

Deep Discovery Is Not Reward Shaping: Precondition Reasoning, Subgoal Navigation and Capacity Diagnosis Solve a Real Adventure Game with No LLM and No Script

Cercle Digital (independent research)
Correspondence: research@cercle.digital

Abstract

Long-horizon discovery—reaching a reward that requires an *ordered chain of prerequisites never rewarded along the way*—is commonly attacked with intrinsic rewards, i.e. by shaping a scalar exploration bonus. On the real 1977 Colossal Cave Adventure (141 rooms, non-destructive interventions `do()` by copying the full game state, nothing scripted, no large language model (LLM) anywhere in the agent), we show that the scalar bonus is the wrong lever, and identify the right one, in four measured steps. (1) *Value, not depth*: learned value propagation (tabular value iteration with R-max optimism, an optimism-under-uncertainty exploration rule) reaches a sparse goal seven steps deep, beyond the horizon where bounded lookahead search is null (steps-to-goal $54 \rightarrow 8$ across episodes while the bounded planner never converges). (2) *Lifting, not options*: lifting the state from the room identifier to (room, flags), with the flags discoverable by intervention, makes value iteration Markov and closes exploitation of the deep gated chain (1.00 ± 0.00 late-episode reach for the lifted oracle vs 0.08 ± 0.11 flat); mechanistically, the lift removes the curl (the cyclically inconsistent part) of the observed value flow to numerical zero ($0.030 \rightarrow 0.000$, where a same-budget random refinement leaves 0.065). (3) *Reasoning, not shaping*: myopic and value-propagated intrinsic state-change bonuses both fail the discovery chain (0/6 seeds), because a scalar cannot assemble an ordered prerequisite sequence whose bottleneck event must be experienced once before it can be credited; explicit backward-chaining over the prerequisite graph recovered by intervention succeeds (6/6 vs 0/6 on the local precondition), and the same demarcation reproduces on executable code-synthesis benches (backward-chaining 1.00 ± 0.00 flat in depth and in distractor load, with a real test-runner touch, while the propagated intrinsic collapses to 0.03). (4) The residual is not exotic: subgoal *navigation carrying the tool* (3/4 vs 1/4 on freeing the snake) and a *carry-capacity limit diagnosed by intervention* (4/4 vs 0/4 on the isolated final take) complete an autonomous reproducible chain, catch $\rightarrow$ free $\rightarrow$ reach $\rightarrow$ take, end-to-end treasure in hand in 3 of 4 seeds, game score 31.5 ± 3.3 vs 25.5 ± 1.0 for the reasoning-only baseline. Deep discovery is structural reasoning over an intervention-recovered graph; reward shaping is the wrong abstraction level. A self-contained code supplement reproduces every number.

1 Introduction

Some rewards cannot be stumbled into. In the 1977 Colossal Cave Adventure, the deep treasure behind the snake requires an exact, ordered sequence: drop the rod (which, unannounced, scares the bird), catch the bird with the cage, carry it to the snake’s hall, free it there (the bird drives the snake off, opening the deep passage), walk into the treasure room, and make room in a full inventory before taking the treasure. No step yields game score until the whole chain completes; several steps are *invisible in the room prose*; and the key event (the bird freed at the right place) must be *experienced once* before any learning signal can attach to it. This is the long-horizon sparse-reward problem in its hardest everyday form: not a distant reward to propagate, but a structured discovery to assemble.

The dominant reflex of the field is to attack such problems with *intrinsic rewards*: a scalar exploration bonus for novelty (Pathak et al., 2017; Burda et al., 2019), model disagreement (Pathak et al., 2019), empowerment (Klyubin et al.,

2005; Salge et al., 2014), or return-then-explore schedules (Ecoffet et al., 2021). The bonus reshapes the reward landscape so that ordinary value maximisation is pulled toward the unknown. This paper measures a case where that entire lever fails for a structural reason, and where a different, older lever succeeds: explicit backward-chaining (Newell & Simon, 1963; Fikes & Nilsson, 1971) over a prerequisite graph that the agent *recovers by intervention*.

Terminology, up front. Four terms recur from the title onward. `do()` denotes a Pearl-style *intervention* (Pearl, 2009): acting on the world (here, on a full copy of the game state, so the probe is non-destructive) and reading the world’s reproducible response, as opposed to passively reading the prose. *Reward shaping* is the addition of a scalar bonus to the reward; potential-based shaping (Ng et al., 1999) is its policy-invariant form, and intrinsic-motivation bonuses (Singh et al., 2010) are its exploration-directed form. A *state lift* maps the observed state (a room identifier) to a richer state (room plus Boolean flags) so that transitions

become Markov, i.e. outcome depends on the lifted state only (Amarel, 1968; Li et al., 2006). *Backward-chaining* is means-ends analysis: to achieve a goal, recursively achieve its discovered prerequisites first (Newell & Simon, 1963). Finally, *no LLM and no script* means: no large language model proposes actions anywhere in the agent, and no puzzle solution, no room identifier semantics, and no prerequisite fact is coded in; every such fact is discovered by `do()`.

The research question. *When a reward is gated by an ordered chain of prerequisites that is never rewarded along the way, is the missing capability a better scalar exploration bonus, or explicit structural reasoning over an intervention-recovered prerequisite graph?*

What existing approaches do not capture. Text-game agents from the literature attack Adventure-class games with a language prior: a language model proposes plausible actions (Yao et al., 2020), or a knowledge graph built from the text constrains generation (Ammanabrolu & Riedl, 2019; Ammanabrolu & Hausknecht, 2020), on top of the Jericho benchmark (Hausknecht et al., 2020) or TextWorld (Côté et al., 2019). The prior helps exactly because it *imports* the affordance knowledge (a snake fears a bird) that our agent must discover; it does not answer the discovery question, it rents the answer. Intrinsic-reward methods answer the wrong half: they excel at *covering* state space (and we confirm coverage benches pass here too) but, as we measure, no scalar bonus, myopic or value-propagated, assembles the ordered chain, because the bonus for the bottleneck event exists only after the event has been produced once, and producing it requires already holding the right tool at the right place with the right (invisible) precondition satisfied. Classical planning owns the right mechanism (means-ends analysis, landmarks; Hoffmann et al., 2004) but assumes the operator model is *given*; here nothing is given, and the contribution is the demarcation plus the recovery of the needed structure by intervention.

Contributions. On one real game, with fixed seeds, confidence intervals, strong baselines, ablation-style controls and accepted nulls, we establish a four-step chain of measured results (each step names its bench; every bench is a script in the code supplement):

1. **Substrate premise** (TOUCH-VS-READ): the room prose underdetermines the affordances; a frozen-sentence-encoder reader drops from 0.86 to 0.60 AUC on held-out rooms while the `do()` probe is exact everywhere. Reading is not a substitute for intervention on this substrate (section 3).
2. **Value, not depth** (GOAL-VALUE): bounded lookahead is null on sparse goals (an accepted null), while tabular value iteration with R-max optimism (Brafman & Ten-

nenholtz, 2002) bootstraps a goal beyond the planning horizon, steps-to-goal $54 \rightarrow 8$ (section 4).

3. **The lift closes exploitation** (STATE-LIFT-VALUE, LIFT-KILLS-CURL): value iteration over (room, flags) reaches the deep treasure 1.00 vs 0.00 flat; geometrically, the flat value flow has curl that the lift, and not a same-budget random refinement, removes to numerical zero (section 5).
4. **Reasoning, not shaping** (INTRINSIC-MYOPIC, INTRINSIC-PLANNED, BACKWARD-CHAINING, CODE-CHAIN, CODE-CHAIN-LATENT, CODE-CHAIN-PYTEST): every scalar-bonus arm fails the chain (0/6); backward-chaining over the `do()`-recovered prerequisite graph cracks the local precondition (6/6), and the demarcation reproduces on executable code with a real `pytest` touch (section 6).
5. **The last links** (SUBGOAL-NAVIGATION, CAPACITY-TAKE): means-ends navigation over the learned map carries the tool to the gate (3/4 vs 1/4), the carry-capacity gate on the final take is diagnosed by intervention (4/4 vs 0/4 in isolation), and the integrated agent completes the chain end-to-end, treasure in hand, in 3 of 4 seeds with score 31.5 ± 3.3 (section 7).

The synthesis (Figure 1) is the paper’s claim: *deep discovery decomposes into exploitation (closed by a state lift plus value iteration), precondition discovery (closed by intervention toggling), ordering (closed by backward-chaining), navigation (closed by means-ends planning on the learned map), and capacity diagnosis; no scalar intrinsic reward substitutes for any of the last three, and each mechanism is classical. What is new is the measured joint necessity, and the fact that the required graph is recoverable by `do()` alone.*

2 Related work

Agents for text games. TextWorld (Côté et al., 2019) and Jericho (Hausknecht et al., 2020) framed parser-based games as RL environments; Jericho explicitly lists Adventure-class games among the hardest because of sparse, deeply gated rewards. The strongest agents attack the action space with a language prior: KG-DQN and KG-A2C build a knowledge graph from the observed text and use it to constrain action generation (Ammanabrolu & Riedl, 2019; Ammanabrolu & Hausknecht, 2020); CALM fine-tunes a language model on human transcripts to propose candidate actions (Yao et al., 2020). All of these import affordance knowledge from human text. Our question is orthogonal: with *no* language prior at all, what mechanism discovers the gated chain? We therefore compare against the levers such agents rely on implicitly (exploration bonuses, value propagation, search depth) rather than against their rented priors; running the published LM-driven agents on this exact substrate is future work (section 9).

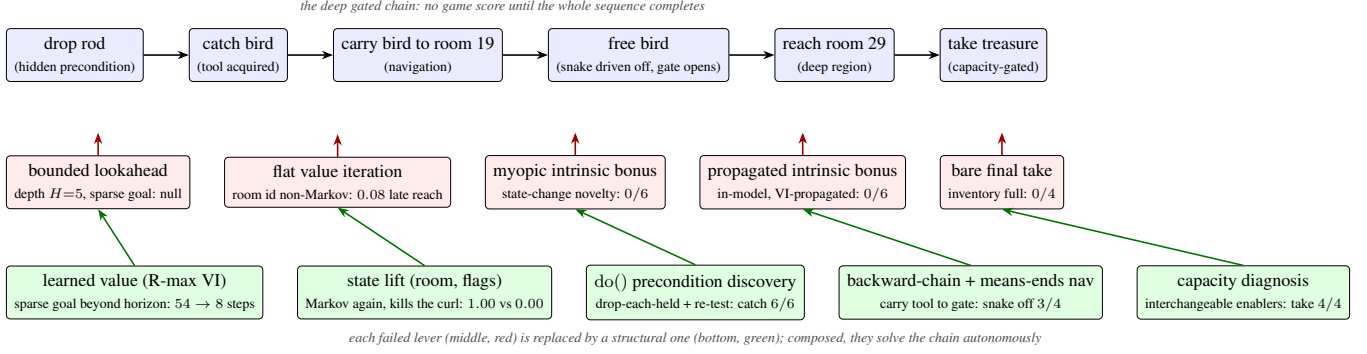


Figure 1: The demonstration at a glance. Top: Adventure’s deep gated treasure chain; no step is rewarded before the end, the rod precondition is invisible in the prose, and the gate event (free the bird at the snake) must be produced once before any learning signal exists for it. Middle: the levers that fail, each an accepted, pre-registered null (bounded search depth; flat value iteration; myopic and value-propagated intrinsic state-change bonuses; the bare final take). Bottom: the structural levers that close each failure, all driven by non-destructive `do()` interventions, composed into an autonomous solve (score 31.5 ± 3.3 vs 25.5 ± 1.0). Bench names and numbers in [section 4–section 7](#).

Intrinsic exploration. Curiosity by prediction error (Pathak et al., 2017), random network distillation (Burda et al., 2019), ensemble disagreement (Pathak et al., 2019), empowerment (Klyubin et al., 2005; Salge et al., 2014), and Go-Explore’s return-then-explore (Ecoffet et al., 2021) all shape a scalar signal that makes coverage efficient. Our nulls do not contradict that literature: coverage is exactly what our exploration benches confirm ([section 3](#)). The measured claim is narrower and sharper: on an *ordered* prerequisite chain whose bottleneck event carries no signal until produced, a scalar bonus, whether applied myopically or propagated through the value function, does not assemble the order ([section 6](#)). Ng et al. (1999) prove that potential-based shaping cannot change the optimal policy; our result is the empirical cousin one level up: shaping the exploration scalar does not produce the structured behaviour either, at matched budgets, while explicit reasoning does.

Classical planning and state abstraction. Means-ends analysis (Newell & Simon, 1963), STRIPS operators (Fikes & Nilsson, 1971) and ordered landmarks (Hoffmann et al., 2004) are the direct ancestors of our reasoning arm: our agent backward-chains over discovered preconditions and navigates to subgoal sites. The difference is that planning classically assumes the domain model; here every operator fact (the rod blocks the catch; the freed bird opens the gate; the take is capacity-gated) is *recovered by intervention* on the real game. Likewise the state lift instantiates the classical insight that representation determines problem difficulty (Amarel, 1968), formalised in MDP state-abstraction theory (Givan et al., 2003; Li et al., 2006; Abel et al., 2018). Hierarchical RL frameworks (options, option-critic, off-policy goal-conditioned hierarchy; Sutton et al., 1999; Bacon et al., 2017; Nachum et al., 2018) offer machinery for the same residual; our measurements say the machinery that suffices here is smaller: a lift plus value iteration closes exploitation

without an options stack, and an explicit subgoal (carry the tool to the gate) closes navigation.

3 Substrate, interventions, metrology

The world. All game benches run on the real Colossal Cave Adventure (Jerz, 2007), in the faithful pure-Python port of the 1977 original (141 rooms, *adventure* on PyPI). The game state is cheap to copy, which gives the agent a non-destructive `do()`: execute a candidate command on a deep copy and read the world’s reproducible response (moved or not, score delta, death), without committing. Everything the agent does goes through the game’s own parser and rules; no simulator abstraction is inserted.

The deep chain. The benches target the deep treasure behind the snake ([Figure 1](#), top). The full Markov state for this chain is the room plus four Boolean facts: holding the rod, holding the cage, holding the bird, and the snake’s state. The flat room identifier is *not* Markov: the same (room, action) pair succeeds or fails depending on hidden bits (the catch fails while the rod is held; the deep passage opens only after the snake is driven off).

Metrology. Every bench fixes its seeds and threads explicit random generators, and the seed is also passed to the *game’s own* internal generator (which drives its stochastic elements: wandering dwarves, pit falls), so every run is reproducible bit-for-bit; results are mean $\pm$ half-width of the 95% z-interval over seeds. The game benches use 4–8 seeds; seeds perturb the agent’s tie-breaking and exploration order and the game’s stochastic events. We flag this seed count explicitly in [section 9](#). Hypotheses are pre-registered in each bench script with the accept-null discipline: a failed hypothesis is reported as a result with its diagnosed mechanism, and

three of the paper’s pillars *are* such nulls. Discovery benches force the start at the bird room holding cage and rod, so that the separate, already-characterised problem of reaching the chain from the surface does not confound the discovery measurement; the chain itself ($13 \rightarrow 19 \rightarrow 29$) is always walked autonomously unless a bench explicitly states otherwise.

3.1 The premise: reading underdetermines, `do()` does not (TOUCH-VS-READ)

Hypothesis (pre-registered). The room prose underdetermines the exit affordances: a reader trained on prose drops on held-out rooms, while the `do()` probe is exact everywhere. (Accepted-null alternative: prose predicts exits near-perfectly, and reading suffices.)

Experiment (operationalisation). Map the puzzle-free reachable region (17 rooms; a mean of 3.7 of 8 directions are valid exits). The *observation* arm embeds each room’s prose with a frozen MiniLM sentence encoder (Reimers & Gurevych, 2019; Wang et al., 2020) and fits a logistic regression from (prose embedding, direction) to exit validity, trained on 60% of rooms and tested on held-out rooms; 12 random room splits give the interval. The *touch* arm issues `go d` on a copy and reads whether the room changed. Downstream payoff: a navigation policy must pick a valid exit in a held-out room.

Result. Observation AUC 0.861 ± 0.016 in distribution $\rightarrow 0.598 \pm 0.026$ on held-out rooms; the touch is exact (1.000) everywhere with no training. The observation navigation policy picks a blocked exit in a quarter of held-out rooms (success 0.762 ± 0.092); the touch policy succeeds always (1.000).

Conclusion. On this substrate, affordances must be measured, not read. Every later bench therefore grounds its facts in `do()`; this is also why an agent without a rented language prior is the right instrument for the discovery question.

3.2 Context: coverage works, and planning depth adds nothing (EXPLORE-PLAN)

Two context measurements bound what the simple levers already achieve on the full game. First, a 1-step intervention learner (deep-copy lookahead over the full verb $\times$ object action space, count-based novelty over (room, inventory), score as extrinsic value) covers 16.7 ± 2.6 unique rooms against 10.3 ± 3.1 for random at a 320-action budget (6 seeds): intervention plus curiosity demonstrably *learns to explore*, with nothing scripted. Second, adding receding-horizon best-first planning in the deep-copy world model (horizon 5, 220 expansions per decision) does *not* beat the reactive learner (both at 17.5 ± 0.6 rooms, 4 seeds).

In both measurements the in-game score tracks only the game’s depth-progression points (learner 58.3 ± 0.8 vs random 44.8 ± 11.3 , i.e. coverage in another currency): no gated treasure chain is ever completed, because those are the 10-plus-step unrewarded sequences no 1-step novelty and no bounded search sees. Both facts are accepted results, not straw men: they set up exactly which capability is missing.

4 Step 1: value, not depth (GOAL-VALUE)

Hypothesis (pre-registered). (A) On a sparse goal at intermediate depth, a bounded best-first planner does not beat a reactive novelty agent (expected null: an undirected bounded search has no heuristic toward a specific goal). (B) A learned value, propagated backward over experience by value iteration with R-max optimism, reaches a goal *beyond* the planning horizon, and its steps-to-goal drop across episodes (bootstrapping).

Experiment (operationalisation). (A) Reward only at a target room four steps deep; arms random, reactive (1-step), planner (horizon 5, 300 expansions); 8 seeds, budget 30 actions. (B) The decisive isolation: target room seven steps deep, planner horizon fixed at 4, so the reward is provably outside the planner’s tree. The learned-value arm learns the transition map $T(\text{room}, \text{dir})$ from its own moves and runs tabular value iteration where untried directions carry R-max optimism (Brafman & Tenenholz, 2002; Sutton & Barto, 2018); the value persists across 12 episodes (cap 60 steps); 4 seeds.

Result. (A) Null, as pre-registered: planner 19.0 ± 7.8 steps-to-goal at 0.88 ± 0.24 reach vs reactive 23.5 ± 6.7 at 0.62 ± 0.36 (random 25.2 ± 7.4 at 0.25 ± 0.32): directionally better, nowhere CI-separated; bounded depth buys no reliable edge. (B) The learned value bootstraps: mean steps-to-goal per episode falls 41, 61, 61, 61, 48, 38, 34, 12, 9, 8, 8, 9 (61 = censored), i.e. early 54.4 $\rightarrow$ late 8.1, which is the optimal path length; the horizon-4 planner, whose search tree provably never contains the reward, drifts in and out on novelty alone and never converges (late 39.1, CI-separated from the learned value’s 8.1) (Figure 2).

Conclusion. On distant sparse rewards the lever is a learned, propagated value, not raw search depth. This closes the *propagation* half of the problem: once a reward has been encountered, value iteration carries it back. It says nothing yet about *discovering* a reward whose chain is never completed by chance; that is the subject of the rest of the paper.

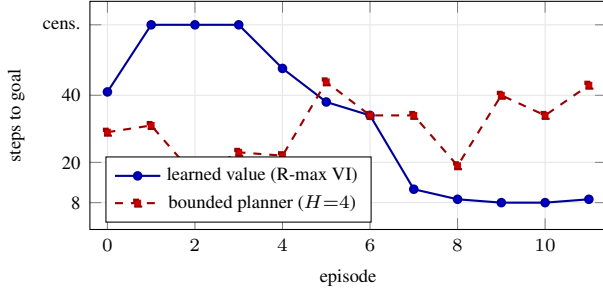


Figure 2: Value, not depth (GOAL-VALUE, part B). Mean steps to a sparse goal seven steps deep, per episode (4 seeds; 61 = not reached within the 60-step cap). The goal is outside the horizon-4 planner’s search tree, so its search never sees the reward: it drifts in and out on novelty alone and never converges. The learned value stumbles onto the goal under R-max optimism, then value iteration propagates the reward backward: later episodes go straight there (~ 8 steps = the optimal path).

5 Step 2: the lift closes exploitation (STATE-LIFT-VALUE, LIFT-KILLS-CURL)

Flat value iteration cannot even *exploit* the deep chain once known, because the room identifier is not Markov there: the same room catches or fails to catch the bird depending on a hidden bit, so the learned transition table carries contradictory entries and no stable value field forms. The classical fix is state abstraction in reverse: *lift* the state.

5.1 Value over the lifted state (STATE-LIFT-VALUE)

Hypothesis (pre-registered). With navigation forced along the chain skeleton (so that only the affordance choices gated by hidden flags remain), value iteration over (room, flags) reaches the deep treasure in late episodes, while flat room-id value iteration never stabilises. The lifted flags are discoverable by intervention; an oracle arm (flags given) upper-bounds the abstraction, isolating “is the lift the lever” from “does the touch discover the flags in this budget”.

Experiment (operationalisation). Arms flat / lifted (flags discovered by interventional toggling: drop each held item and re-test a failing action; plus any world variable observed to change) / oracle (the four necessary bits given). Navigation is forced along $13 \rightarrow 19 \rightarrow 29$ via an advance macro, *except* the $19 \rightarrow 29$ leg, which stays gated by the real game state (the snake must actually have been driven off). Episodic value iteration, value persisting across 15 episodes (cap 60), 8 seeds; metric: treasure-reach rate over the last three episodes.

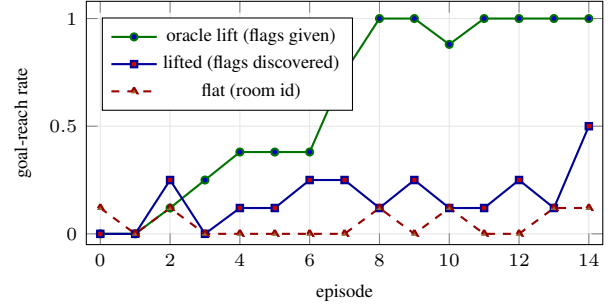


Figure 3: The lift closes exploitation (STATE-LIFT-VALUE). Per-episode deep-treasure reach rate, mean over 8 seeds, value persisting across 15 episodes; navigation forced along the skeleton so only the flag-gated affordance choices are measured. The oracle lift bootstraps to 1.00 and stays; flat room-id value iteration never stabilises (its non-Markov transition table is contradictory at the gates); the autonomously discovered lift sits in between, bounded by flag-discovery coverage.

Result. Late-episode reach (Figure 3): oracle 1.00 ± 0.00 , with a textbook bootstrap curve (per-episode reach rises $0.12 \rightarrow 0.25 \rightarrow 0.38 \rightarrow 0.75 \rightarrow 1.00$ and holds ~ 1.00 over the last six episodes), flat 0.08 ± 0.11 (half the seeds reach once, by accident, and no seed ever stabilises), lifted (flags autonomously discovered) 0.29 ± 0.26 late, directionally above flat but not CI-separated, with 62% of seeds reaching at least once. The load-bearing contrast is oracle vs flat (CI-clean); the oracle-lifted gap is flag-discovery *coverage*: the rod flag is found immediately by the toggling touch, but the snake flag can only be observed to change after the bird has been freed once at the right room, the chicken-and-egg this paper is about.

Conclusion. The lift is the lever for exploitation: given the right four bits, plain value iteration solves the deep chain, no options framework, no language model. What remains open after this bench is precisely the *acquisition* of the chain (discover and order the events), carried by the oracle-lifted gap; steps 3 and 4 close it.

5.2 The geometric reading (LIFT-KILLS-CURL)

Hypothesis (pre-registered). A non-Markov state manifests as *curl* in the observed value flow: project a representation-independent progress potential (oracle distance-to-goal on the full state) onto each state representation’s graph and Hodge-decompose the resulting edge flow (Jiang et al., 2011); then (HC1) the flat room graph carries a non-zero cyclically inconsistent fraction, (HC2) the lift removes it while a same-budget *random* refinement does not (structure, not node count), and (HC3) the inconsistency localises on the prerequisite-chain rooms.

Experiment (operationalisation). Execute the real chain and probe the local room subgraph at each phase with movement-only `do()` probes; compute the oracle distance by reverse breadth-first search on the full state (non-circular: no optimal value is used); aggregate the per-transition progress onto flat / lifted / random-lift (room plus k random bits, k = number of oracle flags) representations; inconsistency = curl-plus-harmonic fraction of the Hodge decomposition. 6 seeds.

Result. Inconsistency: flat 0.030 ± 0.003 , lifted 0.000 ± 0.000 (numerically zero), random-lift 0.065 ± 0.045 (a same-size but irrelevant refinement makes the flow *worse*, not better); restricted to the non-Markov subregion (rooms hosting more than one oracle distance) the flat inconsistency rises to 0.099 ± 0.013 ; the flat residual concentrates on the chain rooms (0.045 ± 0.000 vs 0.017 ± 0.003 elsewhere). All three pre-registered conditions hold.

Conclusion. The flat-value failure of STATE-LIFT-VALUE has a mechanistic reading: the room graph’s value flow is non-conservative exactly at the gates, so no value-per-room potential exists there, and the lift is the exactness repair. The honest magnitude matters: the curl is globally small (the game is mostly Markov at room level), which is precisely the profile of a world where flat value iteration mostly works and fails *specifically* on the deep gated chain. This subsection adds no capability; it explains one.

6 Step 3: reasoning, not shaping

Steps 1–2 close propagation and exploitation. The remaining wall is *acquisition*: complete the ordered, unrewarded chain once, so that there is something to propagate. This section measures the field’s default lever (a scalar intrinsic bonus) in two escalating forms, accepts both nulls, then measures the structural alternative, on the game and on code.

6.1 Scalar intrinsic bonuses: two pre-registered nulls (INTRINSIC-MYOPIC, INTRINSIC-PLANNED)

Hypothesis (pre-registered). An intrinsic reward for *causing a novel world-state change* (a world variable takes a new value; a blocked exit opens; a discovered flag flips), count-decayed so the drive is transient, drives the agent through the bottleneck: myopically added at action selection (INTRINSIC-MYOPIC), or, stronger, stored in the model per (lifted state, action) and *propagated by value iteration* so the agent can plan multi-step toward reachable state-change opportunities (INTRINSIC-PLANNED). Accepted-null alternative: no scalar form assembles the chain.

Experiment (operationalisation). Arms per bench: `rmax` (R-max frontier optimism only), the intrinsic arm, and a flat room-id control for the intrinsic. Start forced at the bird room with cage and rod; the agent walks $13 \rightarrow 19 \rightarrow 29$ itself; budget 450 actions; 6 seeds. Milestones: bird caught, snake driven off (the bottleneck), deep room reached, treasure taken.

Result. Both nulls, as pre-registered. *No intrinsic arm ever drives the snake off*: the myopic bonus 0/6, the propagated bonus 0/6, and the propagated bench’s flat control and no-bonus baseline 0/6. Across all 36 runs of the two benches the bottleneck fires exactly twice, and both firings are *without* the tested bonus (the myopic bench’s no-bonus `rmax` baseline once, 1/6, and its flat control once, 1/6): undirected wandering occasionally lucks through at budget 450, and adding the bonus does not raise that rate above its own baseline. The myopic arm does catch the bird (5/6, the local, one-step-visible discovery) but stalls: a 1-step bonus carries no gradient toward the distant site where the next bottleneck becomes possible. The propagated arm fails differently and more instructively: the value field pulls the agent toward *generic* reachable novelty (fresh rooms, fresh acquisitions), which is abundant, rather than toward the single event that matters; at matched budget it does not even retain the bird (0/6 caught at budget 300 in the head-to-head of [subsection 6.2](#)).

Conclusion. The diagnosis is structural, not a tuning accident. The bottleneck event (bird freed at the snake) has zero intrinsic value *before it has ever been produced*, because count-based novelty is defined over experienced changes; producing it requires an ordered conjunction (hold the bird, at the right room, rod long since dropped) that a scalar field, whose maxima are unordered, does not encode. Shaping moves value around; it cannot inject the missing *order*.

6.2 Backward-chaining over the `do()`-recovered prerequisite graph (BACKWARD-CHAINING)

Hypothesis (pre-registered). A means-ends agent that (i) discovers negative preconditions by interventional toggling (when an acquisition verb fails, `do()` drop each held item and re-test; exactly one enabling item = a genuine negative precondition, distinguishing it from the capacity limit where any drop enables), (ii) discovers empowerment effects (an interaction that opens a previously blocked exit, read by a two-step `do()` probe), and (iii) backward-chains (drop the blocker *before* acquiring; pursue the unlock that opens the deep region), cracks the chain where the best scalar bonus does not.

Experiment (operationalisation). Arms: `reward` (the propagated intrinsic of INTRINSIC-PLANNED, the strongest shaping baseline) vs `reason` (the structural agent), matched start (bird room, cage and rod held) and budget (300 actions); 6 seeds. Nothing is scripted: the rod-blocks-catch fact and the free-bird-unlocks fact are discovered by the probes above.

Result. `reason` catches the bird 6/6 with a tight score of 25.3 ± 0.7 ; `reward` catches it 0/6 (score 0.0 ± 0.0). The bottleneck yields partially: `reason` frees the snake 1/6 (only when its undirected wandering happens to carry the bird to the right room) and never reaches the deep room (0/6). See Table 2.

Conclusion. The affirmative half of the paper’s title: the *precondition* link of deep discovery cedes to explicit reasoning over intervention-recovered structure, and to nothing scalar we tried. The measured residual is equally sharp: caught 6/6 $\rightarrow$ snake off 1/6 is neither a reward problem nor a precondition problem; it is *navigation to the subgoal site while carrying the tool*, addressed in section 7.

6.3 The demarcation reproduces on executable code (CODE-CHAIN, CODE-CHAIN-LATENT, CODE-CHAIN-PYTEST)

If “reasoning, not shaping” is a fact about ordered unrewarded chains rather than about one 1977 game, it should reproduce in a different substrate with the same signal structure. We build one from program synthesis, where the touch is a real test execution.

Experiment (operationalisation). **World.** The target is a pipeline $\text{solve} = \text{op}_1 \circ \dots \circ \text{op}_d$ over lists; the agent picks each stage’s operation from a bank (one correct, several decoys). The top-level behavioural test (the extrinsic goal) passes only when *all* d stages are correct: unrewarded until complete. Prefix micro-checks (real execution of $\text{op}_1 \dots \text{op}_i$ against the hidden reference) form a genuine prerequisite chain; n_{side} distractor actions each flip a private side-check once (abundant generic novelty, the game’s confound made explicit). **Arms.** `search`: blind joint enumeration, cost $\sim B^d$; `planned`: the propagated count-decayed intrinsic over any newly passing check, value-iterated over the bit-mask state (the faithful port of INTRINSIC-PLANNED); `dag`: backward-chaining over the program’s own prerequisite graph, solving subgoals in dependency order by touch, never considering the side distractors. 6 seeds $\times$ 12 tasks, budget 90 actions. **Hardened variant** (CODE-CHAIN-LATENT): the prerequisite structure becomes a *latent, non-positional call graph* between helper functions that the agent must read (a structural touch on the code’s syntax tree) and confirm behaviourally, and K *deceptive* helpers form their own passing

Table 1: Real test-runner variant (CODE-CHAIN-PYTEST; every micro-check = a `pytest` subprocess on real files, memoised on the call closure; 3 seeds, latent world, $K=2$). The backward-chainer’s few-touch property becomes literal wall-clock: full solve in ~ 6 –11 real test runs.

depth L	search	planned	dag (real runs)
2	1.00	0.33	1.00 (5.7)
3	0.67	0.22	1.00 (8.3)
4	0.44	0.00	1.00 (10.6)

check-chains that look exactly like real subgoals, with decoy candidates that lure the backward-chainer into the trap chain. **Real-toucher variant** (CODE-CHAIN-PYTEST): every micro-check writes real files and runs `pytest` in a subprocess (memoised on the helper’s call closure); the reported cost is real test-runner invocations.

Result. Figure 4 and Table 1. On the pipeline world, `search` collapses with depth ($1.00 \rightarrow 0.15 \pm 0.07$ at $d=5$) and `planned` collapses with depth and with distractor load ($0.68 \pm 0.08 \rightarrow 0.03 \pm 0.03$ over depth; $0.06 \rightarrow 0.00$ as n_{side} grows to 120 at $d=5$): the propagated scalar drowns in abundant novelty, the exact mechanism of the game null. `dag` solves 1.00 ± 0.00 , *flat* in depth and in n_{side} , in $4.8 \rightarrow 12.3$ steps (linear growth). Under the hardening, `dag` still solves 1.00 ± 0.00 at depth 5 with 8 deceptive look-alike subgoal chains (steps rise linearly $11.2 \rightarrow 19.5$, the honest price of backing out of traps by touch) while `planned` stays at 0.05 – 0.07 . With the real `pytest` toucher, `dag` solves 1.00 at every depth tried in $5.7 \rightarrow 10.6$ real test runs, where `planned` drops to 0.00 and `search` to 0.44 by depth 4.

Conclusion. The demarcation is substrate-general in the tested range: whenever completion is unrewarded until the end and generic novelty is abundant, the propagated scalar fails while explicit backward-chaining over the touch-recovered dependency structure succeeds at linear cost, and this survives latent structure, adversarial look-alike subgoals, and a real test runner. The efficiency reading matters for practice: few touches means literally few `pytest` subprocesses.

7 Step 4: the last two links (SUBGOAL-NAVIGATION, CAPACITY-TAKE)

7.1 Navigation carrying the tool (SUBGOAL-NAVIGATION)

Hypothesis (pre-registered). The caught 6/6 $\rightarrow$ snake-off 1/6 residual of subsection 6.2 is navigation: adding

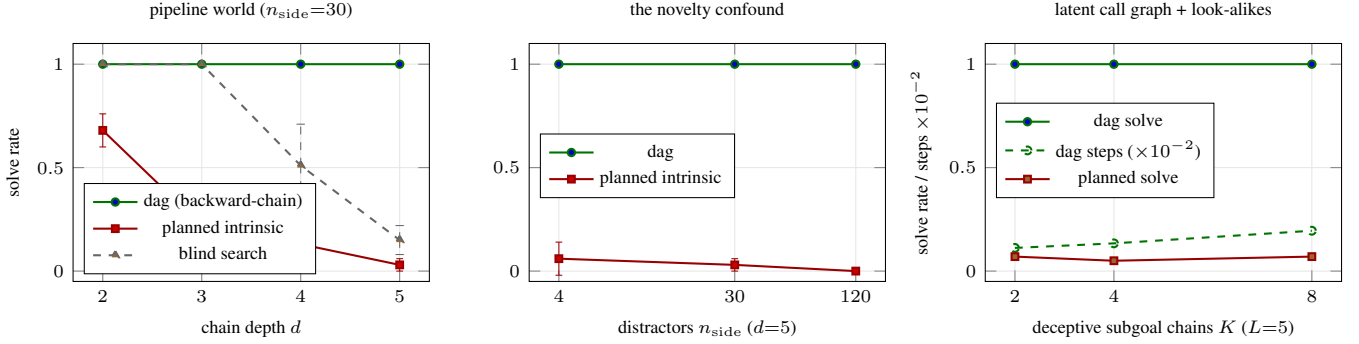


Figure 4: The demarcation on code (CODE-CHAIN, CODE-CHAIN-LATENT; 6 seeds, 95% CIs where wider than the marks). Left: solve rate vs prerequisite-chain depth; blind search collapses exponentially, the propagated intrinsic collapses too, backward-chaining over the touch-recovered dependency graph stays at 1.00. Middle: at depth 5, the propagated intrinsic also collapses as generic novelty grows ($0.06 \rightarrow 0.00$), the mechanism of the game null made explicit; the dag arm is flat at 1.00. Right: hardened world (latent non-positional call graph, K deceptive look-alike subgoal chains): the dag arm still solves 1.00; its step cost grows only linearly ($11.2 \rightarrow 19.5$, dashed), the price of backing out of decoy traps by touch.

means-ends navigation over the touch-learned map (learn $T(\text{room}, \text{dir})$ by acting; carry the held tool toward the exploration frontier so coverage goes deep; at every room with a genuinely blocked exit, try the held tools on the obstacle and read the unlock with the two-step `do()` probe) frees the snake where the wandering agent does not. (Accepted-null alternative: the residual is exploration coverage, not navigation.)

Experiment (operationalisation). Arms: `reason` (the subsection 6.2 agent) vs `reason+nav`; matched start and budget (220 actions); 4 seeds. Nothing scripted: both the rod precondition and the bird-opens-the-gate fact are discovered; the bird is carried to the gate by plan, not luck. One design fact matters and is reported because it changed the outcome: naively navigating to *any* blocked exit oscillates near the start (most blocked directions are walls, not puzzle gates); the working policy carries the tool to the *frontier* and tries it wherever a genuine blocked exit exists.

Result. `reason+nav` frees the snake 3/4, reaches the deep room 3/4 and, with its capacity micro-plan in the loop, takes the treasure 3/4, against 1/4, 0/4 and 0/4 for `reason` (both catch the bird 4/4); score 31.5 ± 3.3 vs 25.5 ± 1.0 (Table 2). The one failing seed is an exploration miss, not a mechanism failure: under that seed’s game randomness the frontier walk does not carry the bird through the gate room within the 220-action budget.

Conclusion. The chain caught $\rightarrow$ snake off $\rightarrow$ deep room $\rightarrow$ treasure is autonomous and reproducible end-to-end; the next bench isolates *why* the bare final take would fail without the capacity micro-plan.

7.2 The capacity diagnosis (CAPACITY-TAKE)

Hypothesis (pre-registered). The final failure is not conceptual: `take` is valid but the inventory is full. The same interventional toggling that discovers negative preconditions distinguishes the case: if exactly *one* held item enables the acquisition after being dropped, it is a genuine negative precondition; if *many* held items are interchangeable enablers, it is a capacity limit, and the generic micro-plan drop-one $\rightarrow$ take follows.

Experiment (operationalisation). Arms `take-only` vs `capacity-plan`, from the exact state the navigation agent reaches (deep room, prerequisite inventory carried); 4 replicate seeds of a deterministic diagnostic.

Result. `take-only` 0/4; `capacity-plan` 4/4: the touch finds multiple interchangeable enablers, classifies the gate as capacity, drops one item and takes the treasure.

Conclusion. With this mechanism in the loop the full chain closes: catch (4/4) $\rightarrow$ free (3/4) $\rightarrow$ reach (3/4) $\rightarrow$ take (3/4; every seed that reaches the deep room takes the treasure), autonomously, with no language model and no scripted fact, from the forced start at the bird room. Every mechanism used is classical (means-ends analysis, value iteration, state abstraction); the measured content is that all of them are needed, that each was identified by the failure mode of the previous lever, and that no scalar intrinsic bonus substitutes for the structural ones.

Table 2: Milestones on the deep chain (forced start at the bird room with cage and rod; the chain $13 \rightarrow 19 \rightarrow 29$ walked autonomously; nothing scripted, no language model). Left block: the strongest reward-shaping baseline vs the structural reasoner (BACKWARD-CHAINING; 6 seeds, budget 300). Right block: adding means-ends navigation over the learned map (SUBGOAL-NAVIGATION; 4 seeds, budget 220), and the final take isolated (CAPACITY-TAKE). Score is the in-game score delta. The two intrinsic-bonus benches (INTRINSIC-MYOPIC, INTRINSIC-PLANNED, budget 450) never free the snake in any intrinsic arm (0/6; only the no-bonus baseline and one flat control fire once each by undirected luck).

arm	bird caught	snake off	deep room	treasure	score
reward (propagated intrinsic), 6 seeds	0/6	0/6	0/6	0/6	0.0 ± 0.0
reason (backward-chaining), 6 seeds	6/6	1/6	0/6	0/6	25.3 ± 0.7
reason, 4 seeds (navigation baseline)	4/4	1/4	0/4	0/4	25.5 ± 1.0
reason + means-ends navigation, 4 seeds	4/4	3/4	3/4	3/4	31.5 ± 3.3
final take only (deep room reached)	–	–	–	0/4	–
final take + capacity diagnosis	–	–	–	4/4	–

8 What curiosity does and does not buy here

Nothing in this paper argues against intrinsic exploration for what it is good at. Our own coverage measurements (subsection 3.2) show interventional curiosity nearly doubling map coverage over random, and the exploration literature has strong evidence that disagreement-style epistemic bonuses avoid the noise traps that raw prediction-error curiosity falls into (Pathak et al., 2019). Go-Explore’s insight, remember promising states and return to them before exploring (Ecoffet et al., 2021), even shares a component with our navigation link (deliberate return to a frontier). The measured boundary is elsewhere: coverage-style bonuses value *being somewhere new*; an ordered prerequisite chain requires *doing specific things in a specific order while carrying specific state*, and the value of the bottleneck action is undefined for any experience-counting bonus until the action has succeeded once. Backward chaining does not search that space better; it *avoids searching it*, by deducing the order from discovered preconditions and effects. That is why its cost stays linear where search is exponential and shaping is flat-out null (Figure 4).

9 Limitations

One game, plus code worlds. The game-side results are on a single real game, chosen for its canonical deep gated chain; the code benches show the demarcation is not game-specific in the tested range, but no cross-game generalisation (other Jericho titles) is measured. **Baselines are lever-level, not agent-level.** We compare the levers (search depth, value propagation, scalar bonuses, structural reasoning) at matched budgets; we did not run published LM-based agents (CALM, KG-A2C) or a full Go-Explore implementation on this exact substrate, and the paper’s claims are correspondingly about mechanisms, not leaderboards. **Forced starts**

isolate discovery. Reaching the chain from the surface is a separate coverage problem (subsection 3.2); discovery benches start at the bird room, and the state-representation benches additionally force the navigation skeleton. The autonomous solve is therefore autonomous *from the forced start*. **Interventions are cheap here.** The non-destructive `do()` relies on the game state being copyable; the precondition search costs a number of probes linear in inventory size per site, and the exit probes are cached per room. In worlds where interventions are expensive or irreversible the same structure would have to be recovered under a probe budget, which we do not study. **Seeds.** The game benches use 4–8 seeds, each seeding both the agent and the game’s own random generator (runs are reproducible bit-for-bit); the pivotal contrasts (6/6 vs 0/6 on the precondition; 0/6 across every intrinsic arm) are wide, but replication at 10 or more seeds is the standard we hold ourselves to elsewhere and remains to be run before conference submission. **The lift’s flag inventory is small.** Four Boolean facts suffice for this chain; scaling interventional flag discovery when candidate facts are numerous is open (the parsimony filter used here admits a flag only when an intervention or an observed change shows it matters).

10 Conclusion

Each failure in this sequence named its lever. Bounded search depth was null (the planner never converges on a goal beyond its horizon); the bottleneck was value, and learned value propagation closed it ($54 \rightarrow 8$). Flat value was null on the deep chain (late reach 0.08 ± 0.11 , never stable); the bottleneck was state, and the intervention-discoverable lift closed it (1.00 oracle reach; curl $0.030 \rightarrow 0.000$, against a random-refinement control). The scalar bonus was null on the chain (0/6 in every intrinsic arm, myopic and propagated, lifted and flat); the bottleneck was *order*, and backward-chaining over the `do()`-recovered prerequisite graph closed the local precondition (6/6 vs 0/6), a demarca-

tion that reproduced on executable code at 1.00 flat in depth and distractor load, including with a real test runner. Reasoning alone still wandered (1/6 snake off): the residual was navigation, and means-ends planning over the learned map closed it (3/4 end-to-end, treasure in hand), with the final take’s carry-capacity gate diagnosed by the same toggling touch (4/4 vs 0/4 in isolation), for an autonomous score of 31.5 ± 3.3 vs 25.5 ± 1.0 . Every mechanism is classical; the paper’s contribution is their measured *joint necessity* on a real game, the demonstration that the required structure is recoverable by intervention alone (no language model, no script), and the demarcation itself: on ordered unrewarded chains, reward shaping, myopic or propagated, is the wrong abstraction level.

Three questions stay open. Does the demarcation hold across games with different gate structures (Jericho’s harder titles), and against full agent-level baselines run on this substrate? Can the interventional recovery of preconditions and effects operate under a hard probe budget, where every `do()` costs? And can the discovered prerequisite graphs be *accumulated* across tasks into a reusable operator model, turning the classical planning assumption (the model is given) into a measured acquisition process?

Reproducibility statement

A self-contained code supplement (the `code/` directory alongside this paper) reproduces every number: one script per bench (Table 3), public dependencies only (numpy, the adventure PyPI port of the 1977 game, torch for the Hodge decomposition of one bench, sentence-transformers and scikit-learn for the reading baseline, pytest for the real-toucher bench), fixed seeds with explicit generators, and pre-registered hypotheses printed by each script together with its controls and verdict.

References

- Abel, D., Arumugam, D., Lehnert, L., and Littman, M. L. State abstractions for lifelong reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 10–19, 2018.
- Amarel, S. On representations of problems of reasoning about actions. In Michie, D. (ed.), *Machine Intelligence* 3, pp. 131–171. Edinburgh University Press, 1968.
- Ammanabrolu, P. and Hausknecht, M. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*, 2020.
- Ammanabrolu, P. and Riedl, M. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3557–3565, 2019.
- Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 1726–1734, 2017.
- Brafman, R. I. and Tenenbholz, M. R-max: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.
- Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.
- Côté, M.-A., Kádár, Á., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., El Asri, L., Adada, M., Tay, W., and Trischler, A. TextWorld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with IJCAI 2018, Revised Selected Papers*, pp. 41–75. Springer, 2019.
- Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *Nature*, 590:580–586, 2021.
- Fikes, R. E. and Nilsson, N. J. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.
- Givan, R., Dean, T., and Greig, M. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.
- Hausknecht, M., Ammanabrolu, P., Côté, M.-A., and Yuan, X. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7903–7910, 2020.
- Hoffmann, J., Porteous, J., and Sebastia, L. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.
- Jerz, D. G. Somewhere nearby is colossal cave: Examining will crowther’s original “adventure” in code and in Kentucky. *Digital Humanities Quarterly*, 1(2), 2007.
- Jiang, X., Lim, L.-H., Yao, Y., and Ye, Y. Statistical ranking and combinatorial Hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.
- Klyubin, A. S., Polani, D., and Nehaniv, C. L. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135, 2005.

- Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics*, 2006.
- Nachum, O., Gu, S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems 31*, 2018.
- Newell, A. and Simon, H. A. GPS, a program that simulates human thought. In Feigenbaum, E. A. and Feldman, J. (eds.), *Computers and Thought*, pp. 279–293. McGraw-Hill, 1963.
- Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pp. 278–287, 1999.
- Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2778–2787, 2017.
- Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 5062–5071, 2019.
- Pearl, J. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.
- Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992, 2019.
- Salge, C., Glackin, C., and Polani, D. Empowerment: An introduction. In Prokopenko, M. (ed.), *Guided Self-Organization: Inception*, pp. 67–111. Springer, 2014.
- Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.
- Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.
- Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.
- Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33*, 2020.
- Yao, S., Rao, R., Hausknecht, M., and Narasimhan, K. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 8736–8754, 2020.

A Bench index

Table 3: Every measured claim maps to one script in `code/`; each script prints its pre-registered hypotheses, its controls, and its verdict, and is runnable as `python bench_<name>.py` from the `code/` directory.

bench	script	paper claim
TOUCH-VS-READ	<code>bench_touch_vs_read.py</code>	prose underdetermines affordances; <code>do()</code> exact (§3.1)
EXPLORE-PLAN	<code>bench_explore_plan.py</code>	coverage works; planning depth adds nothing (§3.2)
GOAL-VALUE	<code>bench_goal_value.py</code>	bounded planning null; learned value bootstraps (§4)
STATE-LIFT-VALUE	<code>bench_state_lift_value.py</code>	the lift closes exploitation, 1.00 vs 0.00 (§5)
LIFT-KILLS-CURL	<code>bench_lift_kills_curl.py</code>	non-Markov = curled value flow; the lift removes it (§5)
INTRINSIC-MYOPIC	<code>bench_intrinsic_myopic.py</code>	myopic state-change bonus: null (§6)
INTRINSIC-PLANNED	<code>bench_intrinsic_planned.py</code>	propagated state-change bonus: null (§6)
BACKWARD-CHAINING	<code>bench_backward_chaining.py</code>	structural reasoning cracks the precondition, 6/6 vs 0/6 (§6.2)
SUBGOAL-NAVIGATION	<code>bench_subgoal_navigation.py</code>	means-ends navigation frees the snake, 4/4 (§7)
CAPACITY-TAKE	<code>bench_capacity_take.py</code>	the final take is a capacity limit, 4/4 vs 0/4 (§7)
CODE-CHAIN	<code>bench_code_chain.py</code>	the demarcation on code; dag flat at 1.00 (§6.3)
CODE-CHAIN-LATENT	<code>bench_code_chain_latent.py</code>	holds under latent structure + deceptive subgoals (§6.3)
CODE-CHAIN-PYTEST	<code>bench_code_chain_pytest.py</code>	holds with a real <code>pytest</code> toucher (§6.3)

B The shared game harness

All game benches import one module, `adventure_world.py`, which contains no puzzle knowledge. Its contents: (i) the canonical opening (enter the lit cave with the standard starting gear; picking up the lamp is itself in the action space, pre-done only to start inside the cave); (ii) the intervention primitives: `_verify` (execute a command on a deep copy; return the copy, whether the room changed, and the score delta, or nothing if invalid or lethal), `_acquires` (did a command bring a target object into the inventory, on a copy), `_discover_negative_precondition` (`do()` drop each held item and re-test a failing acquisition; the unique enabling item is the discovered negative precondition), `_unblocks` (after an interaction, does a previously blocked exit now open: a two-step probe on copies), and `_capacity_acquire_plan` (the same toggling logic, classifying multi-enabler gates as capacity limits); (iii) the lifted-state machinery: flags of the form `hold:<item>` and `prop:<object>`, a state key (room, flag values), conflict-driven flag admission (a flag is admitted only when an intervention or an observed change shows it flips an outcome), and tabular value iteration with R-max optimism over the learned lifted transition graph; (iv) the isolation scaffolds: the forced start at the bird room, and, for the state-representation benches only, the forced navigation skeleton whose deep leg stays gated by the real game state. The action space given to generic agents is the game parser’s verbs crossed with accessible objects plus the eight directions, capped and shuffled per step; no verb list encodes a solution.