fork, the percentage of episodes with at least three branches rises from 41% before RL to 56% at RL200; on 4-box Structural Fork, it

rises from 35% to 54%. Each alternative branch still contains one imagined action. The critical-step analysis in Figure 6 measures whether these branches occur at decisions where a wrong action could make the puzzle unsolvable.

Table 8: **Number of counterfactual branches per episode.** Mean counts branches across all 100 episodes in each row, including episodes with no branch. Columns 0, 1, 2, and ≥ 3 give the percentage of episodes with that many complete branches. Each branch contains one imagined action.

Set	Checkpoint	Mean	0	1	2	≥ 3
3-box Deadlock	Pre-RL	1.18	34%	31%	18%	17%
	RL50	1.13	33%	36%	16%	15%
	RL100	1.20	34%	30%	18%	18%
	RL150	1.25	35%	25%	20%	20%
	RL200	1.25	35%	25%	20%	20%
4-box Deadlock	Pre-RL	1.67	22%	23%	21%	34%
	RL50	1.67	23%	20%	24%	33%
	RL100	1.73	17%	29%	18%	36%
	RL150	1.88	13%	24%	25%	38%
	RL200	1.76	19%	22%	23%	36%
3-box Structural Fork	Pre-RL	2.08	7%	19%	33%	41%
	RL50	2.14	5%	23%	25%	47%
	RL100	2.23	4%	18%	29%	49%
	RL150	2.43	5%	11%	20%	64%
	RL200	2.30	6%	14%	24%	56%
4-box Structural Fork	Pre-RL	1.88	10%	27%	28%	35%
	RL50	2.00	6%	26%	30%	38%
	RL100	2.04	7%	22%	31%	40%
	RL150	2.25	5%	19%	22%	54%
	RL200	2.24	6%	18%	22%	54%

C.3 Task Success and Rollout Length

Table 9 reports task success and the mean numbers of imagined and executed actions. From Pre-RL to RL200, forward thinking reduces imagined actions from 27.82 to 22.18 per episode, while task success rises from 30.3% to 41.0%. Branching reduces imagined actions from 23.62 to 21.59, while task success rises from 30.0% to 50.3%. Thus, both models improve without generating longer imagined trajectories. These are action counts; we do not measure inference time or FLOPs here.

Table 9: **Mean rollout lengths and pooled task success on the four 3/4-box sets.** Each row pools 400 episodes. Task success is a percentage; Imagined and Executed are mean action counts per episode.

Format	Checkpoint	Success	Imagined	Executed
Forward	Pre-RL	30.3%	27.82	17.88
	RL50	35.0%	24.95	17.36
	RL100	37.5%	20.87	16.96
	RL150	38.5%	21.40	16.97
	RL200	41.0%	22.18	17.50
Branching	Pre-RL	30.0%	23.62	18.63
	RL50	32.0%	22.60	18.15
	RL100	40.5%	21.88	17.95
	RL150	45.8%	23.16	18.87
	RL200	50.3%	21.59	18.24

D Qualitative Dynamics Predictions and Rollouts

The following examples compare predicted visual states with ground-truth states along action sequences in each environment. These selected trajectories provide qualitative evidence; they do not report aggregate prediction accuracy. The examples show complete recorded trajectories, from the initial observation to episode termination. Every action and saved state is displayed; adjacent rows or pages repeat their shared endpoint. Examples at different chunk sizes may start from different initial states. Additionally, we show example traces of counterfactual branching and forward thinking.

D.1 Maze2D Rollouts

Task configurations are described in Appendix A.

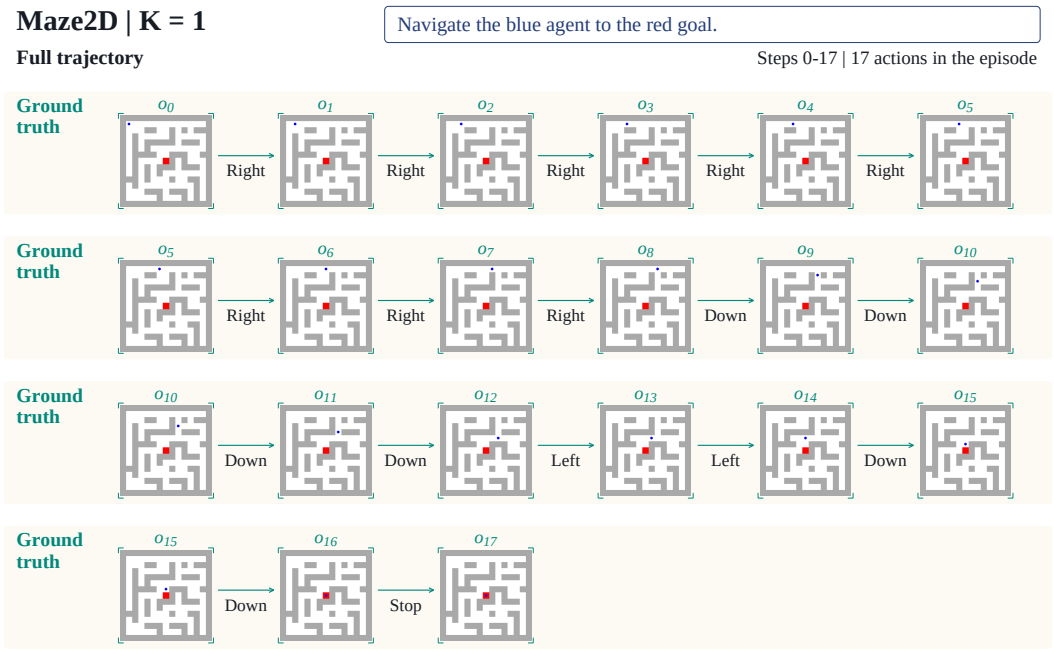


Figure 11: **Complete Maze2D trajectory at $K = 1$.** Selected successful Ordinary Hard episode; all 17 actions and their states are shown. The reactive reference receives a real observation after every action.

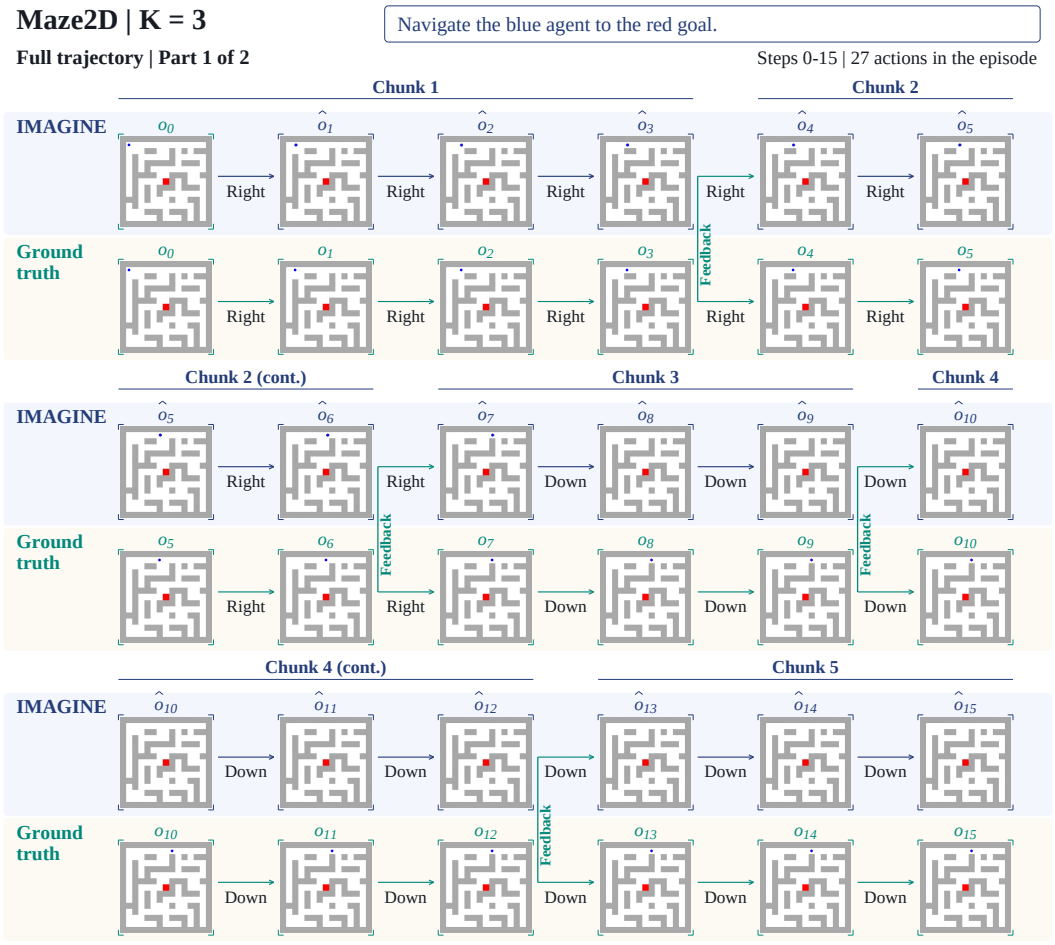


Figure 12: **Complete Maze2D trajectory at $K = 3$.** Selected successful Ordinary Hard episode; this page shows steps 0–15 of 27. The trajectory continues on the next page. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.

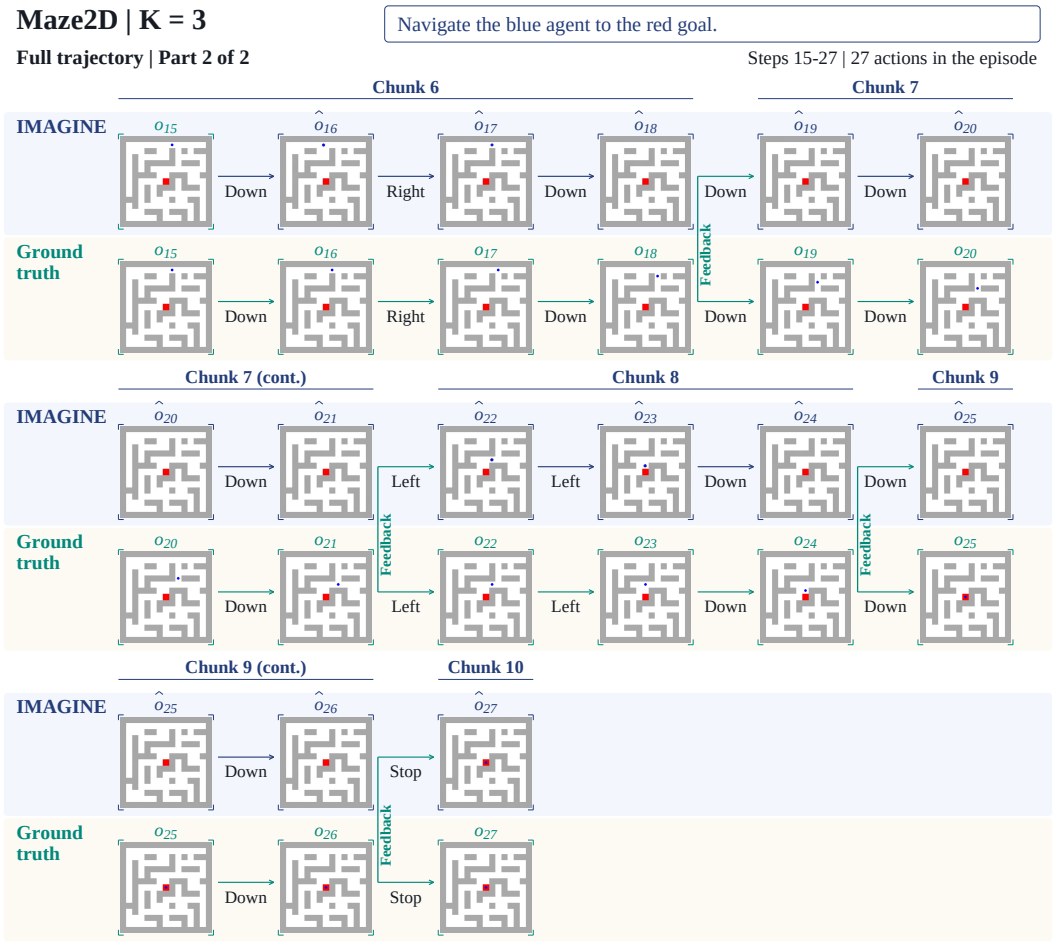


Figure 12 (continued). **Maze2D,** $K = 3$**.** Steps 15–27, ending in success. The shared endpoint connects the two pages.

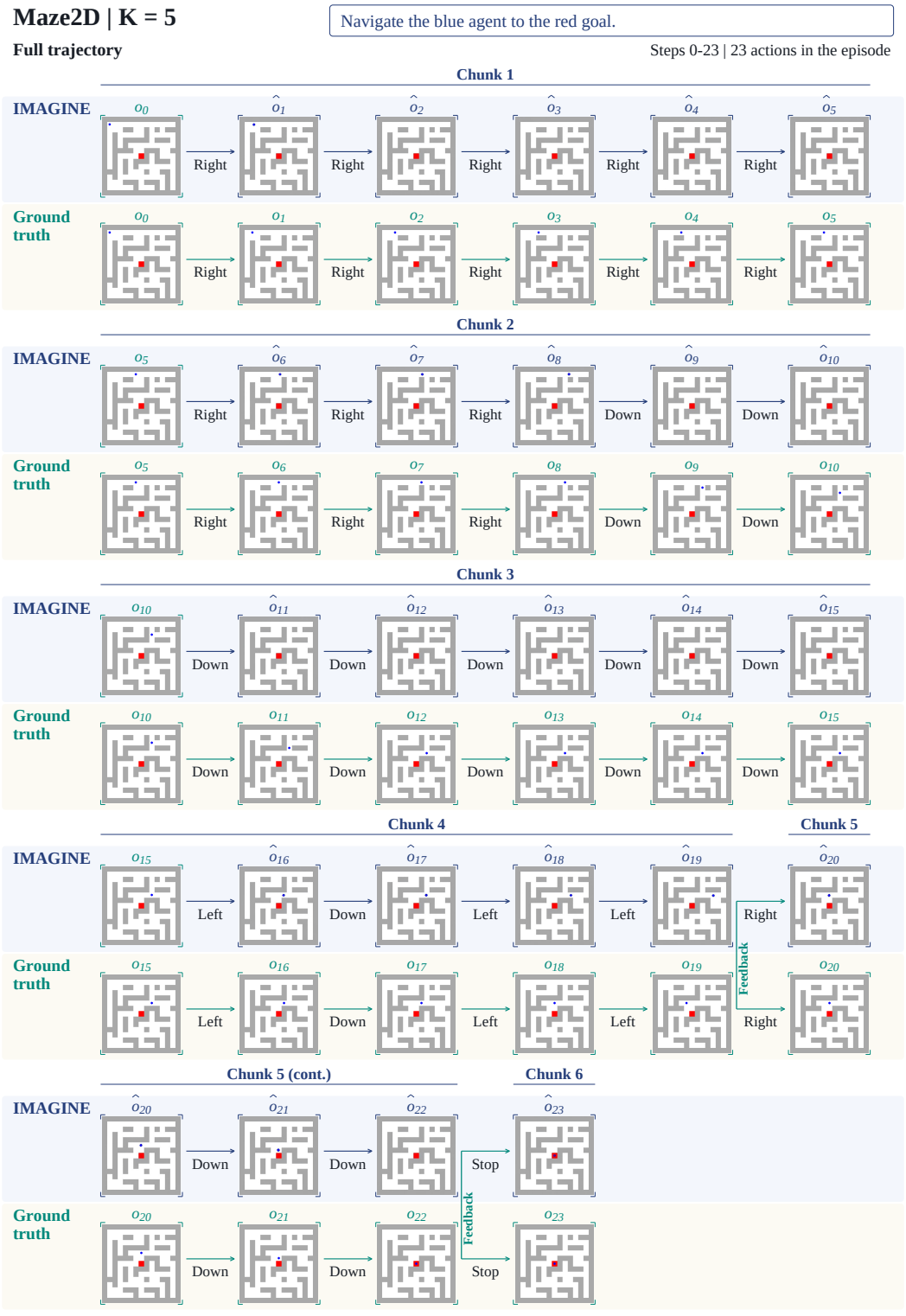


Figure 13: **Complete Maze2D trajectory at $K = 5$.** Selected successful Ordinary Hard episode; all 23 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.

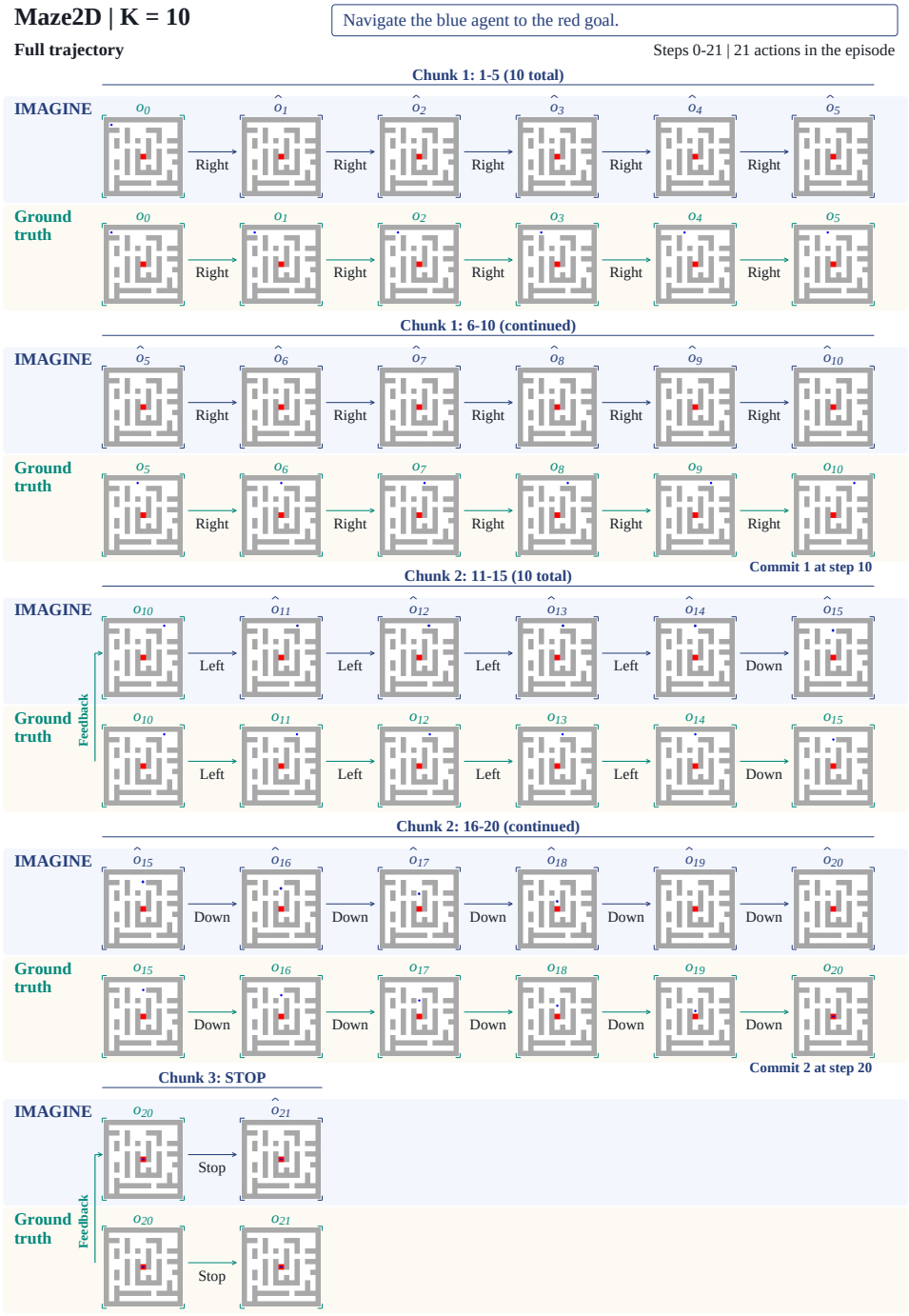


Figure 14: **Complete Maze2D trajectory at $K = 10$.** Selected successful Ordinary Hard episode; all 21 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.

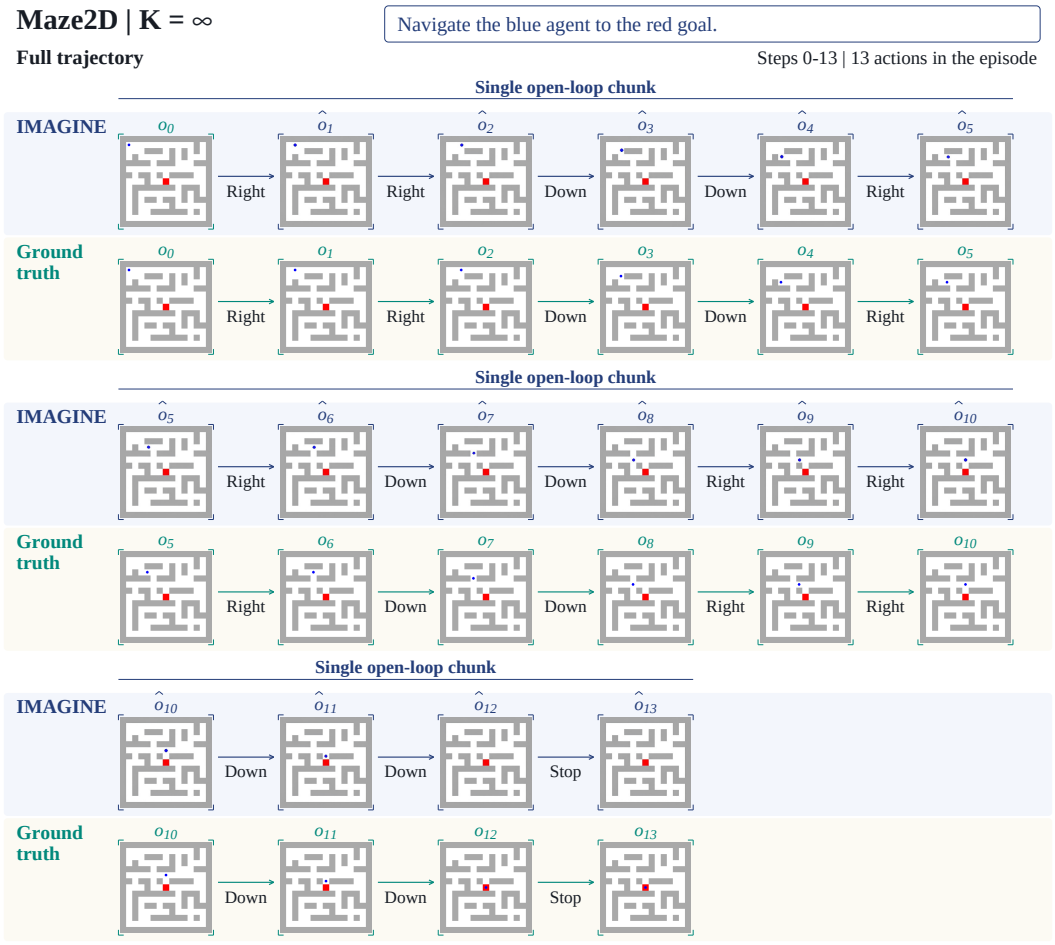


Figure 15: **Complete Maze2D trajectory at $K = \infty$.** Selected successful Ordinary Hard episode; all 13 actions and their states are shown. The model receives only the initial observation, with no intermediate feedback. Ground-truth states are shown for comparison.

D.2 SOKOBAN ROLLOUTS

Task configurations are described in Appendix A.

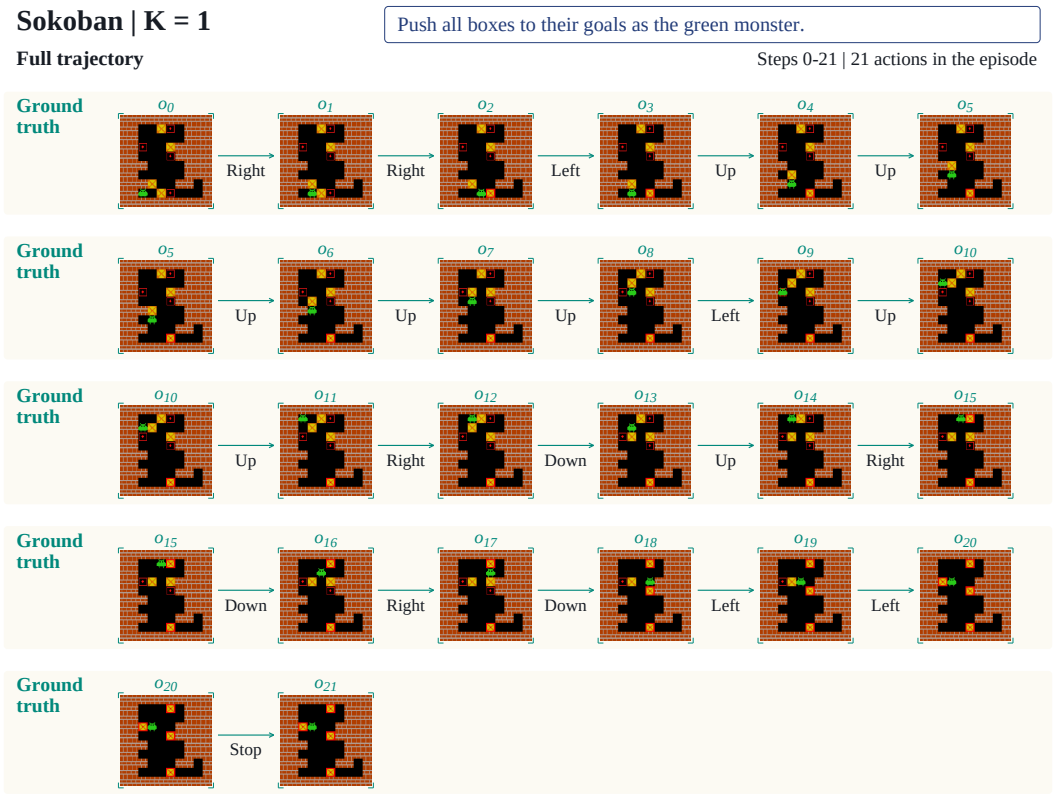


Figure 16: **Complete Sokoban trajectory at $K = 1$.** Selected successful Ordinary Hard episode; all 21 actions and their states are shown. The reactive reference receives a real observation after every action.



Figure 17: **Complete Sokoban trajectory at $K = 3$.** Selected successful Ordinary Hard episode; all 21 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.



Figure 18: **Complete Sokoban trajectory at $K = 5$.** Selected successful Ordinary Hard episode; all 19 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.



Figure 19: **Complete Sokoban trajectory at $K = 10$.** Selected successful Ordinary Hard episode; all 19 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.



Figure 20: **Complete Sokoban trajectory at $K = \infty$.** Selected successful Ordinary Hard episode; all 17 actions and their states are shown. The model receives only the initial observation, with no intermediate feedback. Ground-truth states are shown for comparison.

D.3 PACMAN ROLLOUTS

Task configurations are described in Appendix A.



Figure 21: **Complete Pacman trajectory at $K = 1$.** Selected successful Ordinary Hard episode; all 33 actions and their states are shown. The reactive reference receives a real observation after every action.

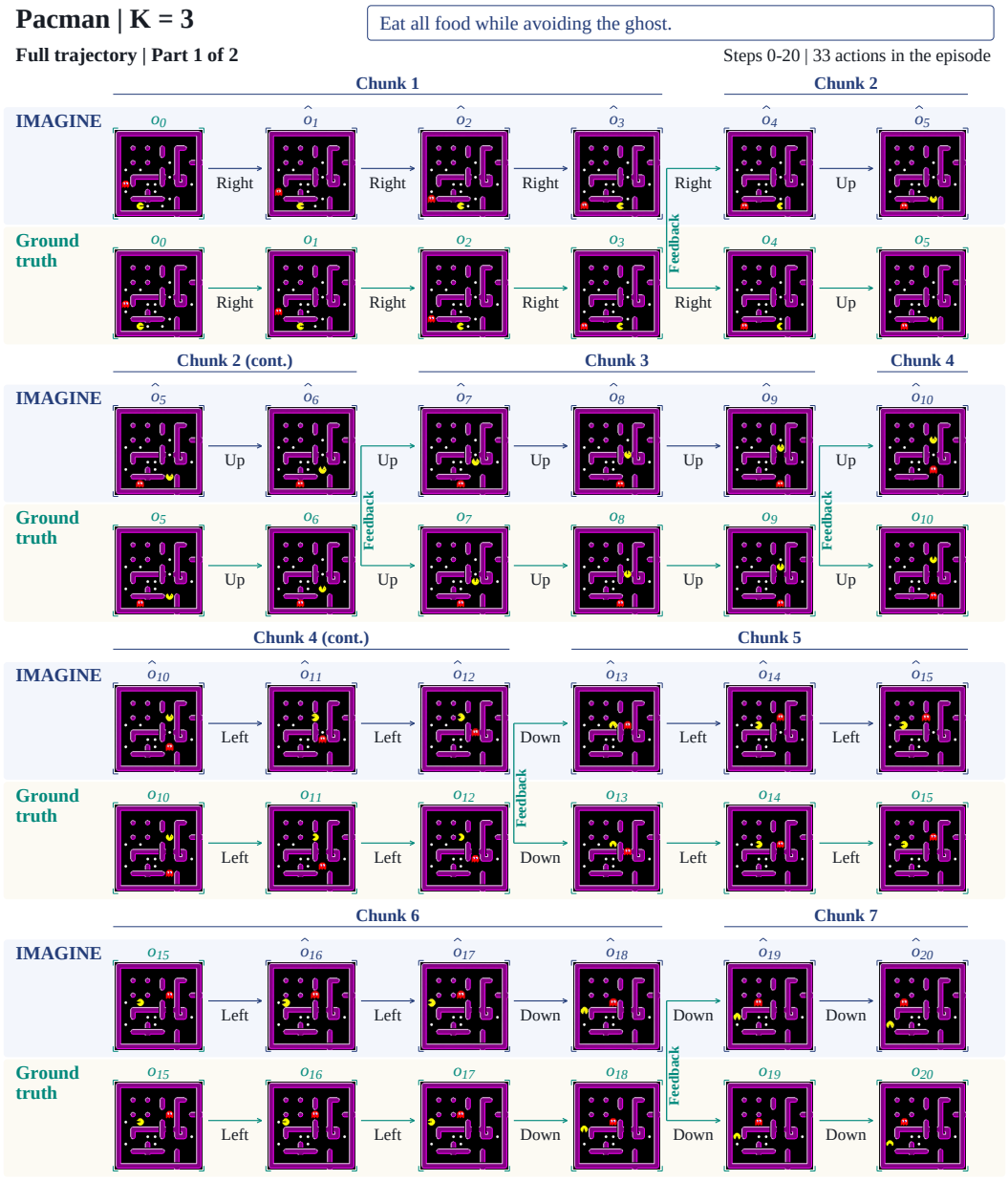


Figure 22: **Complete Pacman trajectory at $K = 3$.** Selected successful Ordinary Hard episode; this page shows steps 0–20 of 33. The trajectory continues on the next page. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.

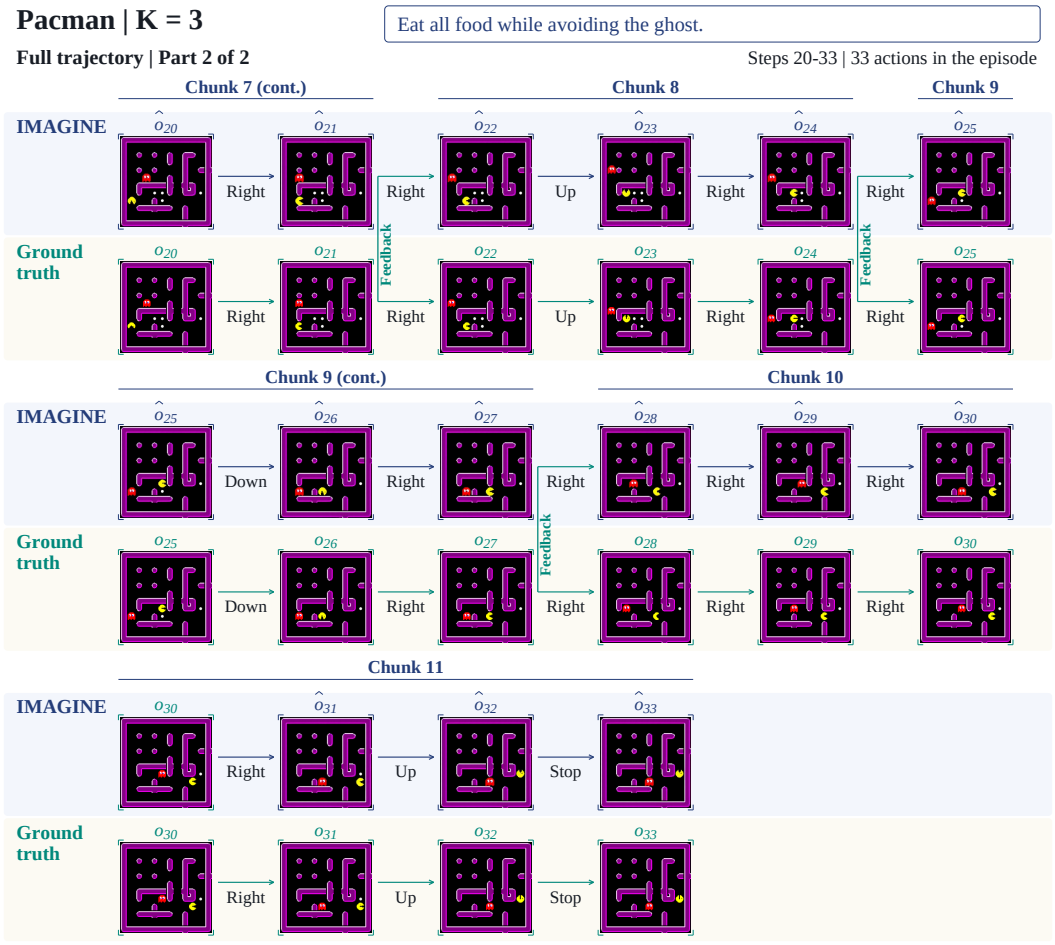


Figure 22 (continued). **Pacman,** $K = 3$**.** Steps 20–33, ending in success. The shared endpoint connects the two pages.

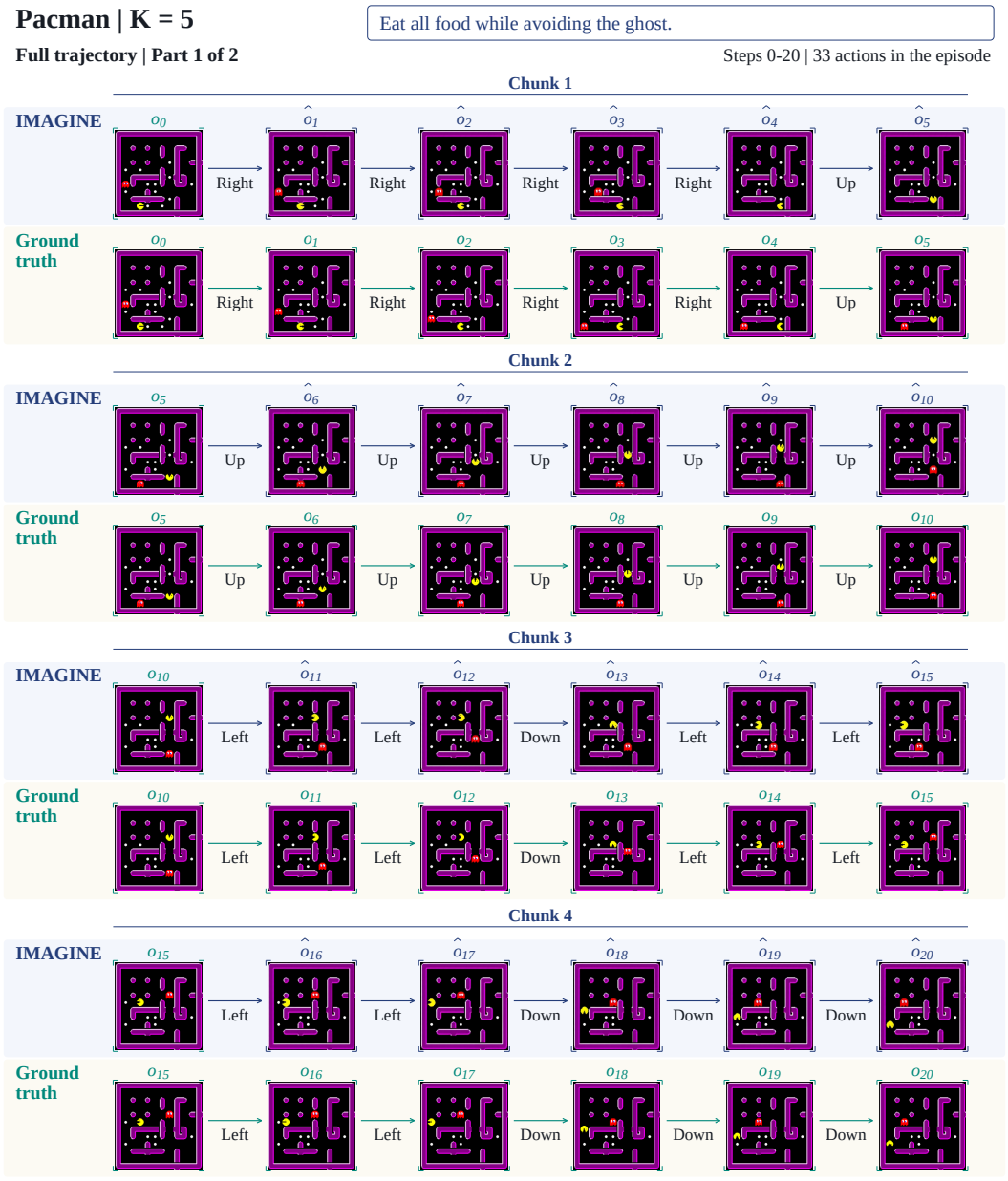


Figure 23: **Complete Pacman trajectory at $K = 5$.** Selected successful Ordinary Hard episode; this page shows steps 0–20 of 33. The trajectory continues on the next page. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.



Figure 23 (continued). **Pacman,** $K = 5$**.** Steps 20–33, ending in success. The shared endpoint connects the two pages.

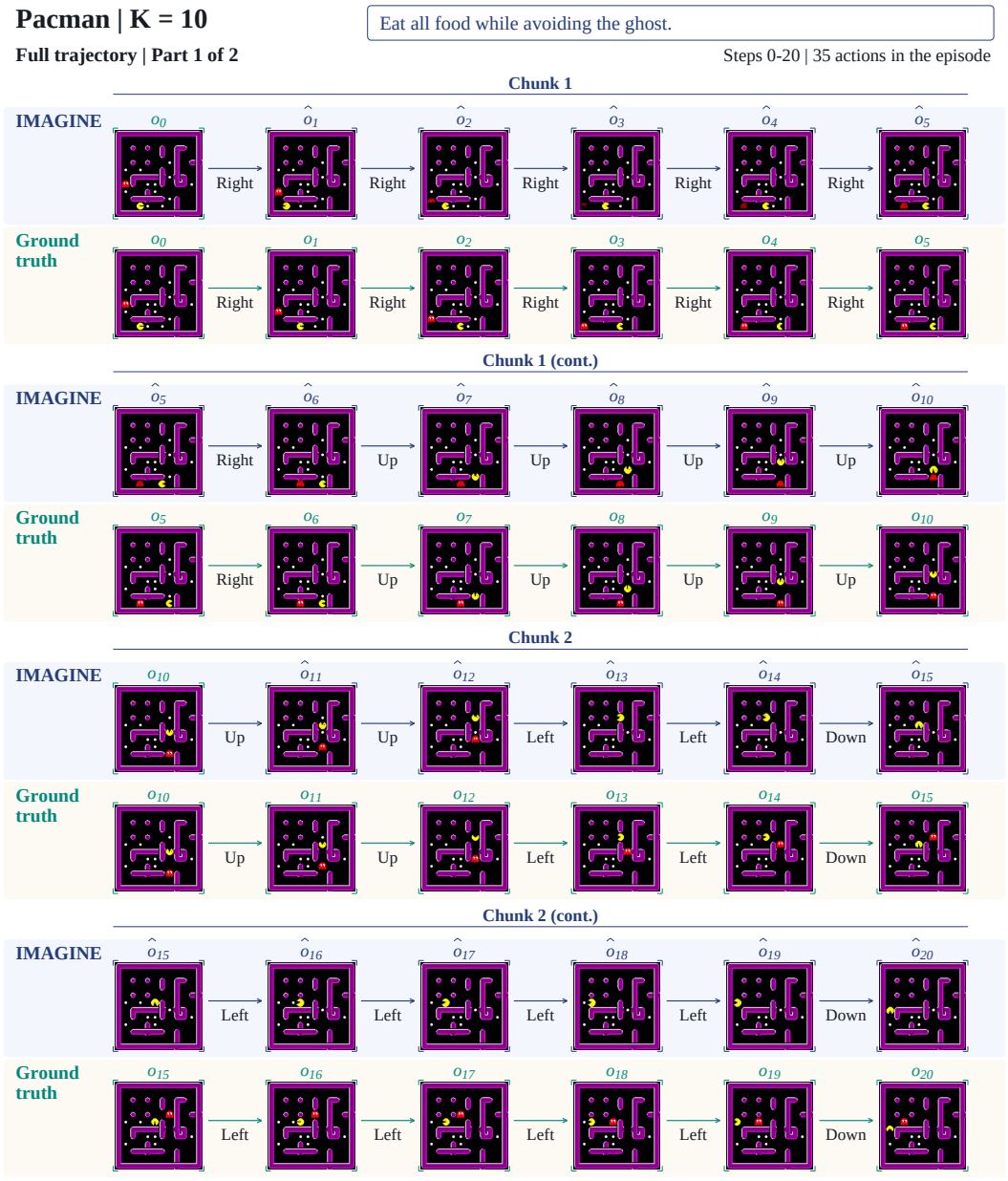


Figure 24: **Complete Pacman trajectory at $K = 10$.** Selected successful Ordinary Hard episode; this page shows steps 0–20 of 35. The trajectory continues on the next page. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks.

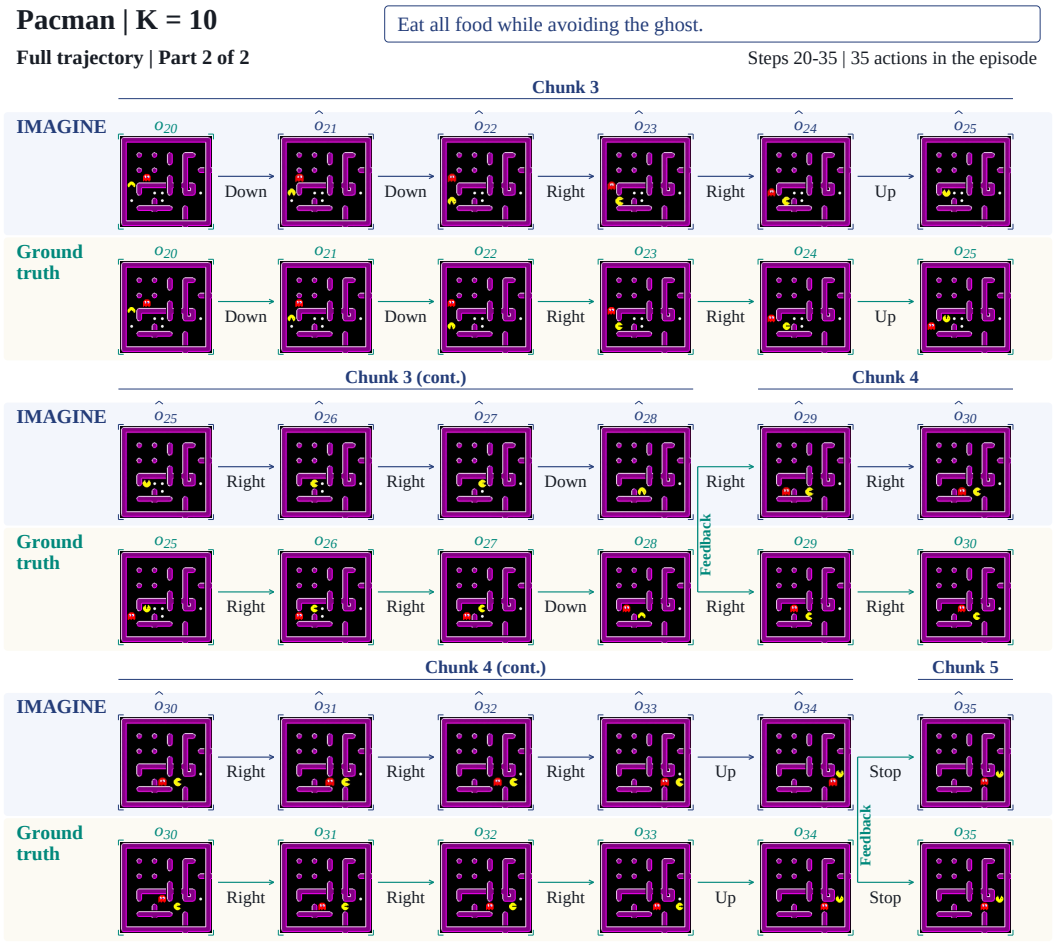


Figure 24 (continued). **Pacman,** $K = 10$**.** Steps 20–35, ending in success. The shared endpoint connects the two pages.

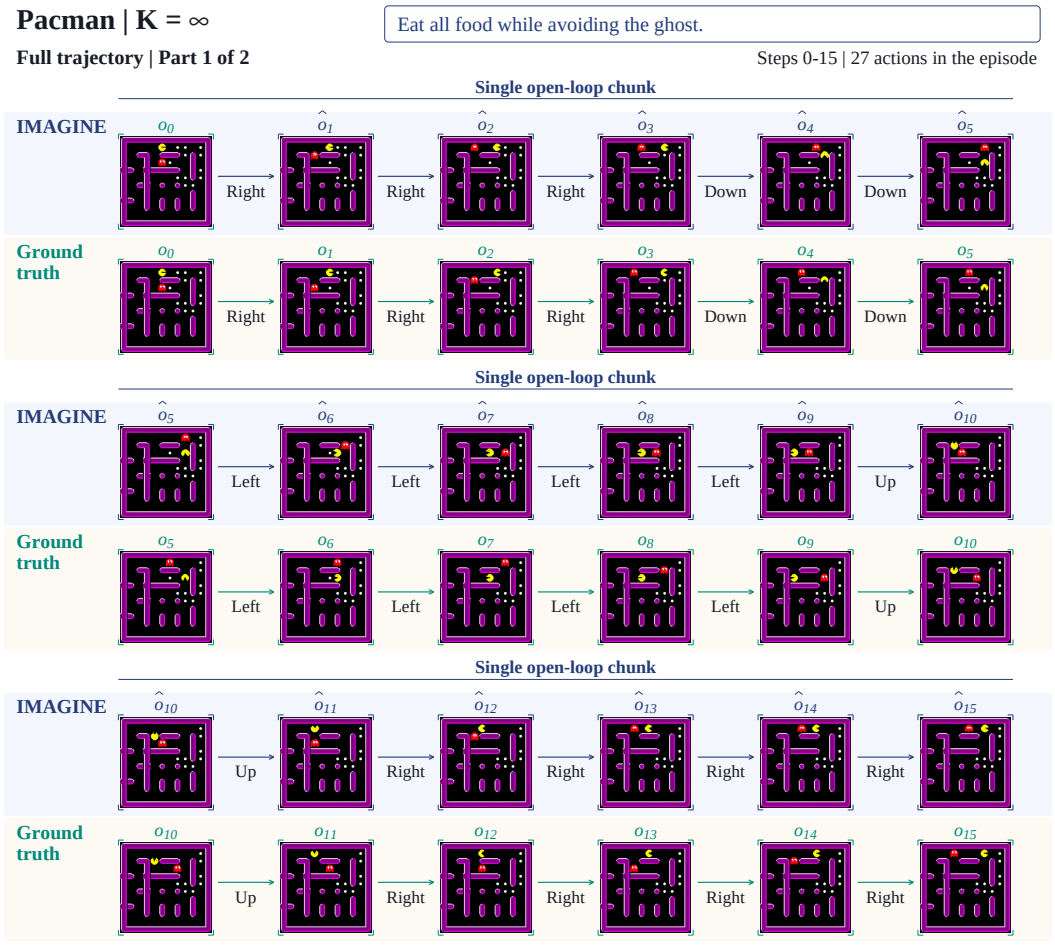


Figure 25: **Complete Pacman trajectory at $K = \infty$.** Selected successful Ordinary Hard episode; this page shows steps 0–15 of 27. The trajectory continues on the next page. The model receives only the initial observation, with no intermediate feedback. Ground-truth states are shown for comparison.



Figure 25 (continued). **Pacman,** $K = \infty$**.** Steps 15–27, ending in success. The shared endpoint connects the two pages; no new observation is supplied.

D.4 PUSHT ROLLOUTS

The starting-position shift is described in Appendix A.

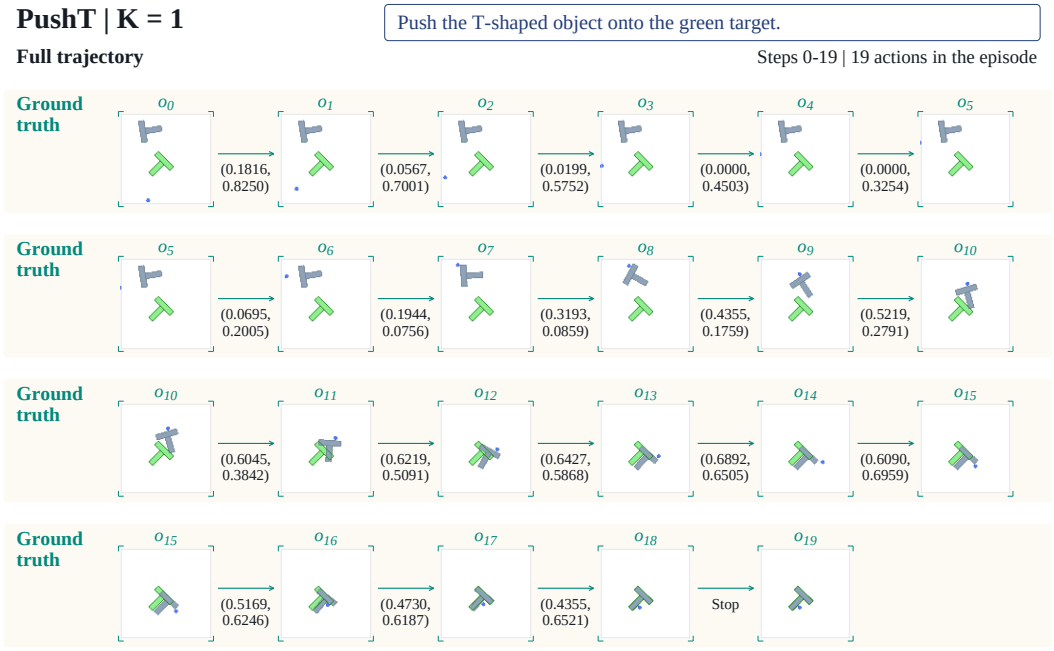


Figure 26: **Complete PushT trajectory at $K = 1$.** Selected successful Far episode; all 19 actions and their states are shown. The reactive reference receives a real observation after every action. Actions give normalized target coordinates (x, y).



Figure 27: **Complete PushT trajectory at $K = 3$.** Selected successful Far episode; all 18 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks. Actions give normalized target coordinates (x, y).



Figure 28: **Complete PushT trajectory at $K = 5$.** Selected successful Far episode; this page shows steps 0–15 of 27. The trajectory continues on the next page. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks. Actions give normalized target coordinates (x, y).



Figure 28 (continued). **PushT,** $K = 5$**.** Steps 15–27, ending in success. The shared endpoint connects the two pages.

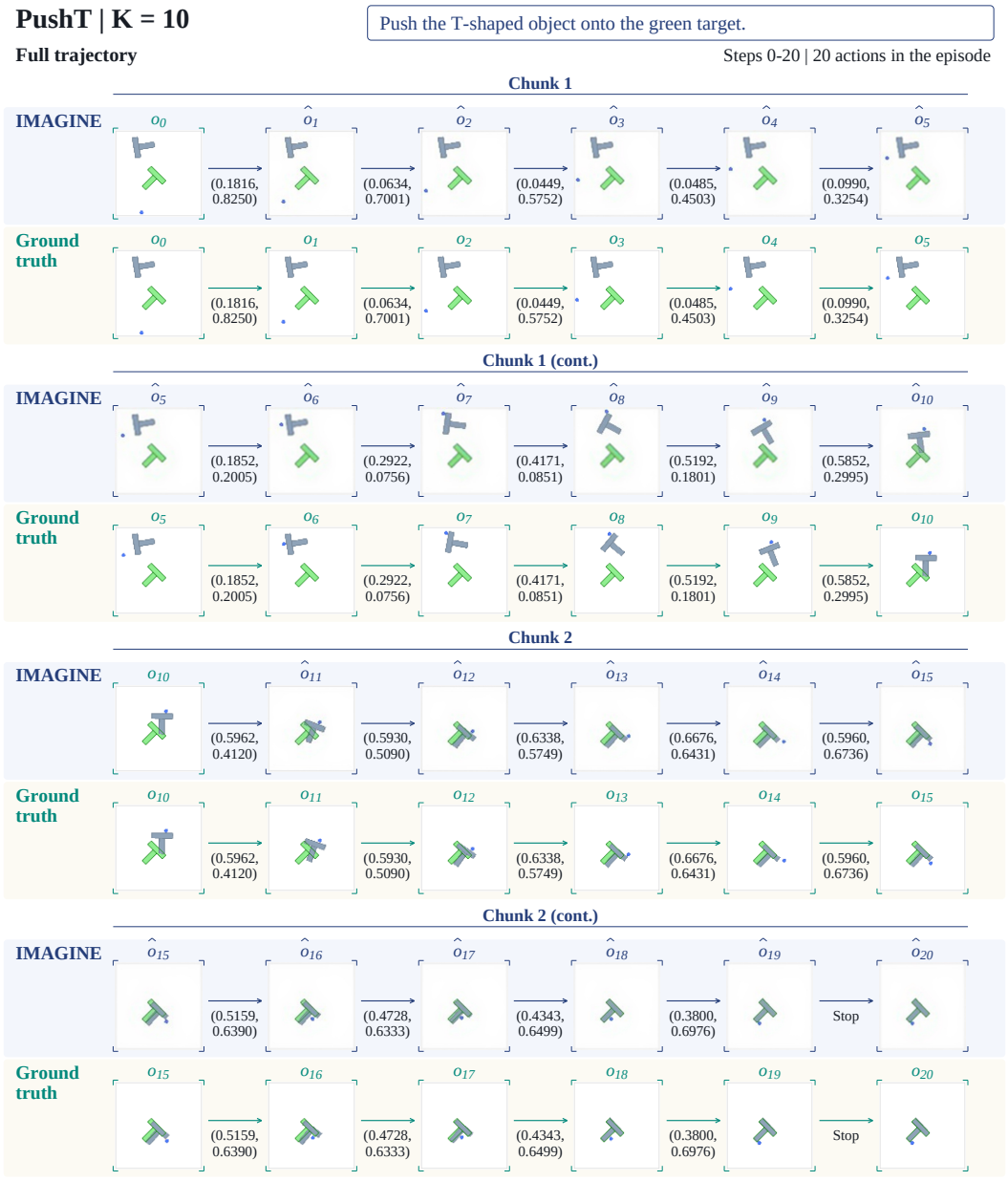


Figure 29: **Complete PushT trajectory at $K = 10$.** Selected successful Far episode; all 20 actions and their states are shown. Blue rows show predictions; teal inputs and feedback arrows show the real observations supplied between chunks. Actions give normalized target coordinates (x, y).



Figure 30: **Complete PushT trajectory at $K = \infty$.** Selected successful Far episode; this page shows steps 0–15 of 28. The trajectory continues on the next page. The model receives only the initial observation, with no intermediate feedback. Ground-truth states are shown for comparison. Actions give normalized target coordinates (x, y).



Figure 30 (continued). **PushT,** $K = \infty$**.** Steps 15–28, ending in success. The shared endpoint connects the two pages; no new observation is supplied.

D.5 ROLLOUTS FOR DIFFERENT REASONING FORMATS

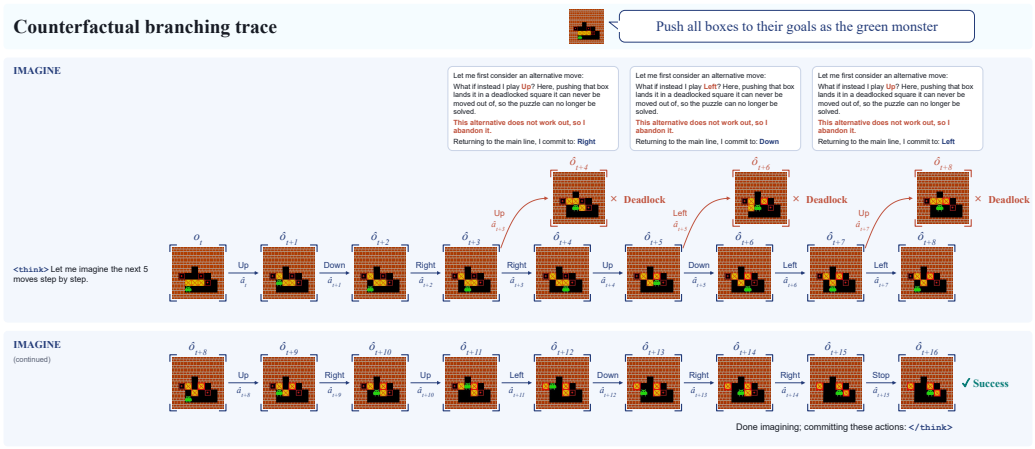


Figure 31: **Counterfactual Branching.** Example trace of counterfactual branching in Sokoban.



Figure 32: **Forward Thinking.** Example trace of Forward thinking in Sokoban.

D.6 RL Trajectories across Checkpoints

We show complete trajectories at RL50, RL100, RL150, and RL200 on the three- and four-box Deadlock and Structural Fork sets. Each group uses one fixed initial state across all four checkpoints. The Deadlock examples use Agima (Forward), and the Structural Fork examples use Agima (Branching).

Blue rows show model-generated states. Ground-truth rows show execution of the committed plan; unexecuted candidates have no ground-truth row. Orange panels show the alternatives rejected by the model and identify where each branches from the main plan. Quoted assessments are the model's own text. The recorded execution result is reported separately. All saved generated frames, executed states, and actions are shown.

Table 10: **Outcomes of the illustrated RL episodes.** Each row follows one selected initial state across four checkpoints. Aggregate performance is reported in Figure 4.

Test set	Model	RL50	RL100	RL150	RL200
3-box Deadlock	Forward	Fail	Success	Success	Success
4-box Deadlock	Forward	Fail	Fail	Success	Success
3-box Structural Fork	Branching	Fail	Success	Success	Success
4-box Structural Fork	Branching	Fail	Fail	Success	Success

RL50 | Forward 3-box Deadlock

Full trace | 21 generated frames | 10 executed actions | episode 84



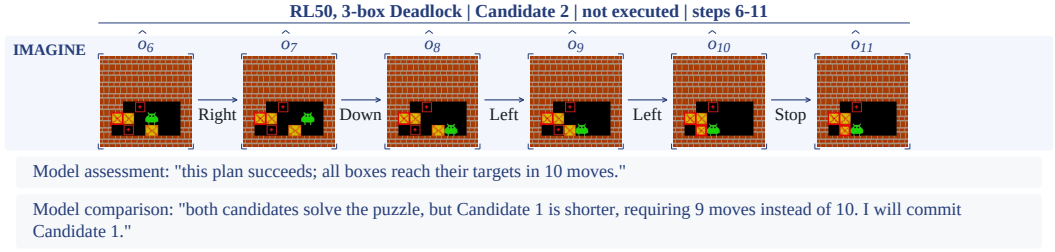


Figure 33: **Forward at RL50 on three-box Deadlock.** Complete trace for episode 84; recorded execution ends in failure.

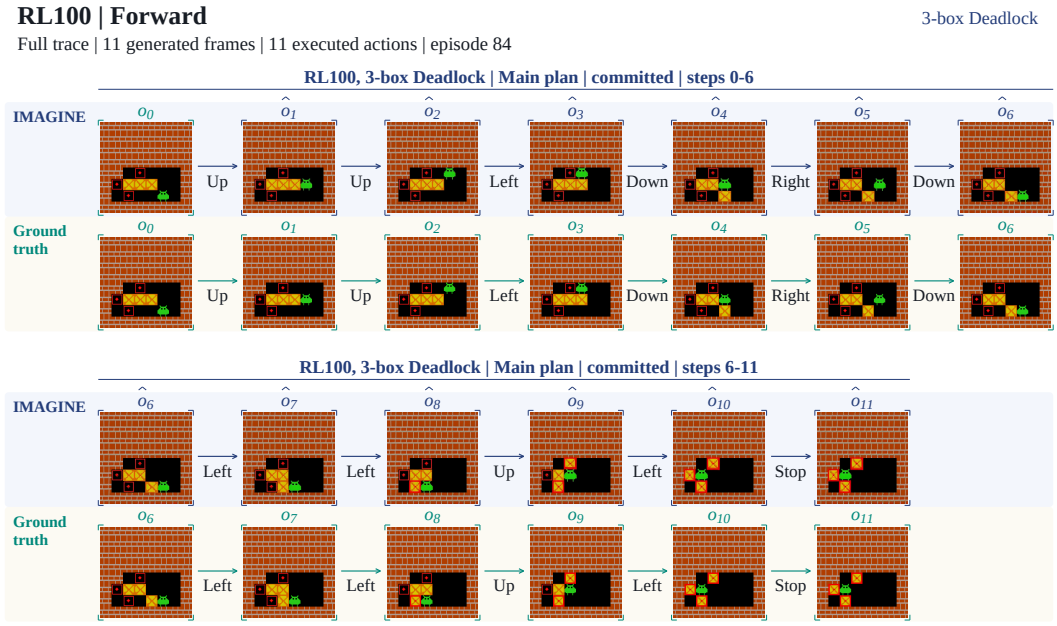
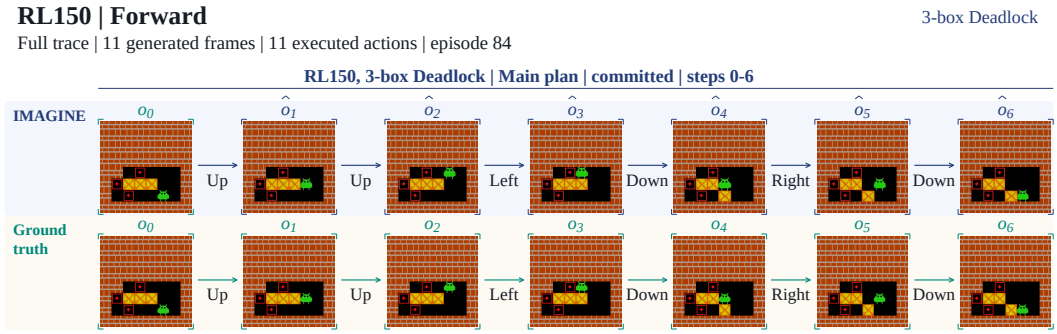


Figure 34: **Forward at RL100 on three-box Deadlock.** Complete trace for episode 84; recorded execution ends in success.



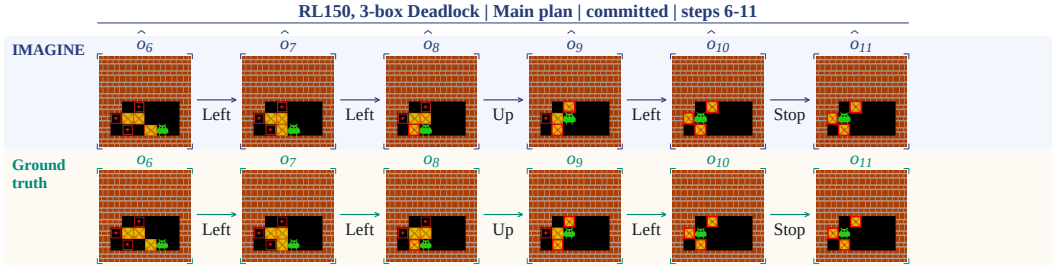


Figure 35: **Forward at RL150 on three-box Deadlock.** Complete trace for episode 84; recorded execution ends in success.

RL200 | Forward 3-box Deadlock

Full trace | 11 generated frames | 11 executed actions | episode 84

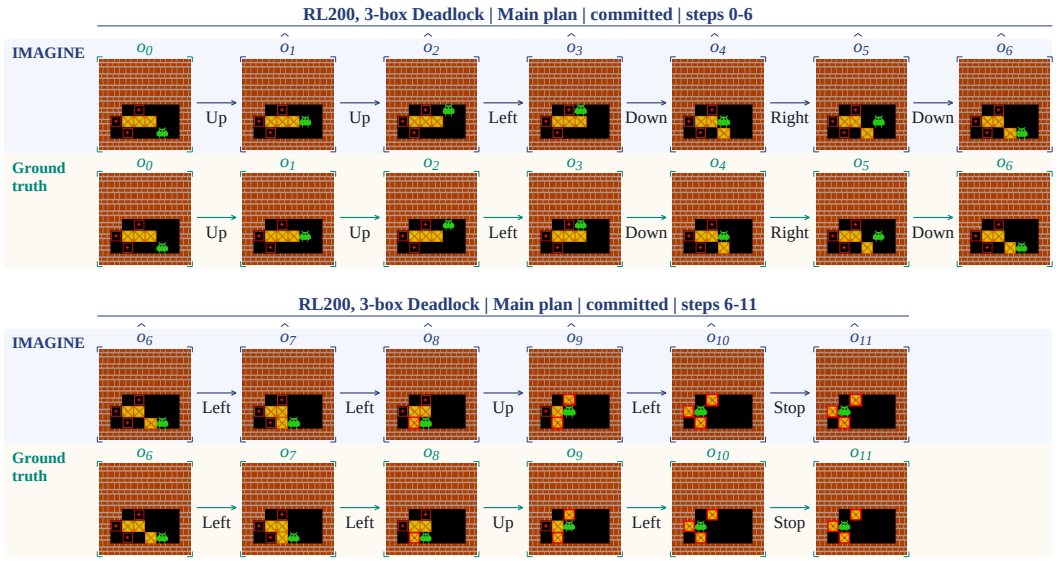


Figure 36: **Forward at RL200 on three-box Deadlock.** Complete trace for episode 84; recorded execution ends in success.

RL50 | Forward 4-box Deadlock

Full trace | 34 generated frames | 14 executed actions | episode 21

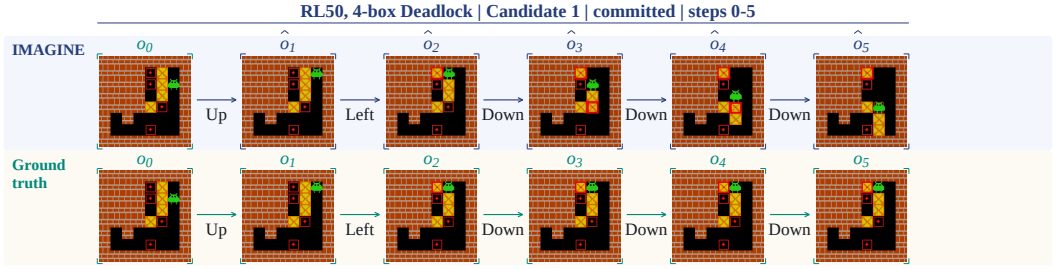




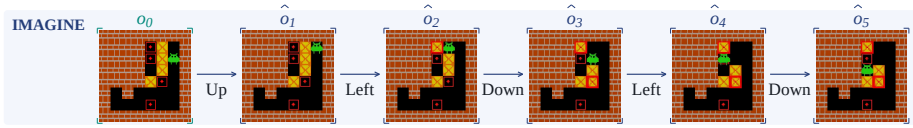
Figure 37: **Forward at RL50 on four-box Deadlock.** Complete trace for episode 21; recorded execution ends in failure.

RL100 | Forward

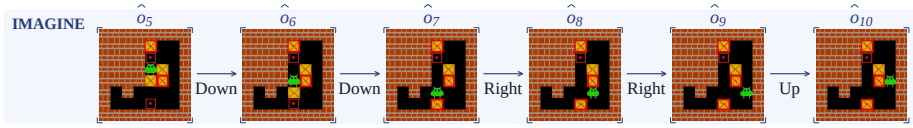
4-box Deadlock

Full trace | 34 generated frames | 14 executed actions | episode 21

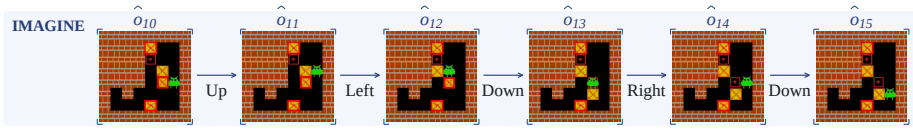
RL100, 4-box Deadlock | Main plan | not executed | steps 0-5



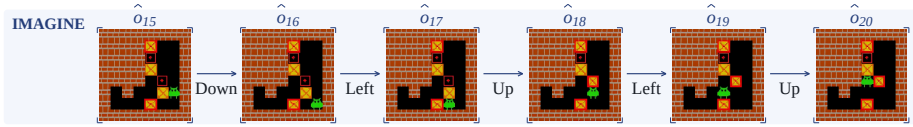
RL100, 4-box Deadlock | Main plan | not executed | steps 5-10



RL100, 4-box Deadlock | Main plan | not executed | steps 10-15

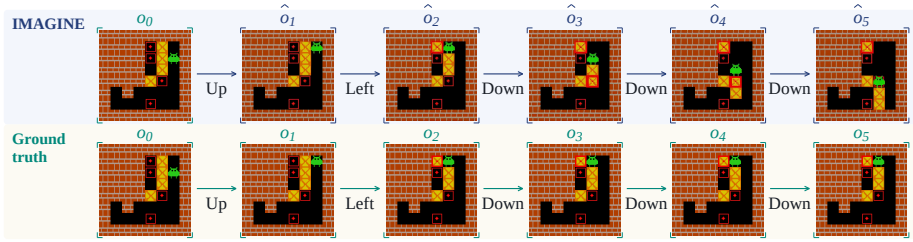


RL100, 4-box Deadlock | Main plan | not executed | steps 15-20

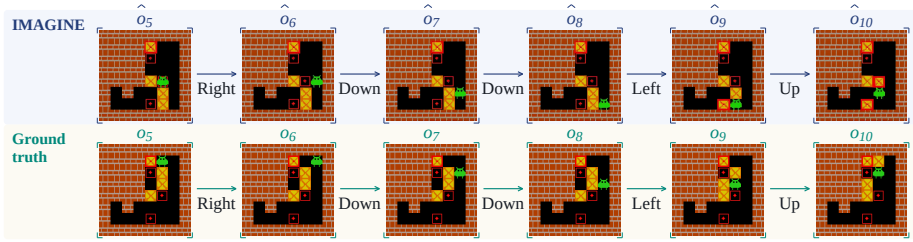


Model assessment: "this plan has not solved the puzzle after 20 moves. I stop evaluating this candidate at the task horizon."

RL100, 4-box Deadlock | Candidate 2 | committed | steps 0-5



RL100, 4-box Deadlock | Candidate 2 | committed | steps 5-10



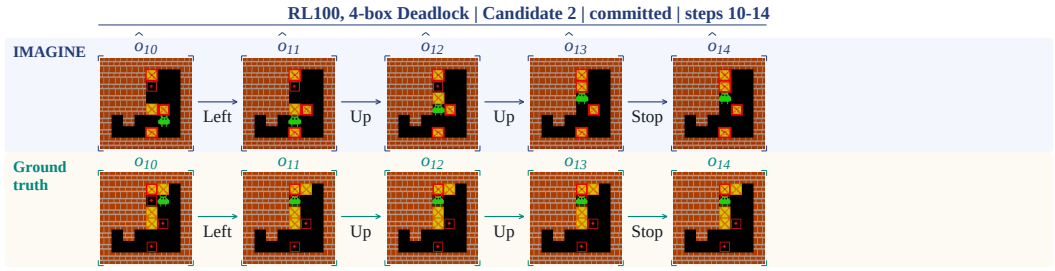


Figure 38: **Forward at RL100 on four-box Deadlock.** Complete trace for episode 21; recorded execution ends in failure.

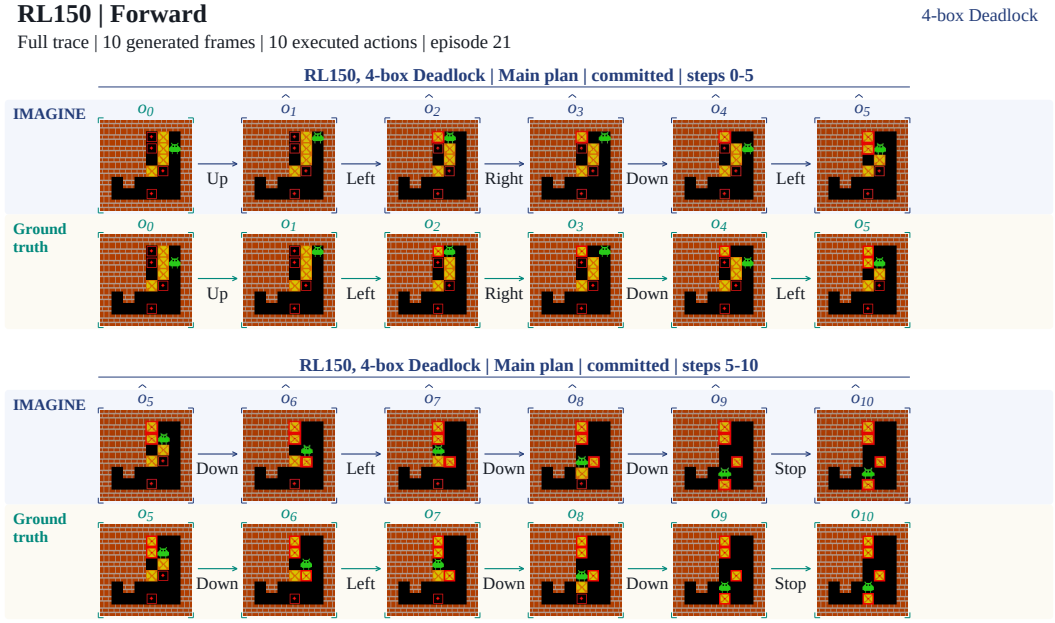
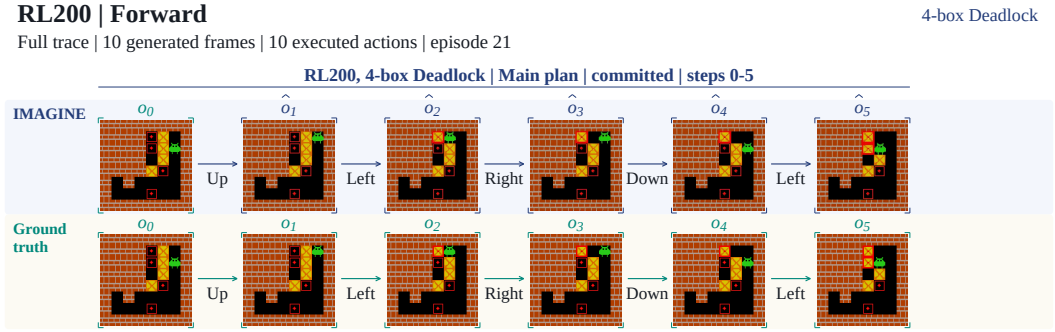


Figure 39: **Forward at RL150 on four-box Deadlock.** Complete trace for episode 21; recorded execution ends in success.



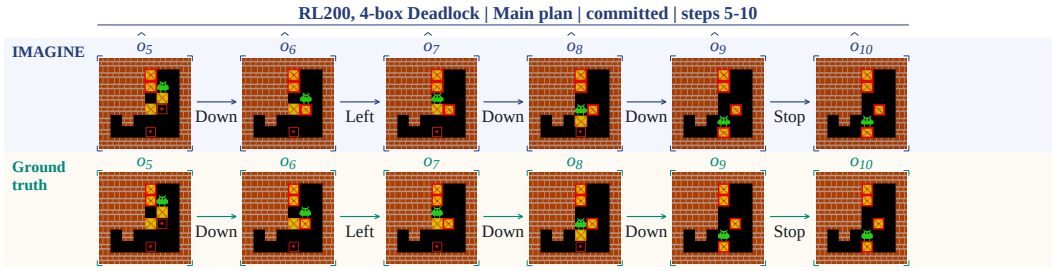


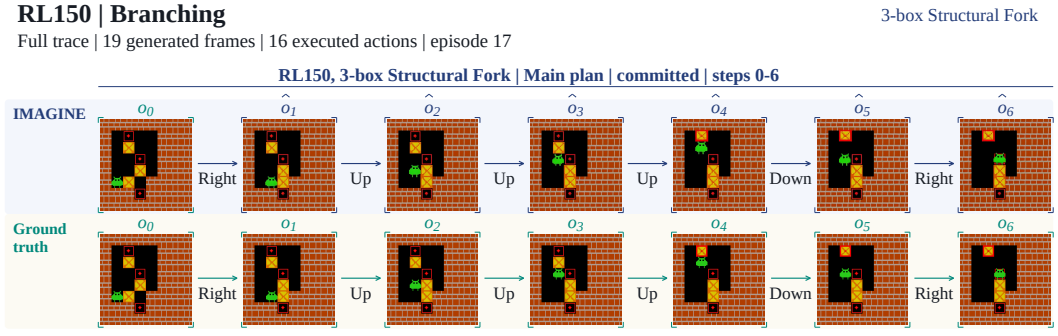
Figure 40: **Forward at RL200 on four-box Deadlock.** Complete trace for episode 21; recorded execution ends in success.



Figure 41: **Branching at RL50 on three-box Structural Fork.** Complete trace for episode 17; recorded execution ends in failure.



Figure 42: **Branching at RL100 on three-box Structural Fork.** Complete trace for episode 17; recorded execution ends in success.



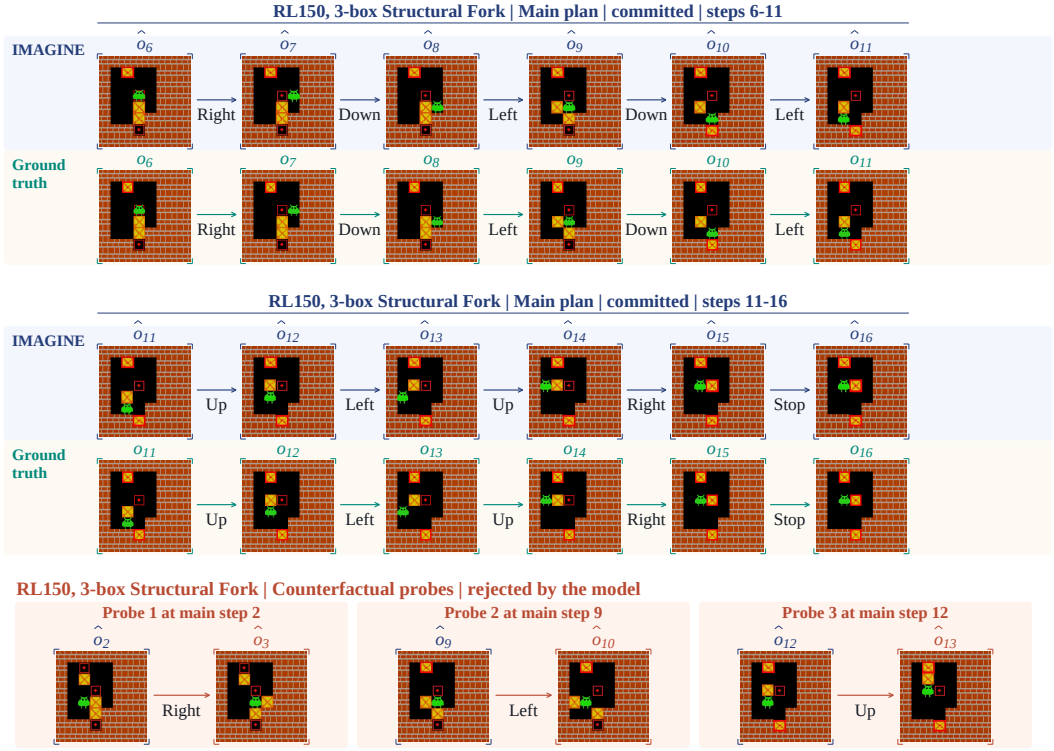
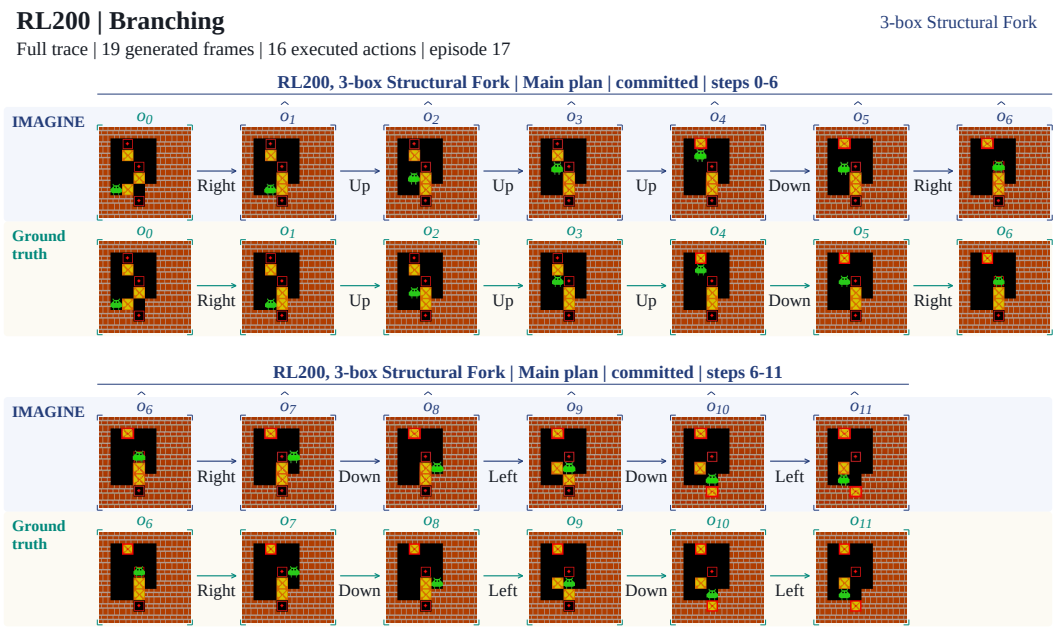


Figure 43: **Branching at RL150 on three-box Structural Fork.** Complete trace for episode 17; recorded execution ends in success.



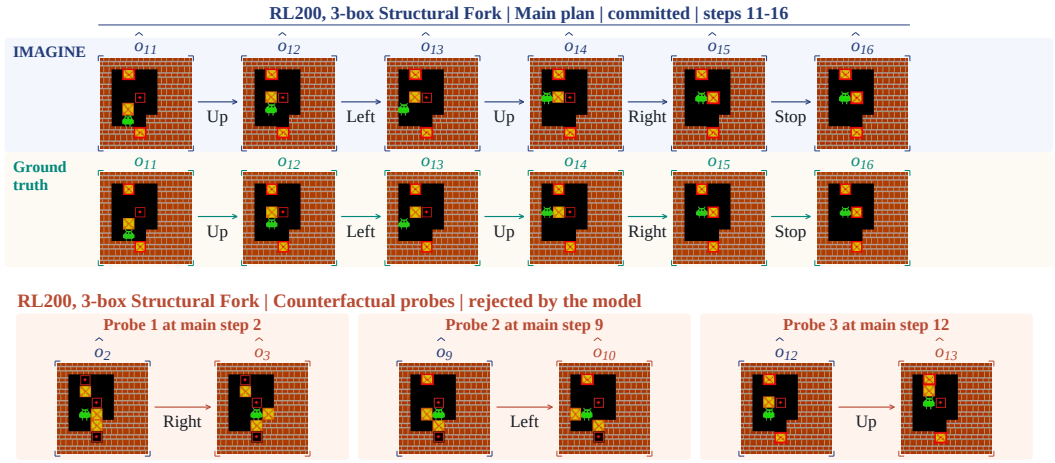


Figure 44: **Branching at RL200 on three-box Structural Fork.** Complete trace for episode 17; recorded execution ends in success.

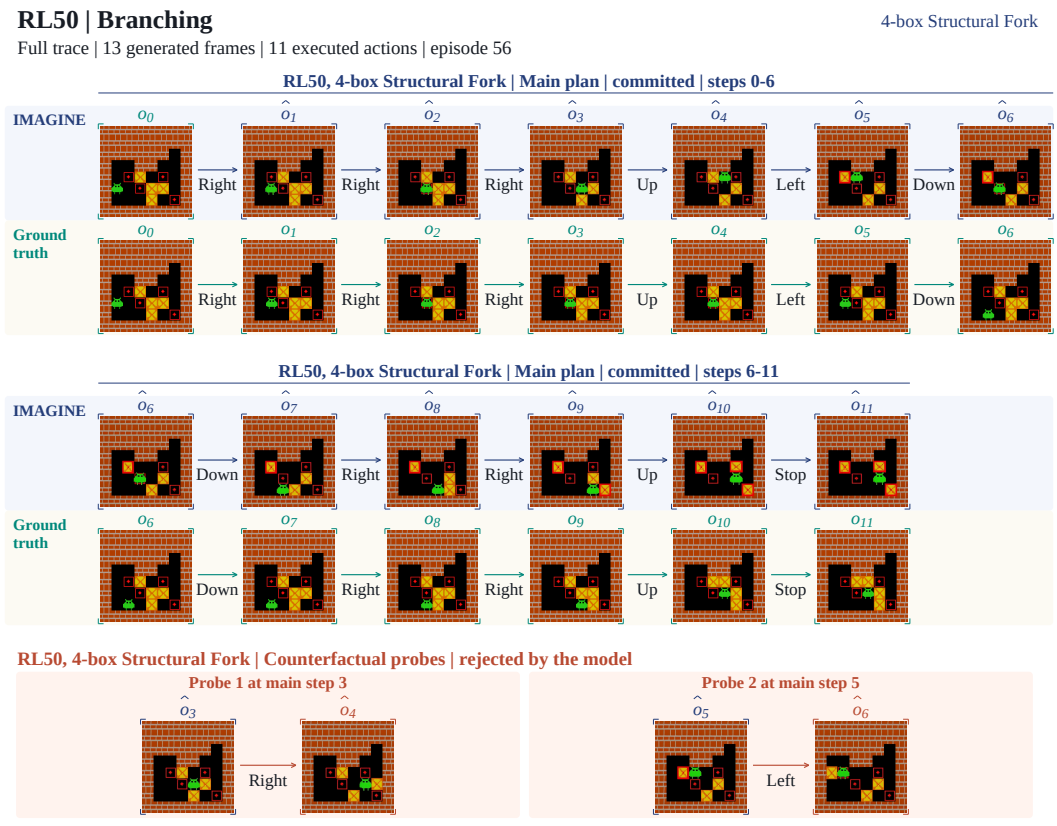


Figure 45: **Branching at RL50 on four-box Structural Fork.** Complete trace for episode 56; recorded execution ends in failure.



Figure 46: **Branching at RL100 on four-box Structural Fork.** Complete trace for episode 56; recorded execution ends in failure.





Figure 47: **Branching at RL150 on four-box Structural Fork.** Complete trace for episode 56; recorded execution ends in success.



Figure 48: **Branching at RL200 on four-box Structural Fork.** Complete trace for episode 56; recorded execution ends in success.

D.7 Failure Cases

The following recorded $K = \infty$ episodes show failures in all four environments: stopping before reaching the goal in Maze2D, leaving boxes off their goals in Sokoban, being caught by the ghost in Pacman, and stopping with insufficient target coverage in PushT. These examples use the same task settings and checkpoints as the $K = \infty$ evaluations in Section 3.

The real execution can also disagree with the model's outcome assessment. For example, the RL50 Forward model describes its chosen plan as successful in Figure 33, but the executed plan fails. In the Pacman example below, execution ends at the collision. Later predicted frames remain visible, with the corresponding real states marked as not executed.

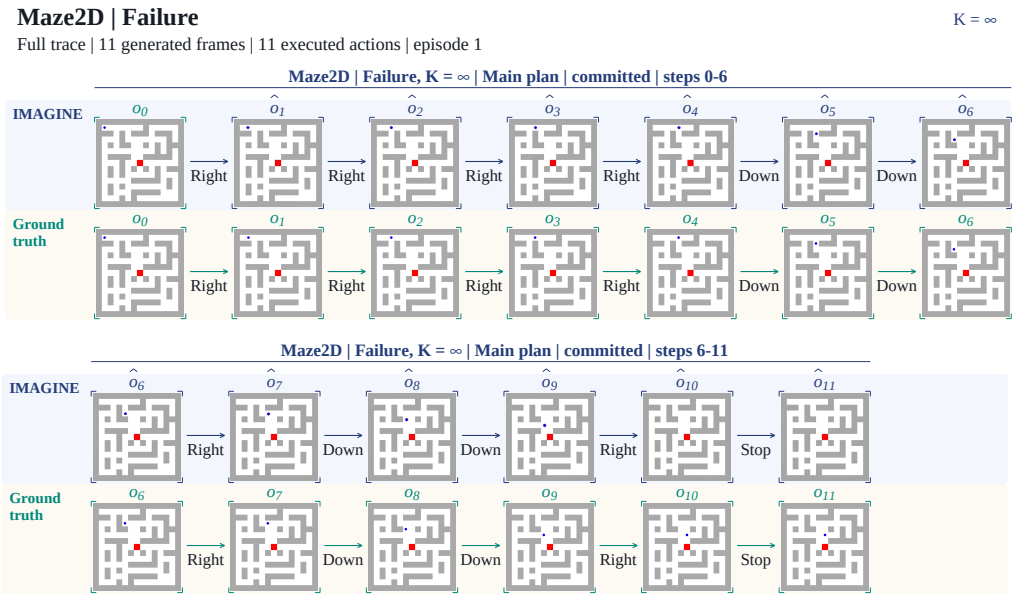


Figure 49: **Maze2D failure at $K = \infty$.** The agent issues `STOP` before reaching the goal.



Figure 50: **Sokoban failure at $K = \infty$.** The agent stops with 2 of 4 boxes on goals.

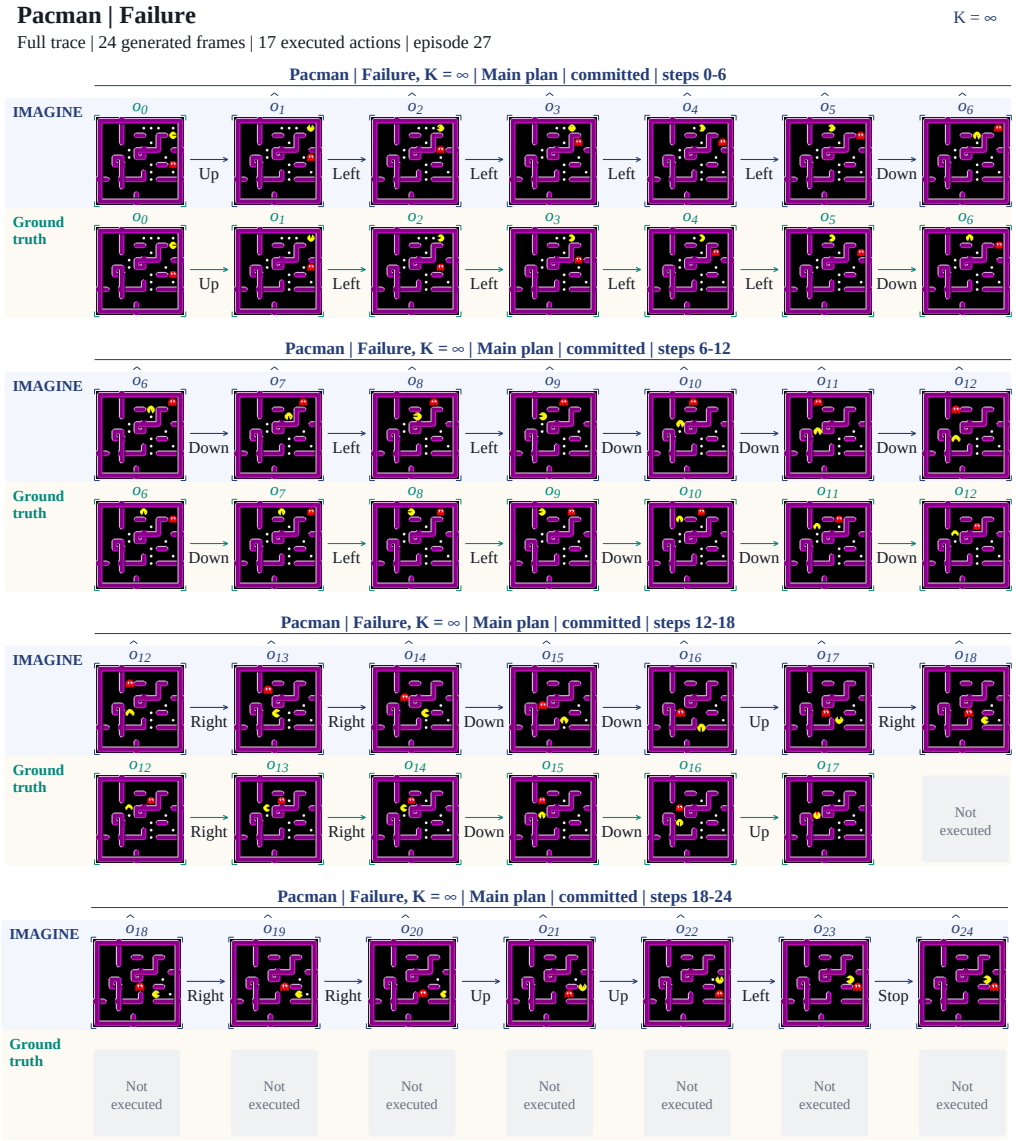
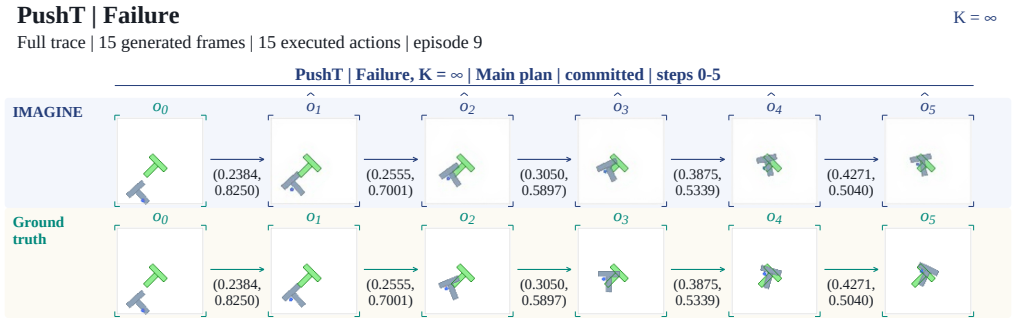


Figure 51: **Pacman failure at $K = \infty$.** The ghost catches Pacman at action 17. The remaining seven predicted states are never executed.



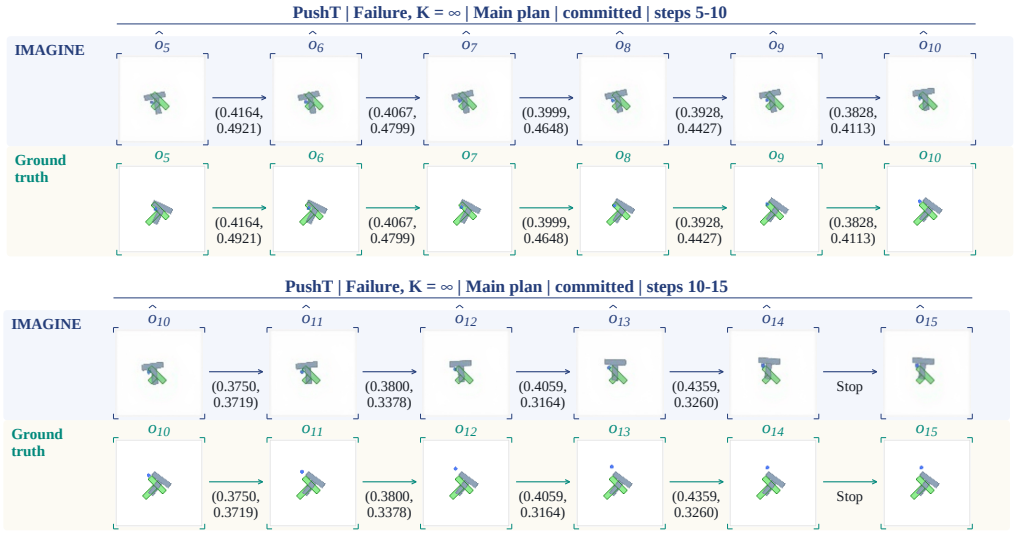


Figure 52: **PushT failure at $K = \infty$.** The agent stops with 32.4% final target coverage, below the paper's 80% success threshold.