WHICH PREDICTIONS SUFFICE FOR A WORLD-MODEL PLANNER'S DECISION? CERTIFYING REWARD AND VALUE EVIDENCE BY RE-PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

World-model planners allocate search or optimization among actions using predicted rewards and values. We study the minimum reward and value evidence sufficient to preserve a decision's action and preference statistics under native re-execution. Two challenges arise: (1) *Frozen computation*. Scoring recorded computation omits the planner's adaptation to replaced predictions. (2) *Single-set explanation*. One passing set leaves minimum size, alternatives, shared evidence, and reliability across planning streams unresolved. We introduce FutureCert, a sufficiency-certification framework whose interventions govern recorded and newly visited prediction read sites. Exact auditing recovers the minimum size, all minimum certificates, and their core; budgeted search and constrained repair return verified sets. Certificates are execution-conditioned, while independent calibration certifies cross-stream reliability with a simultaneous finite-sample guarantee. Simple constructions show that the original transcript alone cannot generally decide sufficiency and that execution-conditioned and stream-reliable minimum sizes can differ by $\Theta(H)$. Across four planners and six tasks, native re-execution rejects 16.5% of explanations selected on frozen search trees, mostly for preference rather than action changes, and 78.9% on TD-MPC2, where frozen candidate pools leave only 4 of 12 evidence units able to affect the output. At a 384-query cap, default FutureCert recovers the exact minimum size on 93.0% of TD-MPC2 decisions, compared with 86.5% for its Beam-4 variant. Exact audits expose alternative support and repair opportunities, while stream calibration certifies non-full sets on 48 of 50 Finger Spin inputs but on at most 1 of 50 inputs for each of Cheetah Run and Cup Catch.

1 INTRODUCTION

World models let an agent compare possible futures before it acts (Schrittwieser et al., 2020; Ye et al., 2021; Hansen et al., 2024). Their predicted rewards and values guide Monte Carlo tree search in MuZero-style agents (Browne et al., 2012; Hubert et al., 2021; Niu et al., 2023) and trajectory optimization in TD-MPC2 (Hansen et al., 2024; Williams et al., 2018). Explaining such a decision means identifying *which predictions suffice to support it*: which can be replaced while preserving the action and its preference statistics, which admit substitutes, and which remain sufficient across planning streams. These questions matter when an agent acts surprisingly, when alternative supporting evidence must be compared, and when some predictions become unavailable.

Existing approaches decompose rewards (Lin et al., 2019), explain predicted futures (Lin et al., 2020; Singh et al., 2025; Wenninghoff & Schwammberger, 2025), or assign feature importance through LIME (Ribeiro et al., 2016), KernelSHAP (Lundberg & Lee, 2017), RISE (Petsiuk et al., 2018), and temporal attribution. In adaptive planning, however, the predictions used to choose an action also determine which predictions are queried next.

Adapting attribution methods to this setting raises two challenges (Adebayo et al., 2018; Hooker et al., 2019; Atrey et al., 2019; Lam et al., 2022) (Fig. 1). (1) *Frozen computation*. Scoring a recorded tree or a fixed collection of candidate trajectories holds the search computation unchanged while altering its predictions. The native planner can respond differently: replacing a tree value can

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

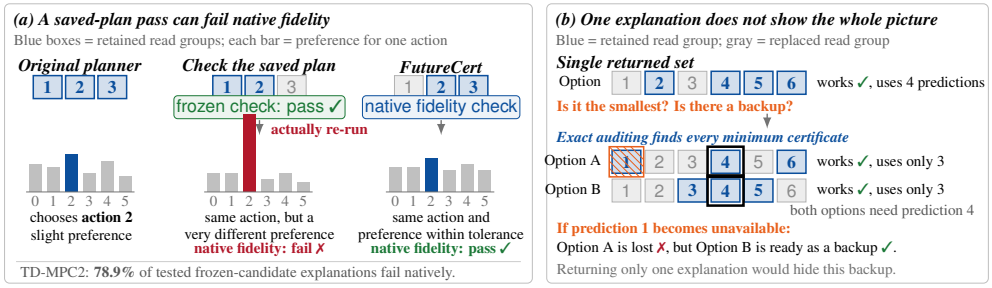


Figure 1: **Why planner explanations require re-planning and multiple certificates.** (a) A MuZero Pong attribution-selected set keeps action 2 under re-planning but changes its preference statistics. FutureCert returns a set preserving both. (b) Two minimum certificates share prediction 4; if prediction 1 becomes unavailable, Option B provides a same-size substitute.

redirect the next expansion, and replacing a trajectory reward can change elites and the trajectories sampled in later iterations. The intervention can therefore create prediction queries absent from the original record (Conmy et al., 2023; Lam et al., 2022; 2021). A set accepted on frozen computation may fail under re-planning. Even when the selected action survives, the preference statistics supporting that action can change, as in Fig. 1(a). (2) *Single-set explanation*. One passing set establishes sufficiency but leaves its structure unresolved. Is the set as small as possible? Are there equally small alternatives, and which predictions do they share? If a retained prediction becomes unavailable, can another set support the same decision? Figure 1(b) illustrates a same-size substitute when prediction 1 becomes unavailable. These questions call for minimum size, the family of minimum certificates, and their shared evidence (Ignatiev et al., 2019; Janota et al., 2015). Planning randomness introduces a further distinction: a set sufficient for one execution need not remain sufficient on another stream (Wang et al., 2021; Ribeiro et al., 2018).

We introduce FutureCert, which formulates planner explanation as an *execution-conditioned audit* over prediction read sites. Each evidence unit supplies fresh predictions when retained and fixed references when removed, including at newly visited sites. *To address frozen computation*, the planner restarts from the same input, warm start, budget, and random stream, adapts freely, and is checked against its original action and preference statistics. A two-oracle construction shows that the original transcript alone cannot generally decide this sufficiency: two prediction oracles can agree on the recorded computation yet disagree at a site first visited after intervention. *To address single-set explanation*, exact auditing recovers minimum size $k^\star$, all minimum certificates $\mathcal{F}^\star$, and their shared *Core*. This structure distinguishes replaceable predictions from those shared by every minimum certificate and supports repair when evidence becomes unavailable. Budgeted search uses the same verifier to recover certificates without enumerating the full search space. For reliability across streams, we freeze candidate sets and calibrate them on independent planning streams with simultaneous finite-sample guarantees (Clopper & Pearson, 1934; Angelopoulos et al., 2025). For H evidence units, a $\Theta(H)$ separation between execution-conditioned and stream-reliable minimum sizes shows why the two notions require separate treatment.

Across four planners and six tasks, native re-execution rejects 16.5% of explanations selected on frozen search trees, mostly for preference rather than action changes, and 78.9% on TD-MPC2, where frozen candidate pools leave only 4 of 12 evidence units able to affect the output. Budgeted FutureCert recovers the exact minimum size on 93.0% of TD-MPC2 decisions. Exact audits expose alternative certificates and same-size replacements for unavailable predictions. Cross-stream experiments separate discovery-stream requirements from independently calibrated reliability.

Our contributions are threefold:

- We formulate minimum sufficient reward and value evidence for adaptive world-model planners, with native interventions at recorded and newly visited read sites.
- We introduce FutureCert, a common verification framework for exact minimum-family auditing, budgeted certificate recovery and repair, and independent cross-stream calibration.
- Experiments on four planners and six tasks quantify frozen-interface failures, minimum-size recovery, alternative support, and task-dependent stream reliability.

2 Related Work

Attribution methods. LIME (Ribeiro et al., 2016), KernelSHAP (Lundberg & Lee, 2017), and RISE (Petsiuk et al., 2018) estimate feature importance using perturbations or random masks (Covert et al., 2021), while Integrated Gradients (Sundararajan et al., 2017) integrates gradients along a path. Time-series explainers assign temporal or window importance, or learn masks and segment selectors for a fixed predictor (Crabbé & Van Der Schaar, 2021; Bento et al., 2021; Leung et al., 2021; Jang et al., 2025; Liu et al., 2024; Kim et al., 2026; Yue et al., 2025). These scores or subsets (Carter et al., 2019; Chen et al., 2018) can propose evidence to the native verifier. FutureCert then audits minimum-set structure and reliability across planning streams.

Explaining RL and world-model agents. DRDRL decomposes value into interpretable reward channels (Lin et al., 2019); contrastive explanations use embedded self-predictions to explain one action relative to another (Lin et al., 2020); Singh et al. generate explanations from a world model (Singh et al., 2025); and Wenninghoff et al. compare DreamerV3's imagined trajectories with observations to diagnose failures (Wenninghoff & Schwammberger, 2025). SMDL selects compact explanatory regions through submodular optimization (Chen et al., 2024), while VPS selects evidence from confidence changes under retention and deletion (Chen et al., 2025). These approaches target policies, values, or trajectories; we audit the planner's adaptive computation.

Intervention-based evaluation and minimal sufficiency. ROAR (Hooker et al., 2019), sanity checks (Adebayo et al., 2018), counterfactual saliency analysis (Atrey et al., 2019), and Planner CheckList (Lam et al., 2022) evaluate explanations through retraining, model randomization, intervention, or search invariances. Activation patching likewise evaluates behavior by replacing internal activations and re-executing a model (Conmy et al., 2023). Abductive explanations (Ignatiev et al., 2019) distinguish minimum from subset-minimal sets and compute shared backbones (Janota et al., 2015). Anchors (Ribeiro et al., 2018) and probabilistic sufficient explanations (Wang et al., 2021) define sufficiency over input distributions (Dasgupta et al., 2022). FutureCert combines native planner interventions with minimum-set structure and reliability across planning streams. Sufficiency is relative to declared replacements rather than every completion, and adaptive execution makes validity nonmonotone.

3 Problem Setup

Planner and read sites. For input x, parameters θ, planning budget b, and random stream ξ, the planner returns $y^\star_\xi = P(x;\theta,b,\xi) = (a^\star_\xi, p^\star_\xi)$. Here $a^\star_\xi$ is the action, while $p^\star_\xi$ contains the declared preference statistics, namely the root-visit distribution in tree search or the first-step optimized action mean in trajectory optimization. A *read site* is any point where the planner consumes a predicted reward or value. Earlier reads determine later sites. A value read can determine the next expanded node, and a reward read can determine which candidates the optimizer retains.

Evidence units. An *evidence partition* $\mathcal{E}_x$ assigns every eligible read site, including sites absent from the original execution, to one of H units. A tree-search unit groups reward/value reads at one depth; a trajectory-optimization unit groups one quantity over an optimizer stage. Certificates describe these read groups, not individual branches or futures.

Definition 3.1 (Audit specification)**.** An audit specification $\Gamma = (\mathcal{E}_x, r, \Phi, \mathcal{I}, \mathcal{Q})$ fixes the evidence partition $\mathcal{E}_x$; a replacement rule r supplying fixed reference $\bar{z}_j$ at every removed-unit read; a fidelity predicate Φ over action and preference statistics, including tolerances; the executable implementation $\mathcal{I}$; and stream distribution $\mathcal{Q}$. References use separate episodes.

Changing the partition, references, tolerances, implementation, or stream distribution changes Γ and can change $k^\star$. FP32 and constant-free arithmetic are distinct executable specifications. Latent dynamics and policy priors remain unchanged. An *evidence unit* is a declared read group; a *mask* represents the retained set S, and $H = |\mathcal{E}_x|$ depends on the input.

Sufficiency check and certificate. For a retained set $S \subseteq \mathcal{E}_x$, native re-execution under the replacement rule gives

$$y_{S,\xi} = P_S(x;\theta,b,\xi,\bar{z}), \qquad C_{\Gamma,\xi}(S) = \Phi(y_{S,\xi}, y^\star_\xi) \in \{0,1\}. \tag{1}$$

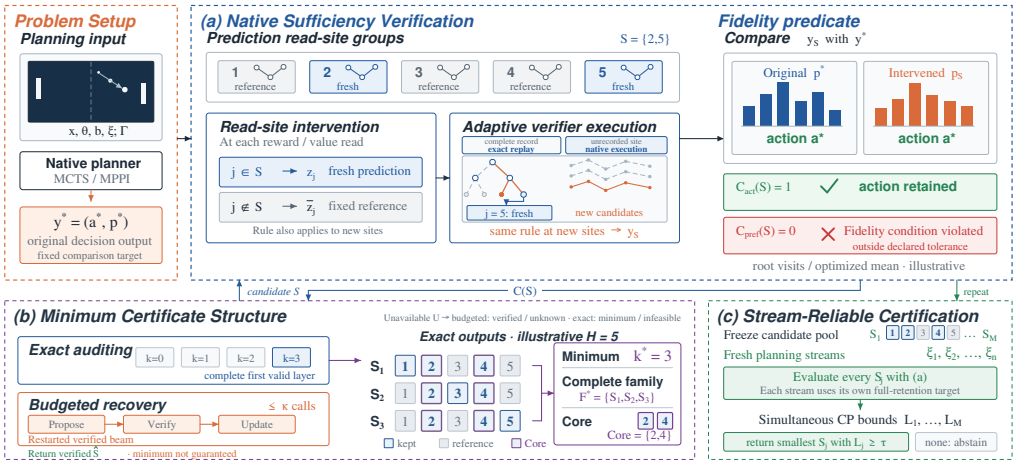


Figure 2: **The framework of FutureCert.** (a) *Native Sufficiency Verification*. Read-site rules supply fresh predictions or fixed references, including at newly visited sites. A record check reuses the transcript only when every prediction the intervened search requests is recorded; otherwise the planner is re-executed natively. (b) *Minimum Certificate Structure*. Exact auditing recovers the minimum size, complete minimum family, and Core. Budgeted recovery and repair query the same verifier. (c) *Stream-Reliable Certification*. Frozen candidates are evaluated on independent planning streams and certified by simultaneous lower bounds.

Retained units use predictions produced during re-execution, while removed units receive their references at every visited site. A set S is an *execution-conditioned certificate*, or verified sufficient set, when $C_{\Gamma,\xi}(S) = 1$ for the fixed specification and stream. The visited sites depend on S, so the check follows the intervened planner's adaptive computation. It need not be monotone: adding a unit can redirect execution and invalidate a certificate. Units outside S are therefore replaceable under this joint intervention, not individually irrelevant.

Native sufficiency includes effects mediated by re-planning; frozen evaluation holds the recorded tree or candidate pool fixed. We target adaptive execution because the deployed planner responds to its predictions. A certificate preserves the decision under this adaptation, possibly along a different search path (Appendix J). It applies to the declared execution; Section 4.3 adds an independently calibrated guarantee across planning streams.

What we ask of a certificate. A single certificate establishes only that one set suffices. We also seek the minimum size

$$k^\star = \min_{S \subseteq \mathcal{E}_x:\, C_{\Gamma,\xi}(S)=1} |S|, \tag{2}$$

the family $\mathcal{F}^\star$ of all certificates of size $k^\star$, and their intersection, the *Core*. These objects identify equally small alternatives and shared evidence. For a set chosen before ξ is drawn, we study its reliability $\Pr_{\xi\sim\mathcal{Q}}[C_{\Gamma,\xi}(S) = 1]$ on a fresh planning stream. Section 4 implements the check, computes the certificate structure, and lower bounds reliability through the same native re-execution.

4 FutureCert

FutureCert separates verification from subset search. The native verifier $C_{\Gamma,\xi}(S)$ supplies exact small-space audits, checks budgeted recovery and repair, and evaluates frozen candidates for independent stream calibration (Fig. 2). Appendix B gives the complete procedure.

A verified set establishes sufficiency for fixed Γ and ξ. Exact auditing additionally proves minimum cardinality and recovers the complete minimum family; Core describes evidence shared by this family rather than all valid sets. Budgeted recovery returns a verified set. Its exact-size recovery is scored against an independent exhaustive audit, separately from online optimality proofs.

4.1 Native Sufficiency Verification

Read rule. The intervention changes reward and value reads while leaving latent dynamics and the policy prior unchanged. A read in unit j is served by

$$\tilde{z}_j(S) = \begin{cases} z_j, & j \in S, \\ \bar{z}_j, & j \notin S, \end{cases} \qquad y_{S,\xi} = P_S(x; \theta, b, \xi, \bar{z}), \tag{3}$$

where z_j is produced during the intervened execution. The input, warm start, budget, and random stream are restored. Computation that consumes randomness still runs at replaced sites, preserving stream alignment while allowing the execution path to change.

Replacing a tree value can redirect expansion; replacing a reward in model predictive path integral (MPPI) optimization can change elites and later trajectories. The rule covers new sites. Full retention must reproduce $y^\star_\xi$ bitwise before any audit.

Fidelity predicate. The verifier operates at two levels,

$$C_{\text{act}}(S) = \mathbf{1}[\mathcal{A}_{\epsilon_a}(a_S, a^\star)], \qquad C_{\text{pref}}(S) = C_{\text{act}}(S) \wedge \mathbf{1}\left[\mathcal{V}_{\epsilon_p}(p_S, p^\star)\right]. \tag{4}$$

For tree search, $\mathcal{A}$ requires the unique most-visited root action to remain unchanged. The preference check requires the Spearman correlation of the root-visit distributions to be at least $1 - \epsilon_\rho$ and the normalized visit-margin error to be at most ϵ_m. For TD-MPC2, the two checks apply L_∞ tolerances to the executed action and the first-step optimized mean μ_0. These tolerances belong to Γ and are specified in Section 5.1. Neither check requires the intervened run to follow the original path. We use $C = C_{\text{pref}}$ unless specified.

Proposition 1 (Fixed-trajectory insufficiency)**.** *Two prediction oracles can produce identical original transcripts yet opposite native sufficiency judgments for the same retained set, so the transcript alone cannot decide sufficiency in general (construction in Appendix K.1.1).*

When the record suffices. For deterministic tree search, replay runs the planner's own selection, expansion, and backup with the same seed, serving recorded predictions or references at each read. If every requested node is recorded, replay equals native re-execution even when expansion order changes (Proposition 9, Appendix K.5). It calls no model. Unrecorded nodes trigger re-execution.

4.2 Minimum Certificate Structure

Nonmonotone validity. Adding a unit can redirect computation or move preference statistics beyond tolerance. A certificate's superset can therefore fail. Likewise, a set for which every one-unit deletion fails need not be minimum: a smaller, non-nested set may pass (Zeller & Hildebrandt, 2002). Local deletion can stop above the minimum, motivating cardinality-layer auditing and searching over alternative combinations. Exact auditing evaluates each set without monotone pruning.

Exact structure. For fixed Γ and ξ, define

$$\mathcal{F}^\star = \{S : C(S) = 1,\ |S| = k^\star\}, \qquad \mathcal{K} = \text{Core} = \bigcap_{S \in \mathcal{F}^\star} S. \tag{5}$$

When 2^H is enumerable, exact auditing completes the first nonempty cardinality layer, proving $k^\star$ and returning $\mathcal{F}^\star$ and $\mathcal{K}$. The family lists equally small alternatives and their Core.

Budgeted recovery. Verified beam search uses R restarts of q fresh queries with perturbed ranks and a shared within-input cache. Unit index or a development prior orders proposals; the native predicate determines acceptance. Restarts share a full-set incumbent, updating it only to smaller passing sets. We score exact-size recovery against offline enumeration (Section 5.3).

Re-solving under unavailable evidence. When units U are unavailable, repair searches only among the remaining units,

$$k^\star_{\neg U} = \min_{S \subseteq \mathcal{E}_x \setminus U:\, C(S)=1} |S|, \qquad k^\star_{\neg U} = +\infty \text{ if no such set exists.} \tag{6}$$

The search first checks members of $\mathcal{F}^\star$ that avoid U and then considers larger sets. A budgeted search returns a checked set or *unknown*; only exhaustive search proves infeasibility.

4.3 STREAM-RELIABLE CERTIFICATION

Single stream versus streams. A certificate found for one stream ξ is *execution-conditioned*. For a set selected before $\xi \sim \mathcal{Q}$ is drawn, define

$$p_\Gamma(S) = \Pr_{\xi\sim\mathcal{Q}}[C_{\Gamma,\xi}(S) = 1], \qquad k^\star_\tau = \min\{|S| : p_\Gamma(S) \geq \tau\}. \tag{7}$$

Each stream is compared with its own full-retention target. Write $k^\star_{\text{path}}(\xi)$ for the single-stream minimum.

Proposition 2 (Pathwise-stream separation)**.** *For every $H \geq 1$ and $\tau \in (0, 1]$, there exists a stochastic read-site planner with H units such that $k^\star_{\text{path}}(\xi) = 1$ on every stream, while $k^\star_\tau = \lceil \tau H \rceil$.*

If the stream selects one necessary unit J uniformly, the singleton $\{J\}$ passes on that stream, while a set chosen before the stream is drawn passes with probability $|S|/H$ (Appendix K). Small certificates on individual streams therefore need not yield one small certificate that is reliable across streams.

Calibration. We freeze M candidates before calibration, draw n independent streams, and record m_j passes for each candidate. A one-sided Clopper-Pearson (CP) bound corrected across the candidate pool is

$$L_j = \text{Beta}^{-1}(\alpha/M;\, m_j, n - m_j + 1), \qquad L_j = 0 \text{ if } m_j = 0. \tag{8}$$

The procedure returns the smallest candidate satisfying $L_j \geq \tau$ and abstains if none does.

Proposition 3 (Simultaneous calibration)**.** *Suppose the candidates are frozen before calibration and the streams are i.i.d. from $\mathcal{Q}$. Conditional on the input, the probability that any certified candidate has true reliability below τ is at most α. Hence the size of the smallest certified candidate upper-bounds $k^\star_\tau$ with probability at least $1 - \alpha$.*

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Planners and tasks. The benchmark has 2,783 decisions across nine settings: MuZero Pong/Breakout and Sampled EfficientZero (SEZ) Pong; MuZero, SEZ, and EfficientZero (EZ) MsPacman; and TD-MPC2 Cheetah Run, Cup Catch, and Finger Spin. Tree planners use 50 simulations; TD-MPC2 uses horizon 3 and six MPPI iterations. Episode splits and additional cohorts are specified in Appendix A.2.

Evidence and fidelity. Units group tree reads by depth ($H = \min(h, 12)$) and TD-MPC2 predictions by optimizer stage ($H = 12$). Tree fidelity requires the same unique most-visited action, root-visit Spearman correlation ≥ 0.95, and normalized margin error (NME) ≤ 0.10; TD-MPC2 requires action and optimized-mean L_∞ errors ≤ 0.20. With four or six untied counts, the ranking criterion preserves the complete ordering. Margin effects depend on the visit-count range (Appendix B.1).

Methods and metrics. Solvers share the native verifier, full-set fallback, and 384-query cap. Default FutureCert uses four width-4, 96-query restarts and a frozen development prior; parentheses identify alternative solvers. Recovery measures attainment of the audited minimum; Gap is excess size. Queries count distinct non-full sets; offline preparation is separate (Appendix C.1).

Statistics. Results weight episodes and settings equally; default FutureCert averages ten solver seeds. Brackets give 95% hierarchical bootstrap confidence intervals (CI); SD denotes solver-seed variation. Paired comparisons resample matched inputs. Bold marks point estimates, not significance.

5.2 FROZEN INTERFACES MISJUDGE NATIVE SUFFICIENCY

Freezing TD-MPC2 candidate pools removes the feedback from early predictions to later candidates. Its final output then depends only on four of twelve units. Under this interface, 78.9% of submitted

Table 1: **Frozen-interface acceptance versus native verification.** False acceptance (%) of final sets from KernelSHAP, LIME, and RISE, averaged over methods; counts in parentheses. The frozen TD-MPC2 interface exposes only four final-stage units out of twelve. Refresh reruns selection with the indicated iterations unfrozen. Time is relative to native execution.

TD-MPC2 interface	Cheetah	Cup	Finger	All	Indep. Cup	Time
Frozen candidates	95.8	96.5	44.5	78.9 (1,421)	95.7	0.22×
Refresh last 2 iterations	92.2	94.3	40.2	75.6 (1,360)	92.7	0.94×
Refresh last 4 iterations	71.0	74.5	35.2	60.2 (1,084)	72.0	1.00×
Tree-search interface	**MZ Pong**	**MZ Brk.**	**SEZ Pong**	**All**		**Action failures**
Frozen search tree	8.0	18.3	23.2	16.5 (551)		57

MZ: MuZero; Brk.: Breakout. Native acceptance has zero false acceptance by definition. Tree percentages are episode-weighted; 551/3,282 is the pooled count, not the weighted percentage.

Table 2: **Recovery at an equal online-query cap.** (a) TD-MPC2 task-macro results at cap 384. Prep. includes full targets; amortized cost adds Prep./200. Default task totals are 17,045/18,679/15,418. (b) Prior ablation with four 96-query restarts and online ordering fixed: recovery means (solver-seed SD) over ten seeds and mean queries. Every returned set passes native verification.

(a) Solver	Recovery (%)	Gap	Online	Prep./task	Amortized
KernelSHAP	54.67	1.527	172.56	0	172.56
FutureCert (Enumeration)	81.83	1.407	135.33	0	135.33
FutureCert (Beam-4)	86.50	0.307	69.44	0	69.44
FutureCert (Surrogate-Search)	92.00	0.505	101.20	204,800	1,125.20
FutureCert	92.97	0.140	69.62	17,047.3	154.86

(b) Solver	Recovery (%)		Queries	
	Cheetah	Cup	Cheetah	Cup
FutureCert (No prior)	92.00 (0.62)	84.50 (1.62)	80.4	114.7
FutureCert	93.05 (1.01)	86.10 (1.29)	81.6	122.0

sets fail native verification (Table 1). This rate largely reflects what the frozen interface can express; the same attribution methods also support native queries. Partial refresh still leaves disagreement.

Among 3,200 preselected masks, native execution rejects 891/1,030 frozen accepts; the frozen interface also rejects 97 native-valid masks (Appendix D.1).

Frozen-tree evaluation yields 16.5% false acceptance [14.4, 18.6], mostly from preference changes, and agrees with native validity on 91.6-97.6% of empty/random probes (Appendix D.4). The three pipelines select identical sets on 908 of 1,094 inputs, so pooled submissions are not independent replicates. The exact record check completes on 109 non-full MuZero probes; all outputs match native execution, including 93 probes with reordered expansions (Appendix K.5). Completion permits a fidelity judgment, which may pass or fail.

5.3 Minimum-size Recovery and Cost

Default FutureCert attains 92.97% minimum-size recovery on TD-MPC2 after development-bank preparation (Table 2). It reduces Gap relative to the displayed search variants. Equal online caps do not imply equal total cost: on Cup Catch, the default also uses more online queries than FutureCert (Beam-4). Over N decisions, amortized queries are $q_{\text{online}} + Q_{\text{prep}}/N$ (Appendix F).

With four restarts and online ordering fixed, adding the prior increases recovery by 1.05/1.60 percentage points on Cheetah/Cup (Table 2b). With the prior fixed, four restarts instead of one add 4.55/6.10 points. Appendices F.2 and F.1 give the factorial study and paired Beam-4 intervals.

Core tree-search spaces are small: excluding empty/full minima, the default, FutureCert (Beam-4), and FutureCert (Enumeration) recover 95-100%. These settings primarily test the audit rather than separate solvers (Appendix F.9).

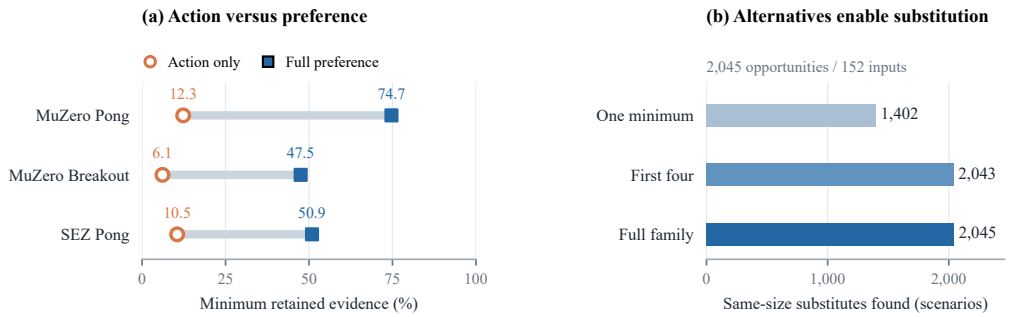


Figure 3: **What certificate structure reveals.** (a) Minimum retained fractions under action-only and preference fidelity. (b) Same-size substitutes using the first minimum, first four, or complete family: 2,045 opportunities among 3,112 unavailable-evidence scenarios on 152 multi-minimum inputs. Counts measure set availability.

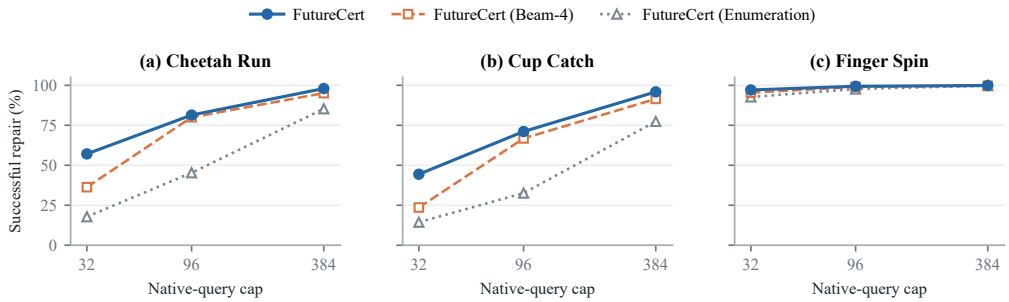


Figure 4: **Repair under evidence unavailability.** Success is finding a verified set avoiding unavailable units, conditional on feasibility. The study covers 46,800 scenarios on 600 TD-MPC2 inputs; the default averages ten seeds. Solvers receive neither $k^\star$, the family, nor Core. Appendix G.3 gives query counts and controls.

Transfer depends on cohort and query cap. On 50 Quadruped Run inputs at cap 384, the default retains 4.47 units. On a separate 100-input cohort at cap 96, FutureCert (Utility-96) retains 7.59 units versus 7.18 for FutureCert (Native greedy). These are verified sizes without exact minima; Appendix F.7 gives all transfer and larger-space results.

5.4 Certificate Structure and Repair

Replacing every reward/value read preserves the action on 72-79% of core tree-search decisions. Dynamics and learned policy priors remain active. Adding preference fidelity raises minimum retained fractions from 6.1-12.3% to 47.5-74.7% (Fig. 3a). The rank criterion also covers rarely visited root actions, whose reordering can cause preference failures (Appendix J). Uniform-prior controls sharply reduce empty-set action retention (Appendix E.1).

Multiple minima occur on 152 inputs. Four enumerated minima recover 2,043 of 2,045 same-size replacement opportunities, compared with 1,402 for one minimum. This prefix is already the complete family on 151/152 inputs. These are exact-audit outputs; budgeted recovery returns a verified set. Completing the family establishes all minimum alternatives and their Core. An unavailable non-Core unit can be avoided by another minimum; an unavailable Core unit belongs to every minimum, so repair must seek a larger certificate or establish infeasibility (Appendix G).

At 32 queries, default FutureCert improves feasible repair by 20.8 percentage points over FutureCert (Beam-4) on Cheetah/Cup (Fig. 4). This measures search without the exact family. Budget exhaustion reports unknown; exhaustive checking establishes infeasibility.

5.5 Reliability across Planning Streams

Compact single-stream sets rarely yield certified non-full sets on Cheetah and Cup. With $\tau = 0.8$, $\alpha = 0.05$, and 64 calibration streams, the 50-input cohorts yield 1, 0, and 48 non-full certifications on Cheetah, Cup, and Finger (Appendix H.3). These guarantees are per input and frozen candidate pool under the declared stream distribution.

Table 3: **Fixed-set transfer and re-solved structure.** Reference tests use 16 bootstrap draws on fixed partial tree-search certificates. Stream/arithmetic tests re-enumerate 34 Cheetah and 33 Cup inputs. A stable minimum size does not establish transfer of a particular set.

Reference change	MuZero Pong	MuZero Breakout	SEZ Pong
Fixed set preserves action (%)	96.7	97.6	95.1
Fixed set preserves C_{pref} (%)	64.4	57.8	67.0
Re-solved structure	**Cheetah Run**	**Cup Catch**	
$k^\star$ unchanged, new stream (%)	41.3	33.3	
$k^\star$ unchanged, FP32 $\rightarrow$ constant-free (%)	94.9	50.8	
Family Jaccard, FP32 $\rightarrow$ constant-free	0.859	0.179	

Exhaustive discovery on six diagnostic Cheetah/Cup inputs separates empirical feasibility from certification. Five need 10-12 units to pass at least 13 of 16 discovery streams. On Cheetah-43, a size-1 set passes 13/16 discovery and 164/200 independent streams, but its simultaneous lower confidence bound (LCB) is 0.730. Only full retention is certified in these six pools (Appendix H.2). Exhaustive discovery establishes minimum size for the finite-stream predicate; calibration supplies population-reliability bounds for frozen candidates. Non-full abstention reflects candidate reliability or finite-sample uncertainty. Candidates outside the frozen pool require separate evaluation.

5.6 Stability across Audit Conditions

Action preservation transfers across references more often than preference fidelity; re-solved families change with streams or arithmetic (Table 3). Fixed-set transfer and re-solved structural agreement measure different properties. Cup Catch illustrates that certificate composition can be implementation-sensitive even when inputs and streams are paired. Shared TD-MPC2 reference shifts cancel in exact arithmetic (Proposition 7); their observed FP32 effects arise from numerical amplification, especially on Cup Catch (Appendix I.2). Exactness is conditional on Γ and ξ. Appendix I.1 separates these factors; Appendix M plots tolerance and budget effects.

5.7 Deployment and Diagnosis

Predicted masks with full-planning fallback skip 50.1% of Finger Spin network forwards, with paired return difference -30.9 $[-37.1, -24.6]$ over 20 episodes; Cheetah and Cup mostly retain full planning (Appendix L.4). These masks are predicted before planning and require a separate check for fidelity on each new input. Family/Core features do not improve area under the precision-recall curve beyond inexpensive diagnostic probes (Appendix L).

6 Conclusion

Which reward and value predictions suffice for a planner's decision depends on how the planner adapts when those predictions are replaced. FutureCert makes this adaptation part of the sufficiency check and uses one verifier for exact auditing, budgeted recovery, repair, and independent stream calibration. The audits reveal distinct roles for prediction evidence: actions often survive its removal while preference statistics change, and alternative minimum sets provide substitutes when evidence becomes unavailable. Complete families establish which alternatives exist and what they share; budgeted search recovers verified sets under fixed query budgets. Cross-stream results further separate compact support for one execution from evidence that remains sufficient across executions. These certificates describe decision support under declared interventions and execution conditions.

Limitations. Exact family/Core audits primarily cover $H \leq 12$; a complete 12-unit audit requires 4,096 verifier queries, with additional runs for calibration. Larger-space studies have partial or no exact ground truth (Appendix F.7). Units group reward/value reads while dynamics and policy priors remain active. Certificates depend on granularity, references, tolerances, arithmetic, and streams; fixed replacement tests one completion rather than all assignments to removed units. Human usefulness remains unmeasured, and deployment measures a computation-return trade-off.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs and to implement the methods evaluated in our experiments. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, or interpret results; translation, dataset cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit experiment code and to edit the manuscript to improve readability. We have reviewed all AI-assisted work: every AI-assisted proof was checked step by step by the authors; all AI-generated code was reviewed and tested for correctness by two authors, and all reported results come from experiments run and logged by the authors; all AI-edited text was checked to ensure it preserves the intended meaning and claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The audit specification, evidence partitions, reference construction, fidelity tolerances, planner configurations, and seed splits are described in Section 5.1 and Appendix A; assumptions and proof locations are indexed in Table 76. The core implementation and scripts for the main experiments are available for review at https://anonymous.4open.science/r/FutureCert-88E5/. The complete code, including all baselines and experimental artifacts, will be publicly released upon acceptance.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743*, 2019.

João Bento, Pedro Saleiro, André F Cruz, Mário AT Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2565–2573, 2021.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 567–576. PMLR, 2019.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International conference on machine learning*, pp. 883–892. PMLR, 2018.

Ruoyu Chen, Hua Zhang, Siyuan Liang, Jingzhi Li, and Xiaochun Cao. Less is more: Fewer interpretable region via submodular subset selection. In *International Conference on Learning Representations*, volume 2024, pp. 18280–18302, 2024.

Ruoyu Chen, Siyuan Liang, Jingzhi Li, Shiming Liu, Maosen Li, Zhen Huang, Hua Zhang, and Xiaochun Cao. Interpreting object-level foundation models via visual precision search. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30042–30052. IEEE, 2025.

Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Jonathan Crabbé and Mihaela Van Der Schaar. Explaining time series predictions with dynamic masks. In *International conference on machine learning*, pp. 2166–2177. PMLR, 2021.

Sanjoy Dasgupta, Nave Frost, and Michal Moshkovitz. Framework for evaluating faithfulness of local explanations. In *International Conference on Machine Learning*, pp. 4794–4815. PMLR, 2022.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pp. 4476–4486. PMLR, 2021.

Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. Abduction-based explanations for machine learning models. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 1511–1519. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33011511. URL https://doi.org/10.1609/aaai.v33i01.33011511.

Hyeongwon Jang, Changhun Kim, and Eunho Yang. Timing: Temporality-aware integrated gradients for time series explanation. *arXiv preprint arXiv:2506.05035*, 2025.

Mikoláš Janota, Inês Lynce, and Joao Marques-Silva. Algorithms for computing backbones of propositional formulae. *Ai Communications*, 28(2):161–177, 2015.

Hwijin Kim, Jaeho Kim, and Changhee Lee. Timeseg: An information-theoretic segment-wise explainer for time-series predictions. In *International Conference on Learning Representations*, volume 2026, pp. 150988–151014, 2026.

Kin-Ho Lam, Zhengxian Lin, Jed Irvine, Jonathan Dodge, Zeyad T Shureih, Roli Khanna, Minsuk Kahng, and Alan Fern. Identifying reasoning flaws in planning-based rl using tree explanations. *arXiv preprint arXiv:2109.13978*, 2021.

Kin-Ho Lam, Delyar Tabatabai, Jed Irvine, Donald Bertucci, Anita Ruangrotsakun, Minsuk Kahng, and Alan Fern. Beyond value: Checklist for testing inferences in planning-based rl. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pp. 606–614, 2022.

Kin Kwan Leung, Clayton Rooke, Jonathan Smith, Saba Zuberi, and Maksims Volkovs. Temporal dependencies in feature importance for time series predictions. *arXiv preprint arXiv:2107.14317*, 2021.

Zhengxian Lin, Kim-Ho Lam, and Alan Fern. Contrastive explanations for reinforcement learning via embedded self predictions. *arXiv preprint arXiv:2010.05180*, 2020.

Zichuan Lin, Li Zhao, Derek Yang, Tao Qin, Tie-Yan Liu, and Guangwen Yang. Distributional reward decomposition for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Zichuan Liu, Tianchun Wang, Jimeng Shi, Xu Zheng, Zhuomin Chen, Lei Song, Wenqian Dong, Jayantha Obeysekera, Farhad Shirani, and Dongsheng Luo. TimeX++: Learning time-series explanations with information bottleneck. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32062–32082. PMLR, 2024. URL https://proceedings.mlr.press/v235/liu24bl.html.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Yazhe Niu, Yuan Pu, Zhenjie Yang, Xueyan Li, Tong Zhou, Jiyuan Ren, Shuai Hu, Hongsheng Li, and Yu Liu. Lightzero: A unified benchmark for monte carlo tree search in general sequential decision scenarios. *Advances in Neural Information Processing Systems*, 36:37594–37635, 2023.

Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 1527–1535. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11491. URL https://doi.org/10.1609/aaai.v32i1.11491.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Madhuri Singh, Amal Alabdulkarim, Gennie Mansi, and Mark O Riedl. Explainable reinforcement learning agents using world models. *arXiv preprint arXiv:2505.08073*, 2025.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Eric Wang, Pasha Khosravi, and Guy Van den Broeck. Probabilistic sufficient explanations. *arXiv preprint arXiv:2105.10118*, 2021.

Nils Wenninghoff and Maike Schwammberger. Interpretable world model imaginations as deep reinforcement learning explanation. In *World Conference on Explainable Artificial Intelligence*, pp. 140–161. Springer, 2025.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Jinghang Yue, Jing Wang, Lu Zhang, Shuo Zhang, Da Li, Zhaoyang Ma, and Youfang Lin. Optimal information retention for time-series explanations. In *Forty-second International Conference on Machine Learning*, 2025.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on software engineering*, 28(2):183–200, 2002.

APPENDIX

NOTATION

Symbol	Meaning
x, θ, b	Planning input, model parameters, and planning budget.
$\xi, \mathcal{Q}$	Realized planning stream and its declared distribution. A seed initializes a stream; solver seeds control a separate source of randomness.
Γ	Audit specification: evidence partition, reference construction, predicate, and executable numerical configuration.
$\mathcal{E}_x, H$	Evidence units for input x and their count $H = \lvert\mathcal{E}_x\rvert$.
j, S	Public unit index and retained set. A binary mask represents the same set S.
$z_j, \bar{z}_j$	Fresh prediction at a read in unit j and its fixed replacement value.
$y^\star_\xi, y_{S,\xi}$	Full-retention and intervened outputs on the same stream.
$C_{\Gamma,\xi}$	Native sufficiency predicate; action and preference versions are C_{act} and C_{pref}.
$k^\star, \mathcal{F}^\star, \mathcal{K}$	Minimum size, complete minimum family, and Core (the family's intersection).
B	Prescribed set size in fixed-size comparisons, distinct from the minimum $k^\star$.
R, q	Solver restart count and fresh-query allowance per restart.
$p_\Gamma(S), k^\star_\tau$	Stream pass probability and minimum size attaining reliability τ.
M, n, m_j, L_j	Frozen candidate count, calibration streams, passes of candidate j, and its simultaneous lower confidence bound.
α, τ	Family-wise error level and required reliability.
$\mathcal{R}, \mathcal{R}_j$	Reference episodes, and episodes observing unit j; $\mathcal{R}$ denotes an execution record only in the record-replay proof.

Locally defined proof symbols are introduced with their roles: ρ is the complete random-generator state in the reuse proof, whereas ρ_{S_0} is the intersection-witness count. The fidelity correlation is written out as Spearman correlation. Model evaluations, logical subset queries, and wall-clock time are distinct cost units. The appendix follows the questions in the main text. Appendix D covers frozen interfaces; Appendix F covers recovery and cost; Appendix G covers structure and repair; Appendix H covers stream reliability; and Appendix I covers numerical stability. Protocols, solver implementations, and comparison methods are in Appendices A-C. Theory and diagnostic applications are in Appendices K and L. Appendix J presents individual cases; Appendix M collects tolerance and budget profiles and numerical controls.

A EXPERIMENTAL PROTOCOL AND REPRODUCIBILITY

This appendix specifies the planning tasks, intervention units, evaluation populations, and statistical aggregation. Table 4 locates the evidence for the main-text results.

Table 4: Main-text questions and supporting appendix sections.

Main text	Evidence	Appendix
Section 5.2	Frozen accepts, false rejections, partial refresh	D.1, D.4
Section 5.3	Recovery, cost, restarts, priors, and transfer	F.1, F.2, F, F.7
Section 5.4	Action versus preference, families, repair	E.1, G
Section 5.5	Compact certification and exhaustive stream cases	H.3, H.2
Section 5.6	Reference, stream, and arithmetic sensitivity	E.1, I.1
Section 5.7	Diagnosis and closed-loop control	L, L.4
Section 4	Proposition assumptions, proofs, and tests	Table 76

A.1 PLANNERS, TASKS, AND CHECKPOINTS

This appendix specifies the implementation behind the intervention audit, exact enumeration, and budgeted recovery experiments. The core tree-search and final TD-MPC2 recovery results use development-only prototype libraries. The MsPacman results use a separately frozen extension configuration; complete per-setting recovery results appear in Appendix F.1. Unless noted other-

wise, tree-search statistics use decisions with a unique deployed action, average within episode, and weight episodes equally. The core macro-average also weights the three settings equally.

The three core settings comprise 338 MuZero Pong, 373 MuZero Breakout, and 383 SEZ Pong decisions, totaling 1,094. The MsPacman extension has 1,089 decisions across MuZero, SEZ, and EfficientZero. Cheetah Run, Cup Catch, and Finger Spin each contain 200 TD-MPC2 decisions. Continuous-control units and tree-search depth units are defined separately; we do not pool their retained fractions.

Full planner parameters and execution checks are given in Appendix B.1.

A.2 Data Splits and Evaluation Populations

Table 5: Evaluation populations. Reused inputs and subsets are not additive.

Study	Population	Role
Core tree search	1,094 decisions	MuZero Pong 338, Breakout 373, SEZ Pong 383.
MsPacman extension	1,089 decisions	Three additional planner settings.
Core TD-MPC2	600 decisions	200 each on Cheetah, Cup, and Finger.
Main recovery total	2,783 decisions	Sum of the preceding three disjoint groups.
Record replay	1,062 decisions	MuZero subsets: Pong 338, Breakout 373, MsPacman 351.
Paired-mask interface audit	3,200 mask pairs	Eight preselected masks on each of 400 Cheetah/Cup inputs.
Frozen final sets	1,800 submissions	Three methods on the same 600 TD-MPC2 inputs.
Cross-stream certification	50 inputs per task	Three TD-MPC2 task cohorts, not 150 new tasks.
Stream/arithmetic structure	67 paired inputs	34 Cheetah and 33 Cup inputs, four streams each.
Current task transfer	50 inputs per new task	Ten fresh episodes; default four-restart solver.
Earlier task transfer	25 and 100 per task	Separate cohorts; FutureCert (Utility-96).

Development and evaluation data The core development sets contain 317, 388, and 388 decisions; evaluation sets contain 338, 373, and 383. Development data prepare the prototypes, router, and surrogate. Within a setting, all methods use the same evaluation reference library and intervention verifier.

Source isolation and reproducibility manifest Development, reference, and evaluation episode-seed sets are pairwise disjoint in every final core and TD-MPC2 setting. We also compare hashes of serialized planning inputs and full planner outputs across splits. TD-MPC2 has no cross-split input or output collisions. The deterministic Atari environments can repeat the same planning input under different seeds; these repetitions are logged explicitly and are not treated as independent evidence that the inputs differ. Recovery libraries are indexed only by the declared development rows, and the evaluation search interface reveals a mask label only after that mask has been proposed.

The independent-episode samples use MuZero Breakout seeds 36161-36170 and TD-MPC2 Cup Catch seeds 64001-64010, with ten decisions per episode. Breakout uses checkpoint SHA-256 `d5c3de235e9d...` and the byte-identical prior reference cache `8e892bafea33...`; Cup Catch uses checkpoint `0b58a9dfb1fb...` and development table `3d51cf7dd9be...`. FutureCert is re-evaluated on them at the 384-query cap (Appendix F.6). Repository scripts implement the collection-exact-audit-search-summary sequence. The machine-readable manifest records their exact paths and hashes together with commands, seeds, aggregation, and oracle-isolation rules before outcomes are summarized.

A.3 Evidence Units and Reference Construction

Tree-search evidence For an original decision with maximum expanded depth h_x, we use $H = \min(h_x, 12)$ evidence units. Depth d maps to $\min(d, H)$, so nodes reached deeper during re-execution still have a defined unit. A retained depth applies across all candidate branches; replacement acts on both reward and value predictions. The policy prior and latent dynamics retain

their original configuration. The planner recomputes selection, expansion, and backup after intervention.

References weight episodes equally. For reference-episode set $\mathcal{R}$, with predictions Z_{ej} observed in episode e at unit j, let

$$\bar{z}_j = \frac{1}{|\mathcal{R}_j|} \sum_{e \in \mathcal{R}_j} \frac{1}{|Z_{ej}|} \sum_{z \in Z_{ej}} z. \tag{9}$$

Here $\mathcal{R}_j$ includes episodes observing unit j. Reward and value references are computed separately and shared by candidates. Reference and evaluation episodes have disjoint seed sets.

Every native tree-search execution uses 50 simulations. Tree search disables root Dirichlet noise, takes the argmax of visit counts, and does not sample through a temperature. Fixed seeds reproduce random tie handling and SEZ action sampling; SEZ also checks the root-action-slot mapping. Full-retention consistency compares the entire root-visit vector, not only the action.

Continuous-control evidence TD-MPC2's horizon includes rewards at steps 0, 1, and 2 and a terminal value: four predicted quantities. Each execution runs six MPPI iterations with 512 sampled trajectories per iteration, 64 elites, and 24 policy trajectories. We divide the six iterations into three stages, iterations 1-2, 3-4, and 5-6. Public unit index $j = 3a + s + 1$ uses quantity index $a \in \{0, 1, 2, 3\}$ and stage index $s \in \{0, 1, 2\}$, giving units 1-12. Unit j covers all candidate-trajectory reads of quantity a in stage s. Zero-based archive indices equal $j - 1$.

Every subset runs all optimization iterations in order. We restore the warm-start mean, model parameters, and random seed each time. Computation that would consume randomness still runs at a replaced site, maintaining random-stream alignment. Full retention is checked coordinate by coordinate against both cached and native planner actions. Repeated subset re-executions check the action and first-step final weighted mean, and random-number-generator states are compared under full, empty, and singleton interventions. We enumerate all $2^{12} = 4,096$ subsets per evaluation decision.

The coarse-granularity control binds three stages of each predicted quantity, leaving four groups. Evaluation of a coarse set reads its corresponding 12-bit-mask result; decisions, references, and planner outputs are identical. A retained coarse group counts as three fine units when comparing quantities.

A.4 Statistical Analysis and Reproducibility

Core results average decisions within episodes and then weight episodes equally; the macro-average weights the three settings equally. Unless specified otherwise, confidence intervals use 2,000 hierarchical episode-to-decision bootstrap draws. Paired comparisons difference methods on the same decisions before resampling. The default-versus-Beam-4 comparison additionally resamples solver seeds and uses 3,000 draws (Table 39); the unavailable-evidence study uses 4,000 episode-block draws and separately reports input-block intervals (Appendix G.2). Fault repair first averages fault instances within a decision. Extension-edge and expandable-decision rates explicitly state their distinct denominators.

A.5 Artifact Manifest

Table 6 collects the identifiers stated throughout Appendices A-B. Each experiment stores a frozen protocol (inputs, seeds, configuration, and SHA-256 hashes of the scripts and archives it reads), written before its outcomes were summarized. These protocols will accompany the complete artifact release upon acceptance.

Table 6: **Artifact manifest.** Seed ranges, checkpoint and cache hashes (prefixes), and freeze points; full hashes are recorded in the protocol files.

Item	Value
FutureCert solver seeds	61831-61840 (retrospective results); 61841-61845 (post-freeze Hopper/Quadruped transfer)
Independent episodes	MuZero Breakout 36161-36170; TD-MPC2 Cup Catch 64001-64010 (ten decisions per episode)
New-task episodes	reference 74001-74005; evaluation 75001-75005 and 76001-76005; post-freeze transfer 79001-79010
Checkpoints and caches	Breakout checkpoint `d5c3de235e9d`, reference cache `8e892bafea33`; Cup Catch checkpoint `0b58a9dfb1fb`, development table `3d51cf7dd9be`
Freeze points	FutureCert configuration frozen before the post-freeze transfer episodes; calibration candidates frozen before calibration streams are drawn; the 30 constant-free inputs fixed before numerical confirmation; supplementary inputs, seeds, and reference groups frozen before execution
Pipeline	collection, exact audit, search, and summary scripts, one protocol file per experiment

B Algorithm and Implementation

All proposal methods submit sets to the same declared verifier. Algorithm 1 summarizes exact auditing, budgeted recovery, and calibration. The following sections specify native execution, search, and explanation-method adaptations.

Algorithm 1 FutureCert for exact auditing, budgeted recovery, and calibration

Require: planner P; (x, θ, b, ξ); specification Γ; restarts R; queries per restart q
$y^\star \leftarrow P(x; \theta, b, \xi)$; **assert** full retention reproduces $y^\star$
VERIFY(S): use the record if every requested prediction is recorded; otherwise restore $(x$, warm start, $b, \xi)$ and run P under Eq. (3); return $\Phi(y_S, y^\star)$
Exact audit
For $k = 0, 1, \ldots$, evaluate every S with $|S| = k$; stop at the first nonempty feasible layer F_k
$\mathcal{F}^\star \leftarrow F_k$; $\mathcal{K} \leftarrow \bigcap_{S \in \mathcal{F}^\star} S$
Budgeted recovery
$\hat{S} \leftarrow \mathcal{E}_x$; rank candidates by unit index or the development prior
for $r = 1, \ldots, R$ **do**
    Run a beam over perturbed ranks with at most q fresh queries
    For each proposed S, if VERIFY(S) and $|S| < |\hat{S}|$, set $\hat{S} \leftarrow S$
return $\hat{S}$
Calibration
Freeze $S_1, \ldots, S_M$; evaluate them on n fresh streams
Return the smallest S_j with $L_j \geq \tau$; otherwise abstain

B.1 Native Intervention and Verification

Every audited input first reproduces the unmodified planner at full retention. Tree-search checks include the complete root-visit vector and root action-slot mapping; TD-MPC2 checks the native and archived action coordinatewise. Repeated full, empty, and singleton interventions also check random-stream alignment. These execution checks precede subset interpretation; adapter details follow below, and Appendix D compares native and approximate interfaces.

Spearman correlation uses average ranks for ties. Action retention requires the deployed action to remain the unique maximum-visit action. Here q_S denotes the root visit-count vector after native re-execution, and the margin is $q_S(a^\star) - \max_{a \neq a^\star} q_S(a)$. The normalized error divides by the larger of one and the range of the original root visit-count vector. It is a planner-internal fidelity measure rather than an estimate of environmental return or action quality. Tree-search conditions are action; action plus ranking; action plus margin; and all three together. Decisions lacking a unique original maximum are logged separately and excluded from the core certificate analysis.

Visit-count margins are discrete. In the example with original range $12 - 5 = 7$, the cutoff $\mathrm{NME} \leq 0.10$ allows an absolute margin difference of at most 0.7, so integer margins must agree exactly. The threshold sweep in Table 32 re-evaluates the same extension edges at relaxed cutoffs; it does not redefine those edges or reselect certificates after seeing the outcome.

B.2 Exact Auditing and Certificate Outputs

The evidence units from Section 3 are the search objects. Exact auditing checks the entire first valid cardinality layer, while budgeted recovery returns a set validated by the same verifier and measures its gap against the exact audit.

We evaluate all subsets from cardinality zero upward and finish the first layer containing a valid set. This yields the exact minimum size and complete family. Empty and full certificates each have only one possible composition, so structure statistics also report the partial stratum $0 < k^\star < H$. When one representative is required, we first minimize the number of contiguous runs and then break ties lexicographically by depth. Uniqueness, family size, and Core are computed over the complete family.

For the capped first-valid baseline, we also freeze 100 independently shuffled orders within each cardinality layer before replaying the exact audit records. Their core macro recovery averages are 80.7%, 92.9%, and 97.3% at caps 16, 32, and 96; the corresponding 2.5th-97.5th percentile ranges are 80.3-81.3%, 92.4-93.4%, and 97.1-97.5%. The primary lexicographic order is near the high end at 96 calls; the shuffled results preserve its lower mean query cost and lower mean recovery than FutureCert.

A fixed-evidence comparison verifies the actual set returned by each method. Asking for exactly B units tests the cardinality-B layer; $\Pr(k^\star \leq B)$ asks whether any valid set of size at most B exists. Because validity is nonmonotone, extending a smaller valid set still needs verification. The result tables distinguish these quantities.

B.3 Budgeted Certificate Recovery

The current default is stochastic verified beam with $R = 4$, width four, temperature two, prototype interval four, prior ordering, and a deterministic first restart. Each restart has an allowance of 96 fresh non-full queries; the global cap is 384, and repeated masks share a per-input native cache. Candidate ranks, rather than planner-dependent raw deficits, receive independent Gumbel perturbations. A queried candidate can update the incumbent only if it passes the native predicate and is smaller. Later restarts target strict improvements. They therefore neither guarantee distinct returned certificates nor recover a complete minimum family. The minimum observed valid size is the solver output for each input; it is not chosen with access to $k^\star$ or to test-set recovery scores.

Main retrospective results average ten complete FutureCert executions with seeds 61831-61840. Reported standard deviations measure variation across these executions after episode/task averaging. Confidence intervals jointly resample solver seeds, episodes, and within-episode decisions with paired comparator samples. Increasing the restart count from two to four or eight is a resource change: the respective caps are 192, 384, and 768. No IID-restart success formula is asserted, because the deterministic first restart, shared cache, and incumbent-dependent proposals make restarts dependent.

The FutureCert procedure preserves native validity and can only decrease the verified incumbent size as it proceeds; its fresh-query count is at most $96R$. These properties follow from the update and accounting rules, rather than from a new optimality theorem. An exact recovery rate compares outputs to a separate exhaustive audit; it does not mean that the online solver proved minimum cardinality.

FutureCert (Beam-4) uses the width-four implementation with its original stopping and deletion rules. We rerun it, the FutureCert (Native greedy) and enumeration variants, and KernelSHAP with the same total cap of 384. Default FutureCert uses its frozen development bank, and offline bank construction costs are separate from online calls. Development costs and older 96-call experiments remain explicitly identified below.

Prospective transfer freezes this FutureCert configuration before collecting ten new episodes per new task (seeds 79001-79010), samples five inputs per episode, and evaluates five new solver seeds (61841-61845). The bank is the existing frozen source-task bank, with no fitting on new-task outcomes. FutureCert (Beam-4), greedy, and KernelSHAP have the same cap. Fresh-task optima are unknown, so only verified size, non-full coverage, and calls are reported. Physical execution records may be shared to avoid recomputation, but each method has a separate logical cache; these comparisons are not wall-time benchmarks.

Current results come from the matched-budget rerun of all comparators and the ten-seed FutureCert runs; their report manifest pins the input files. Results of FutureCert (Utility-96) in Appendix F.3 use the earlier utility-ordering implementation and a 96-query cap, rather than the default four-restart solver. Framework mechanism, exact-structure, diagnostic, and cross-context studies retain their own declared protocols; they are not relabeled as new FutureCert experiments.

A new non-full subset evaluated by the planner counts as one online query. A method may cache repeated subsets within a decision. The original full output is shared input. Reusing existing experiment files across methods reduces physical reruns but does not merge their logical call budgets. We record validity, size gap, actual calls, exact minimum-size recovery, and full-retention rate. Full retention may itself be globally minimum or may indicate that a budgeted method found no smaller valid set, so the last two rates are separate. A returned set is certified valid because it was re-executed; minimum cardinality is proved once the query records rule out every smaller cardinality layer, including the vacuous proof for an accepted empty set. Budgeted recovery can match the known exact size without itself furnishing that proof.

Every returned incumbent passes the native verifier. Its excess size is $\mathrm{Gap}(\widehat{S}) = |\widehat{S}| - k^\star$. Exact-size recovery is scored against the separate exhaustive audit. An online minimum-size proof requires that all smaller sets have been ruled out; an accepted empty set, or an accepted singleton after a failed empty-set query, already meets this condition. The native prefix starts with a singleton, so a run whose optimum is empty can spend several queries before reaching it. Search stops at the cap, on candidate exhaustion, or when its queried records establish the implemented minimum-size stopping condition. We report logical calls, recovery, gap, and full-set fallback separately from offline preparation.

Retrieval feature and FutureCert (Beam-4) details. The TD-MPC2 retrieval vector concatenates the original action $a^\star$, first-step final weighted mean $\mu_0^\star$, original first-step standard deviation, descending-sorted $|a^\star|$, $\|a^\star\|_\infty$, mean standard deviation, and evidence count H. Distances use the coordinates stored by `retrieval_feature`; no evaluation label is an input. TD-MPC2 FutureCert (Beam-4) queries the empty set, expands each retained parent by adding one missing unit, and ranks unique invalid children by native fidelity deficit, segment count, and mask identifier. It keeps at most four parents for the next layer. When a feasible child is encountered in a completed or budget-truncated layer, the choice uses segment count, deficit, and mask identifier, followed by deletion descent. The beam width changes candidate exploration, while every feasibility decision still comes from the shared native predicate.

Cost and validity. The archive stores every proposed non-full mask before reading its native result; repeated masks use the within-decision cache. Offline labels and shared full-retention execution are excluded from the online cap. The recovery tables reuse completed native records and measure logical queries rather than new physical runtime. Physical timing checks in the unified archive match returned masks and query counts to replay records. No timing result is inferred from the framework-control replay.

B.4 Exact Reuse and Approximate Proposals

The exact-reuse backend caches the full-retention run's raw reward/value predictions separately from intervention-dependent return accumulation. A hit requires bitwise equality of the valuation inputs: latent state, complete candidate trajectory tensor, iteration context, and CPU/CUDA RNG states, under the same model and numerical configuration. The backend then applies the requested mask and reference values, accumulates returns in the original FP32 order, and restores the cached post-valuation RNG state. Optimizer updates and final elite/action selection execute natively. A

Table 7: **Exact-reuse profile on the 400 Cheetah Run and Cup Catch FutureCert flows.** Fractions weight charged queries. Divergence refers to the first mean/std update that differs from full retention (iterations numbered 1-6). All optimizer updates still run; reuse avoids matched value-model evaluations.

		First differing update (%)					
Task	Queries	1	2	3	5	Reused/6	Native valuation (%)
Cheetah	16,497	99.26	0.00	0.70	0.04	1.016	83.07
Cup	23,780	97.95	0.00	1.73	0.32	1.047	82.54

failed equality check executes the original valuation; later hits are permitted only after independently checking equality again.

Proposition 4 gives the equivalence conditions; its proof appears in Appendix B.6.

Implementation validation and reuse. A development gate compares 960 native/replay query pairs on action, mean, standard deviation, elite selection, and RNG state with zero mismatches. On the complete FutureCert paths for the 400 Cheetah Run and Cup Catch inputs, all 40,277 charged queries reproduce archived FP32 action/mean outputs bit for bit; final returned masks and logical query counts also match. These paths do not enumerate all masks and hence are not a complete-family equivalence test. Table 7 records first update divergence and actual valuation reuse. No query has an entirely matching mean/std update trace; no first divergence occurs at updates 4 or 6. The profile excludes the full-retention cache construction, which must be included in end-to-end timing.

Timing protocol. The paired timing protocol fixes ten inputs each on Cheetah Run and Cup Catch (indices 0,20,...,180), five repeats, randomized native/replay order, warm-up calls, CUDA synchronization, and both synchronized wall-clock and CUDA-event measurements. Each timed run includes the full target or cache construction and the complete FutureCert search. Isolation requires no foreign CUDA process on the selected device. Shared-load paired timing is complete (200/200 runs; Table 8); correctness-run durations are separate from those timing measurements.

Paired timing under shared GPU load. We additionally measure 20 inputs (10 each on Cheetah Run and Cup Catch) with five repetitions of each complete FutureCert backend on GPUs 0-3. Each input uses the same GPU for both backends and all repetitions, with randomized backend order. Full-target computation, cache setup, candidate ordering, and native queries are included. All queried action/mean outputs, returned sets, and call counts match the native archives. Other workloads remain active; the ratios below describe shared-load measurements.

Table 8: **Shared-GPU end-to-end timing.** Times are seconds, averaged over five repetitions per input and then 10 inputs per task. Parentheses give the SD across input means. Variance is the mean within-input sample variance over five repetitions (seconds squared). Ratios divide the task-level native and reuse means; they are not isolated speedup estimates.

Task	Native time ↓	Reuse time ↓	Native var.	Reuse var.	Time ratio
Cheetah	**5.119 (3.130)**	5.705 (3.563)	0.0019	0.0007	0.897
Cup	**6.085 (4.080)**	6.321 (4.247)	0.0025	0.0061	0.963

B.5 Frozen Proposal Screening

This control fixes a proposal ranking before consulting native labels. The frozen interface projects masks onto the four final-stage units. All non-full masks are ordered by frozen validity, cardinality, frozen output error, and integer mask identifier. We query every mask in the first $K \in \{1, 2, 4, 8, 16\}$ and return the smallest native-valid one; if none passes, the same FutureCert procedure continues with the shared cache and a total cap of 384. ExactSize@K uses the independently enumerated minimum size. The solver seed is fixed to 61831 for both this control and its matched FutureCert baseline.

Table 9: **Frozen proposals do not improve compact recovery in this control.** Task-macro results on the 400 Cheetah Run and Cup Catch inputs. All listed proposal checks share the 384-query cap with fallback. Calls exclude frozen candidate construction. Bold identifies the best final Gap.

Rule	Valid@K ↑	ExactSize@K ↑	Fallback (%) ↓	Gap ↓	ΔGap ↓	Calls ↓
FutureCert	-	-	-	**0.1975**	-	100.69
$K = 1$	3.00	2.75	97.00	0.2150	+0.0175	100.62
$K = 2$	6.00	5.25	94.00	0.2225	+0.0250	101.05
$K = 4$	8.50	6.75	91.50	0.2350	+0.0375	102.30
$K = 8$	15.00	11.25	85.00	0.2325	+0.0350	101.22
$K = 16$	26.25	19.25	73.75	0.2825	+0.0850	104.74

The frozen strategy discovers an accepted candidate more frequently as K grows, but its final Gap is above the matched FutureCert baseline at every tested K. Native acceptance controls validity; stopping at an accepted proposal does not optimize its size. This experiment changes both the proposal queries and the stopping rule, so their separate contributions to the observed Gap are not identified.

B.6 Exact Execution Reuse

Proposition 4 (conditional equivalence of exact replay). Assume deterministic native operators for fixed inputs, execution state, numerical configuration, and RNG state. Cache raw model evaluations with all their determining inputs and pre/post RNG state. At each intervened step, reuse a cached evaluation only if those inputs match bitwise, recompute every intervention-dependent operation in native order, and restore the matching post-evaluation RNG state; otherwise execute the native step. Then the resulting intervened output equals native re-execution bit for bit.

Proof. Both executions start from the same intervened planning state and RNG. Suppose their states agree before a valuation. On a miss, both evaluate the same native operators. On a hit, the matched inputs and deterministic operator assumption give identical raw predictions and post-valuation RNG. Applying the current intervention and native arithmetic order therefore gives identical returns in both executions. Their native optimizer update receives the same returns, trajectories, and random state, so the resulting state agrees. Induction through all iterations and the native final selection establishes output equality. Equality of the optimized mean alone is insufficient: candidate tensors, all relevant state, and RNG alignment are part of the premise. This argument preserves the original FP32 program; it does not replace common constants by a real-arithmetic cancellation. □

C Comparison Methods and Solver Variants

C.1 Explanation-method Adaptations

The 25 entries in Table 10 consist of seven principal method adaptations, twelve simplified mechanisms, and six internal controls. Together with default FutureCert, FutureCert (Native greedy), and enumeration, they give 28 implementations across the experiments. Simplified mechanisms use neutral display names; the table records the source that motivated each implementation. Every returned set is verified by re-executing the original planner.

Solver names. FutureCert (Beam-4), FutureCert (Surrogate-Search), FutureCert (Surrogate-Greedy), FutureCert (Enumeration), and FutureCert (Native greedy) instantiate the same intervention and native-verification framework with alternative search procedures. Unqualified FutureCert recovery results use the default four-restart solver. The same parenthesized names are used throughout the paper; raw result keys remain unchanged.

C.1.1 Comparison tasks

Fixed-evidence evaluation tests evidence selection; budgeted subset search tests recovery of a small valid set; fault diagnosis tests whether the selected depth intersects the injected fault. Attribution

baselines use the current evidence units and score target, with planner-specific input representations. The fixed-evidence KernelSHAP row selects sets using a Shapley-kernel regression prediction; its budgeted guided-search row additionally ranks candidates and verifies them through the planner. FutureCert (Beam-4) has width four and scores candidates through exact planner queries. Its name does not imply exhaustive enumeration.

C.1.2 SURROGATE ADAPTATIONS

Methods that cannot differentiate through tree search propose candidates through a frozen surrogate trained on development-set mask-output pairs. Evaluation decisions provide external assessment. FutureCert (Surrogate-Search) searches that same model, then submits candidates to the planner verifier. On random masks from evaluation decisions, the surrogate's root-argmax agreement is 85.6% (95% CI 83.5-87.8%) and mean root-distribution Spearman correlation is 0.802 (0.784-0.821). For preference-certificate classification, precision is 30.9%, recall 14.4%, and balanced accuracy 54.4%. These characterize action, distribution, and certificate errors separately.

C.1.3 ENTRY NAMES AND IMPLEMENTED ADAPTATIONS

Table 10 describes the evaluated implementations using neutral display names. Implementation aliases with identical outputs are counted once. ORTE uses part of the released code but reconstructs the model forward pass. Measured results belong to the specified implementations.

Table 10: Comparison methods and implemented adaptations. Names describe the evaluated mechanisms.

Method	Implementation and adaptation
TimeSeg	Uses official policy, value, and reward modules; trains a segment selector on a frozen planner surrogate and maps selected segments to depth order.
ORTE	Uses the public MaskGenerator, rebuilds training, and reconstructs the unavailable forward model from TimeX++; averages channel mask probabilities.
TIMING	Runs the released attribution module on the frozen surrogate's target-action probability; sums absolute channel attributions relative to the reference input by depth.
TimeX++	Uses official model modules and loss terms with the frozen surrogate as predictor; sets replacement statistics to zero relative to the reference input and aggregates channel probabilities by depth.
KernelSHAP	Fits masks to root visits with Shapley-kernel weighted ridge regression; fixed-size sets minimize predicted decision-retention error rather than simply taking the largest absolute attributions.
LIME	Fits root visits with locally weighted ridge regression; selects fixed-size sets by predicted retention condition.
RISE	Aggregates random binary masks with quality weights; the quality target combines certificate deficit, rank correlation, and margin error.
Singleton gain	Ranks single-unit gains in target-versus-fixed-competitor visit difference relative to empty retention; no path-integrated gradient is computed through planner or surrogate.
Mask regression	Fits a weighted linear model to temporal-mask queries; fixed-size sets use lexicographic action, rank, margin, and segment-break penalties, while budgeted search uses smoothed coefficient norms. The original dynamic-mask optimizer is not run.
Position coverage	Selects endpoints, then positions farthest from those selected; a positional-coverage heuristic.
Interval-mask regression	Runs Shapley-kernel regression over prefix, suffix, interval, and random masks; fixed-size search optimizes the surrogate retention condition and budgeted search ranks coefficient norms. The original TimeSHAP event pruning is not implemented.
Shallow-first	Selects depths shallow to deep; does not estimate WinIT's window-counterfactual importance.
Deep-first	Selects depths deep to shallow; does not estimate WinIT's window-counterfactual importance.
Center-first	Selects by distance to the depth center, breaking ties toward shallower depths; does not estimate WinIT's window-counterfactual importance.
Deletion divergence	Deletes one depth at a time and ranks the sum of root-rank change and normalized visit-vector change; uses depth-level planner signals.

Continued on next page

Table 10: Comparison methods, continued.

Method	Implementation and adaptation
Policy divergence	Ranks depths by mean Jensen-Shannon divergence between node policies in the original trace and the trace-wide mean policy.
Reward/value deviation	Adds depthwise-max-normalized mean absolute reward and value deviations from reference; no distributed reward-decomposition model is trained.
RewardSal	Ranks mean absolute reward deviation from reference within depth; native planner-signal control.
ValueSal	Ranks mean absolute value deviation from reference within depth; native planner-signal control.
Retain/delete heuristic	Greedily maximizes the sum of a sigmoid action margin for the retained set and a sigmoid margin drop after deleting it; uses our depth-level objective.
Diversity heuristic	Adds diversity and consistency scores from depthwise reward/value statistics to the VPS heuristic; it does not inherit the source objective's guarantees.
FutureCert (Surrogate-Greedy)	Builds an ordering by surrogate certificate-deficit forward greedy search, then verifies it with the planner; surrogate-guided solver variant.
FutureCert (Surrogate-Search)	Enumerates subsets on the same surrogate, proposes candidates by surrogate condition, then verifies and repairs them through planner queries; surrogate-guided solver variant.
FutureCert (Beam-4)	Width-four candidate search scored with exact planner queries; beam-search solver variant.
Random	For fixed-size experiments, averages 20 random sets per decision without selecting the best one.

C.2 Candidates Ranked Directly by the Planner

To test whether candidates derived from the planner itself, rather than through a frozen surrogate, can enter the same audit, we rank units by the deviation of the planner's own reward/value reads from their references and retain the top B. Table 11 shows that the discounted native ranking beats same-size random sets at $B = 3$ and $B = 6$ but stays far below the existence reference: ranking importance does not imply joint sufficiency. This is a native-ranking control. We also adapt the imagination-observation comparison of Wenninghoff & Schwammberger (2025), which contrasts a DreamerV3 agent's imagined and observed trajectories to diagnose failures and does not output an evidence set: after replaying each evaluation episode bitwise to recover the observed rewards, we rank units by the deviation between the planner's predicted quantity and the observed reward or discounted return (IOD). Ranking the largest deviations first gives 37.5% at $B = 3$, close to the discounted native ranking and far below the 80.0% attainable, with 29.0%, 1.0%, and 82.5% on Cheetah Run, Cup Catch, and Finger Spin. This is our adaptation, not a reproduction of the published method.

Table 11: **Candidates ranked directly by native reward/value deviations.** Full-fidelity pass rate (%, TD-MPC2 task macro over 600 inputs) of the top-B units under two rankings computed from the original planner reads, with no surrogate classifier. The uniform row is the exact expectation over same-size random sets; the existence row is whether any set of that size passes (an upper reference, not a method). IOD rows rank units by the deviation between the planner's predicted quantity and the observed reward or return, an adaptation of the imagination-observation comparison of Wenninghoff & Schwammberger (2025); the grouped variant keeps whole quantities ($B = 3, 6$). Computed from the exhaustive native archive without new planner execution.

Ranking	$B = 1$	$B = 2$	$B = 3$	$B = 6$
Discounted native deviation	26.5	30.2	37.2	46.5
Normalized reward/value deviation	23.2	28.3	28.3	41.0
Uniform mask expectation	21.9	24.2	26.0	31.4
Exhaustive existence reference	36.0	57.3	80.0	99.3
IOD, largest deviation first	25.8	30.0	37.5	49.0
IOD, grouped by quantity	-	-	37.5	46.7
IOD, smallest deviation first	19.2	19.2	20.5	22.7

D NATIVE VERIFICATION OF EXPLANATIONS

For Section 5.2, we compare complete selection pipelines and paired native outcomes, then examine numerical confirmation and computational reuse. Final submitted sets and preselected masks have separate denominators.

D.1 FINAL SUBMITTED EXPLANATIONS

Table 12: **Final explanations of published attribution pipelines versus FutureCert certificates.** Each pipeline selects one set per decision through its own interface (frozen candidate trajectories on TD-MPC2; the frozen search tree on tree search) and commits it before any native result is revealed; every committed set is then re-executed natively. *Valid* is the share of committed sets that pass the full fidelity predicate; *Min.* the share that are valid and have the exhaustively audited minimum size. FutureCert sets are valid by construction; its Min. averages ten solver seeds at the 384-query cap. Episode-weighted percentages with 95% hierarchical bootstrap intervals over episodes and decisions (2,000 draws).

Setting	Method	Valid (%)	Min. (%)
Cheetah Run	KernelSHAP	4.5 [1.5, 8.0]	4.0 [1.0, 8.0]
	LIME	5.0 [1.5, 9.0]	4.0 [1.0, 7.5]
	RISE	3.0 [0.5, 6.5]	3.0 [0.0, 6.5]
	FutureCert	100.0 [100.0, 100.0]	93.0 [89.7, 96.0]
Cup Catch	KernelSHAP	3.5 [0.5, 7.0]	3.5 [0.5, 6.5]
	LIME	4.0 [1.0, 7.5]	4.0 [1.0, 7.5]
	RISE	3.0 [0.5, 6.5]	3.0 [0.5, 6.0]
	FutureCert	100.0 [100.0, 100.0]	86.1 [79.8, 91.5]
Finger Spin	KernelSHAP	55.5 [46.5, 64.5]	55.5 [46.0, 64.0]
	LIME	55.5 [46.5, 64.5]	55.5 [46.5, 64.0]
	RISE	55.5 [47.0, 64.0]	55.5 [47.0, 64.0]
	FutureCert	100.0 [100.0, 100.0]	99.8 [99.0, 100.0]
MuZero Pong	KernelSHAP	92.1 [88.5, 95.4]	78.1 [70.7, 86.1]
	LIME	91.8 [88.4, 95.1]	76.7 [68.9, 85.7]
	RISE	92.1 [88.6, 95.0]	75.8 [68.0, 84.8]
	FutureCert	100.0 [100.0, 100.0]	99.2 [98.0, 100.0]
MuZero Breakout	KernelSHAP	81.3 [75.0, 87.9]	61.8 [54.9, 68.0]
	LIME	81.9 [76.7, 87.1]	62.1 [55.9, 68.1]
	RISE	81.8 [75.1, 88.3]	57.1 [51.5, 62.3]
	FutureCert	100.0 [100.0, 100.0]	100.0 [100.0, 100.0]
SEZ Pong	KernelSHAP	75.5 [70.9, 80.3]	50.1 [44.5, 56.0]
	LIME	75.7 [71.0, 80.3]	49.3 [43.8, 55.0]
	RISE	79.3 [74.0, 84.2]	44.6 [38.1, 51.4]
	FutureCert	100.0 [100.0, 100.0]	99.8 [99.2, 100.0]

The interface timing audit uses 30 pipelines per interface, comprising ten Cheetah/Cup inputs and three methods. Mean dynamics-model calls are 320, 1,435, 1,538, and 1,531 for frozen candidates, final-two refresh, final-four refresh, and native re-execution, respectively. Table 1 reports false acceptance and median relative pipeline time.

Preselected masks. The fixed-candidate control freezes the original TD-MPC2 candidate trajectories and recomputes their scores after replacing selected reward and value predictions. It then applies the same action and optimized-mean thresholds used by the native verifier. The native test instead restarts all six MPPI iterations with the same warm start and random stream, allowing weights, updated means, and later candidate trajectories to change. For each of the 400 Cheetah Run and Cup Catch decisions we sample two masks at each of cardinalities 2, 4, 6, and 8, giving 3,200 paired tests. This comparison isolates failures caused by freezing the post-intervention optimization path; it does not compare two evidence-selection methods.

The final-explanation audit in Table 13 instead runs complete fixed-candidate KernelSHAP, LIME, and RISE selection pipelines. Their query code and seed offsets match the native-query adaptations, changing only the oracle interface; every submitted mask is written before native results are

Table 13: Native audit of final explanations selected by complete fixed-candidate pipelines. FA is the percentage of submitted fixed-verified sets rejected by the native declared predicate, and Act. is the percentage also failing the action condition (both ↓). Each method submits one set per decision: 200 decisions per original setting and 100 from independent Cup Catch episodes.

	KernelSHAP		LIME		RISE	
Setting	FA ↓	Act. ↓	FA ↓	Act. ↓	FA ↓	Act. ↓
Cheetah Run	95.5	93.5	95.0	91.0	97.0	93.5
Cup Catch	96.5	94.5	96.0	93.0	97.0	94.5
Finger Spin	44.5	40.0	44.5	40.0	44.5	40.0
Independent Cup Catch	96.0	94.0	95.0	93.0	96.0	95.0

Table 14: Rejection of the same fixed-accepted final sets as progressively more of TD-MPC2 is allowed to adapt (%, ↓). FA applies the declared predicate; Act. counts action-condition failures. Each partial condition preserves the input, warm start, reference, target, and random stream.

	Cheetah Run		Cup Catch		Independent Cup Catch	
Execution	FA ↓	Act. ↓	FA ↓	Act. ↓	FA ↓	Act. ↓
Freeze all six iterations	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Refresh final two iterations	82.5	80.0	92.5	85.2	93.0	91.0
Refresh final four iterations	92.3	81.2	97.2	93.8	97.7	93.0
Refresh all six iterations	95.8	92.7	96.5	94.0	95.7	94.0

Table 15: Paired judgments on 3,200 preselected TD-MPC2 masks, not final explanation outputs. Rejected is the number of fixed accepts that native execution rejects; the percentage uses fixed accepts as its denominator. The independent Cup row uses a separate cohort. The lower panel gives the complete pooled confusion matrix for the 3,200 Cheetah/Cup pairs.

Setting	Fixed accepts	FutureCert confirms	Rejected	Rejected (%)
Cheetah Run	433	80	353	81.52%
Cup Catch	597	59	538	90.12%
Pooled	1,030	139	891	86.50%
Independent Cup Catch	279	22	257	92.11%

	Native valid	Native invalid	Total
Fixed-candidate valid	139	**891**	1,030
Fixed-candidate invalid	97	2,073	2,170
Total	236	2,964	3,200

read. With frozen candidate pools, TD-MPC2's final action and optimized mean depend only on the four last-stage evidence units. We physically evaluate all 16 patterns of those units, verify the projection on random masks and every submitted mask, and then physically re-execute each submitted set with adaptive optimization. Full-retention outputs reproduce the archived target bit for bit, and all 2,100 final native labels agree with the archived native audit. The 600 original decisions span the three TD-MPC2 tasks; the independent Cup Catch row uses 100 decisions from ten previously collected episodes. These are retrospective evaluations of archived observations, not fresh trajectories collected after this audit. Per-decision submissions, output checks, input hashes, and the episode-bootstrap summary are recorded in the audit artifact, which will accompany the code release.

The partial-refresh audit in Table 14 is a mechanism probe on the same already submitted masks, without rerunning selection; it is separate from interface re-selection. In the one-stage condition, iterations 1-4 reuse their original action-sequence pools; iterations 5-6 draw new candidates from the means updated under the intervention. In the two-stage condition, only iterations 1-2 remain fixed. All six iterations still consume the same random draws, so the change isolates whether the

Table 16: Complete query-interface re-selection on Cheetah Run. FA and Act. are native declared-condition and action failure rates among all submitted sets (%); Non-full is non-full output coverage (%). Calls count the selection interface's new non-full queries, excluding the shared original target and separate final native audit. The native interface's zero FA follows from using the declared verifier.

Interface	Method	FA ↓	Act. ↓	Non-full ↑	Size ↓	Calls ↓
Freeze 6	KernelSHAP	95.5	93.5	100.0	1.92	55.0
	LIME	95.0	91.0	100.0	2.35	63.7
	RISE	97.0	93.5	100.0	1.88	47.4
Refresh 2	KernelSHAP	93.5	90.0	100.0	3.19	61.6
	LIME	89.0	88.0	100.0	3.92	70.7
	RISE	94.0	92.5	100.0	3.22	57.3
Refresh 4	KernelSHAP	75.5	73.0	96.0	4.30	65.8
	LIME	67.0	65.0	96.0	5.05	71.6
	RISE	70.5	68.5	96.5	4.75	60.2
Native	KernelSHAP	**0.0**	0.0	94.5	4.93	71.4
	LIME	**0.0**	0.0	95.5	5.29	74.3
	RISE	**0.0**	0.0	94.5	5.21	61.4

intervened scores are allowed to alter later proposal pools. The full-evidence endpoint matches the archive exactly in all 500 decisions. Results are reported for every method and condition; partial-refresh labels never feed back into mask selection.

D.2 Partial Refresh of Adaptive Computation

The two partial interfaces replay the original candidate pools only for their frozen prefix. Subsequent pools are generated from the intervention-updated mean and variance with the original random stream. Each explainer reruns its own proposal, validation, and repair process independently for every interface; no native audit outcomes enter partial-interface selection. Each state/method/interface uses its own 96-call logical cache. Exact native-output caches accelerate the native-interface branch, while every final submitted set is physically replayed and checked coordinatewise against the native archive. The 6,300 newly selected outputs cover all 700 decisions, three methods, and three new interfaces. Adding the original fully frozen outputs gives 8,400 pipeline outputs.

The fully frozen pipelines are detailed in Table 13. In 1,749/1,800 original-task outputs they omit every early-stage unit, consistent with the frozen interface's structural insensitivity. Table 14 holds those selected sets fixed while restoring adaptation; Table 1 instead reruns the entire selection process at each interface. These are different controls and use different submitted sets.

All completed states before workload repartitioning are retained once; a partially completed boundary state is rerun in a disjoint shard. The aggregation script verifies full state/method/interface coverage, agreement with saved submissions, charged-mask counts, and native consistency. Shared physical caches across methods reduce execution time without altering each method's logical budget. Consequently the logged method-order wall times are not used as standalone speed comparisons. Per-method hierarchical episode-decision bootstrap intervals and all mask traces are included in the result archive.

Complete pipeline cost. We execute 120 complete pipelines on 10 fixed archived inputs (the first input of each of five episodes on Cheetah Run and Cup Catch), crossing three selection methods and four query interfaces. Each pipeline has its own cache. Frozen-interface timing includes preparing all 16 final-stage mask projections; partial and native interfaces physically re-execute every new query. Interface order rotates across input and method. CUDA synchronization brackets the full selection interval; common full-target capture and the subsequent independent native audit are outside it. Model-forward hooks count dynamics and reward calls. GPU0 is shared with a pre-existing training process, so elapsed times are descriptive measurements on this implementation, not isolated deployment benchmarks. The partial-refresh interpreter still evaluates the complete optimizer loop and does not implement every possible reuse optimization. Timing and forward counts make

Table 17: Complete query-interface re-selection on Cup Catch. FA and Act. are native declared-condition and action failure rates among all submitted sets (%); Non-full is non-full output coverage (%). Calls count the selection interface's new non-full queries, excluding the shared original target and separate final native audit. The native interface's zero FA follows from using the declared verifier.

Interface	Method	FA ↓	Act. ↓	Non-full ↑	Size ↓	Calls ↓
Freeze 6	KernelSHAP	96.5	94.5	100.0	1.27	48.5
	LIME	96.0	93.0	100.0	1.31	56.4
	RISE	97.0	94.5	100.0	1.32	48.1
Refresh 2	KernelSHAP	92.5	87.5	100.0	2.85	63.2
	LIME	95.5	91.0	100.0	3.33	73.3
	RISE	95.0	89.5	100.0	2.96	57.5
Refresh 4	KernelSHAP	78.5	71.5	98.0	4.80	71.6
	LIME	74.5	68.0	98.0	5.50	78.1
	RISE	70.5	63.5	99.0	5.46	65.0
Native	KernelSHAP	**0.0**	0.0	99.0	5.44	76.6
	LIME	**0.0**	0.0	99.0	5.97	81.0
	RISE	**0.0**	0.0	97.5	5.93	66.8

Table 18: Complete query-interface re-selection on Finger Spin. FA and Act. are native declared-condition and action failure rates among all submitted sets (%); Non-full is non-full output coverage (%). Calls count the selection interface's new non-full queries, excluding the shared original target and separate final native audit. The native interface's zero FA follows from using the declared verifier.

Interface	Method	FA ↓	Act. ↓	Non-full ↑	Size ↓	Calls ↓
Freeze 6	KernelSHAP	44.5	40.0	100.0	0.14	7.6
	LIME	44.5	40.0	100.0	0.14	7.8
	RISE	44.5	40.0	100.0	0.14	7.7
Refresh 2	KernelSHAP	40.0	36.5	100.0	0.24	10.3
	LIME	40.5	37.0	100.0	0.26	10.8
	RISE	40.0	36.5	100.0	0.25	10.2
Refresh 4	KernelSHAP	35.0	32.5	100.0	0.36	13.5
	LIME	35.5	32.5	100.0	0.46	14.7
	RISE	35.0	32.5	100.0	0.43	14.3
Native	KernelSHAP	**0.0**	0.0	100.0	0.65	24.9
	LIME	**0.0**	0.0	100.0	0.75	26.7
	RISE	**0.0**	0.0	100.0	0.72	25.9

the speed-error trade-off explicit on these inputs. They do not identify a universally safe refresh depth. An approximate-interface pass is therefore not a native certificate unless a native test or a separately justified equivalence condition is supplied. The three attribution selection methods are held fixed while the query interface changes; the discrepancy is not attributed to an intrinsic defect of KernelSHAP, LIME, or RISE.

D.3 Numerical Confirmation

Before re-enumeration we fixed 30 inputs: two previously sampled inputs from each of five episodes on each TD-MPC2 task. We re-executed all 4,096 masks per input under the constant-free interpreter, including the frozen-interface projection used by the attribution pipelines. Full-retention outputs and recorded random-stream digests agree with the original execution. The development candidate bank is kept unchanged to isolate numerical implementation effects; this is not a newly tuned constant-free solver.

The aggregate false-acceptance result is unchanged on this fixed sample. Certificate composition is not: nine minimum sizes and ten Cores change, most of them in Cup Catch. The mean change

Table 19: Complete query-interface re-selection on Independent Cup Catch. FA and Act. are native declared-condition and action failure rates among all submitted sets (%); Non-full is non-full output coverage (%). Calls count the selection interface's new non-full queries, excluding the shared original target and separate final native audit. The native interface's zero FA follows from using the declared verifier.

Interface	Method	FA ↓	Act. ↓	Non-full ↑	Size ↓	Calls ↓
Freeze 6	KernelSHAP	96.0	94.0	100.0	1.43	50.7
	LIME	95.0	93.0	100.0	1.55	60.2
	RISE	96.0	95.0	100.0	1.44	48.5
Refresh 2	KernelSHAP	91.0	86.0	100.0	3.06	64.3
	LIME	92.0	84.0	100.0	3.42	74.0
	RISE	95.0	90.0	100.0	3.31	59.2
Refresh 4	KernelSHAP	73.0	66.0	100.0	5.31	76.2
	LIME	78.0	73.0	100.0	5.47	79.3
	RISE	65.0	60.0	99.0	5.81	65.7
Native	KernelSHAP	**0.0**	0.0	99.0	5.74	77.3
	LIME	**0.0**	0.0	99.0	6.50	81.5
	RISE	**0.0**	0.0	99.0	6.05	68.3

Table 20: **Measured cost and native false acceptance of complete pipelines.** Each row has 30 final submitted sets. Ratio is the median paired pipeline time divided by its native-interface time. FA is false acceptance (↓); Seconds and model-forward calls measure this implementation (↓).

Interface	FA ↓	Mean seconds ↓	Time ratio ↓	Dynamics ↓	Reward ↓
Freeze 6	30/30	0.600	0.221	320.0	288.0
Refresh last 2	29/30	2.661	0.944	1434.7	1291.2
Refresh last 4	22/30	2.864	0.996	1538.0	1384.2
Native	0/30	2.847	1.000	1530.7	1377.6

Table 21: **Core numerical confirmation on the same inputs.** False acceptance (FA, ↓) concerns the final accepted sets of complete KernelSHAP/LIME/RISE selection pipelines. Structural columns compare complete minimum families under the two interpreters.

Setting	FP32 FA	Constant-free FA	$k^\star$ changed	Core changed	Family Jaccard
Cheetah Run	30/30	30/30	0/10	0/10	0.950
Cup Catch	30/30	30/30	7/10	9/10	0.175
Finger Spin	21/30	21/30	2/10	1/10	0.590
Pooled	81/90	81/90	9/30	10/30	0.572

in minimum size is 0.000 units over the 30 inputs, but this mean masks individual changes. Thus the interface mismatch persists without candidate-shared constants, while fine certificate anatomy remains a property of the particular numerical interpreter.

D.4 Frozen-Tree Interface for Tree Search

Interface. For each of the 1,094 core decisions, the unmodified search is run once and its final tree is recorded: for every expanded node, identified by its root action path, the predicted reward (MuZero) or value prefix (SEZ), the predicted value, the child priors, and the SEZ reset flag. A frozen query never calls the model. A node at depth d reads unit $\min(d, H)$; removed units receive the same references as the native audit. The 50 simulations are replayed with LightZero's own node, selection, UCB, backup, and min-max routines and the same random seed, expanding recorded nodes from the record. When selection chooses a child absent from the recorded tree, the primary interface F1 redirects it to the best recorded child under the same scores and tie rule, and backs up the node's masked value if no recorded child exists; the pre-declared sensitivity variant F2 stops the simulation instead. The output is the root pseudo-visit vector, judged by the unchanged predicate.

Root argmax-Q was rejected on development decisions because it disagrees with the deployed action under full retention.

Gates. Full retention reproduces every node's visit count and value sum and the root visits of the archived search on all 1,094 decisions under both variants, with no out-of-tree selection. All 126,590 masked reads observed in native traces equal the float32 reference, and on the 101 native masked runs that never left the recorded tree, both variants reproduce the native visits bitwise. Every committed set is re-executed natively; 4,424 comparisons with archived native results match.

Results. KernelSHAP, LIME, and RISE use the same selection pipelines as the TD-MPC2 audit, querying only the frozen tree. Native re-execution rejects 16.5% of the committed sets under F1 and 25.1% under F2 (Table 22); 494 of the 551 F1 rejections keep the action and fail the ranking or margin condition. Among non-full commitments the F1 rate is 26.4%, and all 69 non-full sets committed on decisions whose exact minimum is the full set are rejected. On probe masks (empty plus two random masks per decision), F1 agrees with native validity for 97.6%, 92.5%, and 91.6% of probes in MuZero Pong, MuZero Breakout, and SEZ Pong. A frozen search tree is therefore a much closer approximation than TD-MPC2's frozen candidate pools, whose final action sees only 4 of 12 units, and the false-acceptance rate is a property of the planner and of the frozen interface rather than a constant of attribution methods.

Table 22: **Native audit of final explanations selected through a frozen search tree.** False acceptance (%, episode-weighted; counts in parentheses) of the sets that complete frozen-tree KernelSHAP, LIME, and RISE pipelines commit on the 1,094 core tree-search decisions; the last column counts action failures among all committed sets. F1 (primary) redirects selections of unrecorded children to the best recorded child; F2 (pre-declared sensitivity) stops such simulations. Macro rows give 95% hierarchical bootstrap intervals. All three methods commit the identical set on 908 of 1,094 decisions, so the pooled count overstates independent evidence.

Interface	Setting	KernelSHAP	LIME	RISE	Action fail.
F1	MuZero Pong	7.9 (27/338)	8.2 (28/338)	7.9 (27/338)	13/1014
	MuZero Breakout	18.7 (69/373)	18.1 (67/373)	18.2 (67/373)	5/1119
	SEZ Pong	24.5 (94/383)	24.3 (93/383)	20.7 (79/383)	39/1149
	Macro, all three		16.5 [14.4, 18.6] (551/3282)		57/3282
F2	MuZero Pong	17.7 (61/338)	19.2 (66/338)	18.0 (62/338)	29/1014
	MuZero Breakout	23.6 (88/373)	22.8 (85/373)	20.7 (77/373)	6/1119
	SEZ Pong	35.8 (137/383)	35.6 (136/383)	33.0 (126/383)	90/1149
	Macro, all three		25.1 [22.4, 28.0] (838/3282)		125/3282

E Fidelity and Nonmonotonicity

This section expands the fidelity and stability results in Sections 5.4 and 5.6.

E.1 Fidelity and Evidence Sensitivity

Table 23: Two-unit explanations of the same MuZero Pong decision. FutureCert's exact audit finds $\{2, 3\}$ as a minimum certificate. Both selected sets preserve action 2; the KernelSHAP set changes the visit margin from 2 to 19. Fidelity requires $\rho \geq 0.95$ and NME ≤ 0.10.

Explanation	Retained units	Action	Margin	NME ↓	Fidelity
Original planner	$\{1, 2, 3\}$	2	2	0	Pass
KernelSHAP	$\{1, 2\}$	2	19	2.429	Fail
FutureCert certificate	$\{2, 3\}$	2	2	**0**	**Pass**

Table 23 gives the same-size example.

Supporting evidence requirements and case audit

Table 24: Action agreement and the declared-statistic predicate require different evidence. Minimum retention is the mean percentage of units in an exact minimum certificate. Empty-set pass rate replaces every audited prediction. Parentheses give preference minus action in percentage points.

	Minimum retention (%) ↓		Empty-set pass rate (%) ↑	
Setting	Action	Full (change)	Action	Full (change)
MuZero Pong	12.3	74.7 (+62.4)	76.4	3.4 (−73.0)
MuZero Breakout	6.1	47.5 (+41.4)	78.7	15.0 (−63.7)
SEZ Pong	10.5	50.9 (+40.4)	72.0	9.4 (−62.6)

Table 25: A native re-execution example in MuZero Pong. The original visit-count margin is 13 and its normalization range is 17. Adding depth 1 to a valid depth-4 certificate violates the NME tolerance of 0.10.

Evidence	Visit margin	NME ↓	Full fidelity
Original planner	13	0	Pass
Minimum certificate $\{4\}$	12	0.059	Pass
Expanded set $\{1, 4\}$	17	0.235 (+0.176)	Fail

Table 26: Evidence required to retain the action and preference. Entries in the first block are mean minimum retained fractions; entries in the second block are pass rates after replacing every evidence unit (%). A/U additionally uses uniform MuZero policy priors. n is the number of decisions; "-" means the control was not run.

		Minimum retained fraction ↓				Pass rate after full replacement (%) ↑				
Setting	n	A	A+R	A+M	P	A	A+R	A+M	P	A/U
MuZero Pong	338	.123	.407	.639	.747	76.4	26.2	4.8	3.4	18.1
MuZero Breakout	373	.061	.256	.295	.475	78.7	38.2	23.2	15.0	0.0
SEZ Pong	383	.105	.245	.392	.509	72.0	47.1	15.9	9.4	-

Retention conditions Table 27 gives mean minimum fractions, empty and full-retention rates, partial-certificate uniqueness, and hierarchical bootstrap intervals. For MuZero Pong, the preference certificate retains 0.747 of the units on average and 49.8% of decisions require all units. The corresponding mean/full-retention pairs are 0.475/14.8% for Breakout and 0.509/25.3% for SEZ Pong.

Evidence granularity The TD-MPC2 coarse control binds three optimization stages for each predicted quantity. Expressed in fine units, minimum retention rises from 2.740 to 7.155 on Cheetah Run, 3.060 to 9.390 on Cup Catch, and 0.610 to 1.770 on Finger Spin (Table 28). The coarse grouping restricts expressible sets, so its exact minimum on the same decision cannot be smaller when expressed in fine units. Uniqueness uses all decisions.

Reference construction and transfer We test two distinct questions: whether a certificate fixed under one reference still passes under a new reference, and whether re-solving yields the same minimum size. The former checks action, ranking, and margin separately; the latter re-enumerates every retained set. Reference-episode bootstrap draws 16 references for each fixed set (Table 61).

Table 29 tests representative minimum certificates from the original reference library under an independent library. The action test includes 336, 349, and 334 decisions; equal minimum size after re-solving uses all 338, 373, and 383 core decisions. The two Pong libraries contain repeated deterministic trajectory types.

Reference perturbations after full evidence replacement On decisions with $0 < k^\star < H$ under the fixed reference, we resample references and measure action retention after replacing every evidence unit. Table 30 includes 163, 262, and 250 decisions. Per-node sampling draws a reward/value pair for each node; depth-shared sampling draws one pair for all nodes at a depth; reference-episode bootstrap averages depth means from resampled episodes. We average reference draws per decision,

Table 27: Certificate structure under the four retention conditions. Intervals are 95% hierarchical bootstrap CIs.

Minimum retention and empty certificates

Setting	Condition	n	Mean $k^\star/H \downarrow$	Empty (%)
MuZero Pong	Action	338	0.123 [0.082, 0.167]	76.4 [65.5, 85.1]
MuZero Pong	Action + margin	338	0.639 [0.559, 0.752]	4.8 [1.7, 8.4]
MuZero Pong	Action + ranking	338	0.407 [0.326, 0.498]	26.2 [19.8, 33.7]
MuZero Pong	Full preference	338	0.747 [0.674, 0.839]	3.4 [0.8, 7.2]
MuZero Breakout	Action	373	0.061 [0.046, 0.080]	78.7 [73.9, 83.5]
MuZero Breakout	Action + margin	373	0.295 [0.258, 0.343]	23.2 [17.4, 29.0]
MuZero Breakout	Action + ranking	373	0.256 [0.197, 0.335]	38.2 [27.3, 47.7]
MuZero Breakout	Full preference	373	0.475 [0.401, 0.566]	15.0 [7.8, 21.9]
SEZ Pong	Action	383	0.105 [0.076, 0.134]	72.0 [65.8, 78.2]
SEZ Pong	Action + margin	383	0.392 [0.354, 0.431]	15.9 [11.6, 20.3]
SEZ Pong	Action + ranking	383	0.245 [0.196, 0.302]	47.1 [38.7, 55.8]
SEZ Pong	Full preference	383	0.509 [0.451, 0.571]	9.4 [6.2, 12.8]

Full retention and uniqueness

Setting	Condition	n	Full retention (%)	Uniqueness among partial (%)
MuZero Pong	Action	338	3.5 [1.5, 6.0]	63.1 [42.4, 78.9]
MuZero Pong	Action + margin	338	36.1 [22.7, 53.8]	78.9 [71.7, 85.7]
MuZero Pong	Action + ranking	338	17.4 [10.3, 25.5]	73.5 [63.9, 83.3]
MuZero Pong	Full preference	338	49.8 [35.7, 69.2]	84.7 [76.1, 93.8]
MuZero Breakout	Action	373	0.5 [0.0, 1.6]	67.3 [51.5, 81.3]
MuZero Breakout	Action + margin	373	4.7 [0.0, 11.5]	74.4 [67.3, 80.8]
MuZero Breakout	Action + ranking	373	3.1 [0.5, 6.5]	64.9 [57.9, 72.1]
MuZero Breakout	Full preference	373	14.8 [7.6, 23.4]	78.4 [71.5, 85.1]
SEZ Pong	Action	383	2.6 [0.5, 5.2]	49.8 [39.3, 59.9]
SEZ Pong	Action + margin	383	15.1 [10.7, 20.1]	73.8 [67.9, 79.8]
SEZ Pong	Action + ranking	383	9.7 [5.8, 14.0]	48.4 [36.3, 58.6]
SEZ Pong	Full preference	383	25.3 [17.9, 32.7]	74.1 [67.2, 81.9]

Table 28: Evidence granularity on the same TD-MPC2 decisions. One coarse group contains three fine units; intervals are 95% hierarchical bootstrap CIs.

Evidence size

Setting	Fine units retained ↓	Coarse groups retained ↓	Equivalent fine units ↓
Cheetah Run	2.740 [2.545, 2.945]	2.385 [2.190, 2.590]	7.155 [6.570, 7.770]
Cup Catch	3.060 [2.820, 3.305]	3.130 [2.910, 3.310]	9.390 [8.730, 9.930]
Finger Spin	0.610 [0.500, 0.725]	0.590 [0.480, 0.715]	1.770 [1.440, 2.145]

Certificate uniqueness

Setting	Fine uniqueness (%)	Coarse uniqueness (%)
Cheetah Run	45.0 [34.5, 56.0]	85.0 [78.5, 91.0]
Cup Catch	50.0 [40.0, 60.5]	89.0 [83.5, 94.0]
Finger Spin	60.5 [52.5, 68.5]	71.5 [64.0, 78.5]

Table 29: Certificate transfer to an independent reference library (%). Action retention tests representative certificates from the original library; equal minimum size is assessed after re-solving.

Measure	MuZero Pong	MuZero Breakout	SEZ Pong
Original minimum retains action	100.0	99.7	97.0
Same $k^\star$ after re-solving	99.4	83.0	69.5

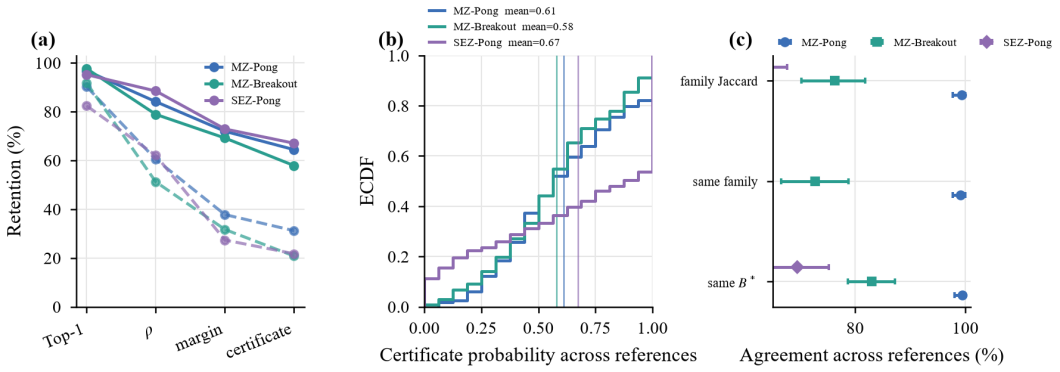


Figure 5: Reference robustness of fixed certificates under action, ranking, margin, and full-preference tests, together with changes in the minimum solution family. Labels $B^\star$ denote the minimum size $k^\star$.

decisions per episode, then episodes equally. These are a different sample and intervention estimator from the fixed-reference, all-decision rates in Table 26.

Table 30: Action retention under reference perturbations (%). The three settings contain 163, 262, and 250 partial-certificate decisions.

Reference perturbation	MuZero Pong	MuZero Breakout	SEZ Pong
Per-node sampling	52.4	64.4	53.3
Depth-shared sampling	78.4	75.1	69.9
Reference-episode bootstrap	83.7	79.9	78.2

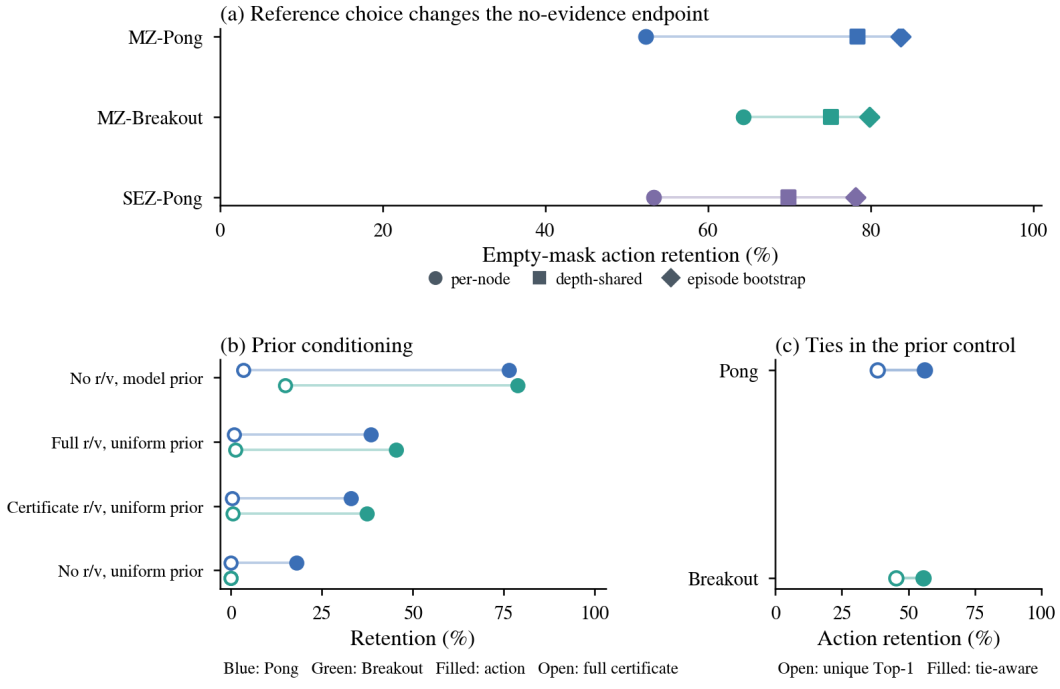


Figure 6: Effects of reference construction and policy priors after evidence replacement. Panel (a) gives reference perturbations for partial-certificate decisions; (b) gives the prior control over all confirmed MuZero decisions; (c) compares unique-action and tie-allowing retention with reward/value evidence intact and a uniform prior.

Action stability and certificate changes We hold one partial minimum certificate fixed, resample replacement values by reference episode, and re-execute 16 times. Action retention is 96.7%, 97.6%, and 95.1% across the core settings, while full preference passes in 64.4%, 57.8%, and 67.0%. Resolving the minimum family under the new reference is a separate test: the full families match in 99.1%, 66.6%, and 42.2%. Pong's repeated reference trajectories make its effective reference change smaller. Table 61 separates the four fixed-set pass rates, and Figure 5 shows certificate stability.

Single-unit extension audit We audit every one-unit extension of each expandable minimum certificate. Table 31 reports frequencies per extension edge and per decision; these percentages have different denominators. Of inspected extensions, 36.1-58.2% fail. Among expandable decisions, 45.6-81.3% have at least one failure. The edge bootstrap keeps all extension edges of a resampled decision together.

Table 31: Audit of single-unit extensions of minimum certificates. Intervals are 95% hierarchical bootstrap CIs; denominators differ by column.

Failure rates

Setting	Witness / all decisions (%)	Witness / expandable decisions (%)	Failed extensions (%)
MuZero Pong	25.9 [13.4, 36.3]	45.6 [26.6, 59.5]	36.1 [22.3, 47.3]
MuZero Breakout	56.4 [45.1, 65.1]	66.0 [55.7, 75.0]	43.1 [38.6, 47.4]
SEZ Pong	60.9 [52.3, 69.1]	81.3 [74.1, 88.1]	58.2 [53.8, 62.8]

Denominators

Setting	n	Expandable decisions	Extension edges
MuZero Pong	338	175	641
MuZero Breakout	373	317	1212
SEZ Pong	383	286	1411

Table 32 decomposes extension failures. Margin failures are the largest component in every setting, while 2.3-6.6% of audited edges also change the action. The reported NME quantiles show a range of errors beyond one threshold value. Relaxing only the NME cutoff from 0.10 to 0.20 reduces the episode-weighted failure rates to 27.1%, 31.2%, and 43.7% on the same original edge population. Because the starting sets were solved at NME 0.10, this sensitivity calculation isolates predicate evaluation; it is distinct from re-solving the minimum family under a new threshold.

Table 32: Failure modes for one-unit extensions of minimum preference certificates. Rates weight episodes equally; component columns may overlap. NME quantiles pool audited edges. The final column re-evaluates the same audited edges after relaxing only the NME threshold from 0.10 to 0.20; it does not re-solve the minimum certificate.

Extension failures

Setting	Edges	Any failure (%)	Action (%)	Ranking (%)
MuZero Pong	641	36.1	2.3	14.4
MuZero Breakout	1212	43.1	3.2	21.0
SEZ Pong	1411	58.2	6.6	19.9

Margin sensitivity

Setting	Margin (%)	Median / 90th percentile NME	Failure at NME ≤ 0.20 (%)
MuZero Pong	31.2	0.062 / 0.389	27.1
MuZero Breakout	36.3	0.074 / 0.333	31.2
SEZ Pong	51.2	0.105 / 0.533	43.7

Same-size attribution example For MuZero Pong cached decision 15 (original index 5803, episode 0), $H = 3$ and $k^\star = 2$. The target, attribution-set, and certificate visit vectors are

$$q^\star = [9, 8, 12, 6, 10, 5], \quad q_{\{1,2\}} = [6, 6, 25, 4, 6, 3], \quad q_{\{2,3\}} = [8, 8, 11, 7, 9, 7]. \tag{10}$$

The normalization range is $12 - 5 = 7$. The attribution set has margin error $|19 - 2|/7 = 17/7$; the certificate has zero. Empty retention and each singleton fail the full condition. Table 33 gives the action slot, correlation, and margin error of each set.

Table 33: Re-execution of one Pong decision with two retained units. Action slots start at zero.

Input	Retained depths	Action slot	Visit-count margin	Spearman ρ	NME	Full condition
Original planner	1, 2, 3	2	2	1.0000	0	Pass
KernelSHAP	1, 2	2	19	0.9411	2.4286	Fail
Minimum certificate	2, 3	2	2	0.9710	0	Pass

Re-execution example of a failed extension The audit covers every one-unit extension of each original minimum certificate. The all-decision witness rate uses all decisions; the expandable-decision rate uses $k^\star < H$; the edge-failure rate uses audited edges. Intervals resample episodes and whole decision clusters to retain dependence among a decision's edges. In MuZero Pong cached decision 44, the representative certificate is depth $\{4\}$ and adding depth 1 fails the full condition. Original, certificate, and expanded-set root visits are

$$[8, 21, 5, 7, 4, 5], \quad [8, 20, 5, 7, 4, 6], \quad [8, 25, 4, 6, 3, 4]. \tag{11}$$

Their margins are 13, 12, and 17. Restoring the unit changes subsequent tree traversal and visit allocation; the margin condition fails.

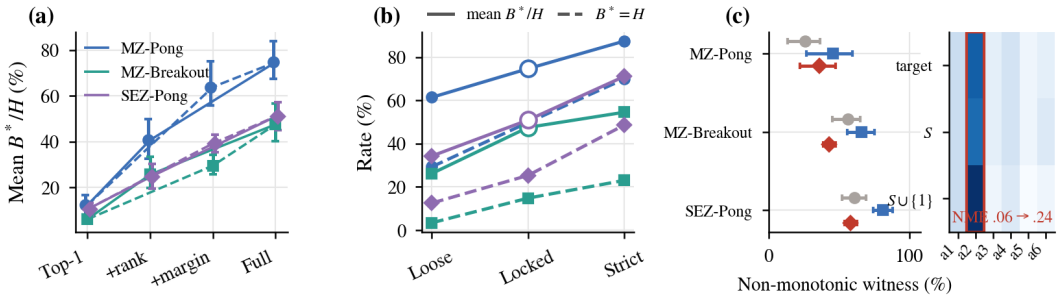


Figure 7: **Fidelity conditions change both the amount and structure of sufficient evidence.** (a) Evidence required by action, ranking, margin, and full predicates. (b) Minimum retained fraction (solid) and full-retention rate (dashed) under loose, locked, and strict threshold pairs. (c) Nonmonotone one-unit extensions, with all decisions, expandable decisions, and audited edges shown under their distinct denominators; the example adds unit 1 and changes NME from 0.06 to 0.24. Labels $B^\star$ denote the minimum size $k^\star$.

E.2 Complete Fixed-size Comparisons

Selection with a fixed number of evidence units See Table 34 for the representative methods; the following table gives all methods, including the time-series explainers TIMING (Jang et al., 2025), TimeX++ (Liu et al., 2024), TimeSeg (Kim et al., 2026), and ORTE (Yue et al., 2025).

Every method in Table 34 retains exactly B units. The exact fixed-size reference enumerates all sets in that cardinality layer and is uncapped, unlike budgeted FutureCert recovery in Section 5.3. The capacity quantity $\Pr(k^\star \leq B)$ instead allows at most B units, and need not equal the exact-size pass rate under a nonmonotone predicate. With two units, KernelSHAP has 85.2% action retention but 22.6% preference-certificate passes; exact enumeration of the two-unit layer reaches 43.6% preference passes.

E.3 Finer Units on the Same Planning Problem

The 102-unit experiments change the planning problem (horizon 16), so they do not isolate granularity. Here the planning problem is unchanged: horizon 3, six MPPI iterations, and the same inputs, warm starts, and streams. Each of the twelve units is split into its two iterations, giving 24 units; a

Table 34: Pass rates at fixed evidence size. Every method retains exactly B units; the exact reference enumerates that cardinality layer. Exact fixed-size audit is an exhaustive reference, not budgeted FutureCert; it is excluded from method-ranking boldface.

Method	Cert.@1 (%) ↑	B=2(%)↑	B=3(%)↑	Action@2 (%) ↑
TimeSeg	8.5	12.9	36.7	79.7
ORTE	10.2	8.8	33.9	73.9
TIMING	11.0	17.5	41.1	81.3
TimeX++	10.9	15.1	42.8	81.7
KernelSHAP	**18.0**	22.6	46.3	85.2
LIME	15.1	19.5	43.9	83.9
RISE	17.2	18.4	42.6	**91.6**
Singleton gain	9.1	9.2	33.1	86.5
Mask regression	15.8	**23.1**	44.5	85.6
Position coverage	11.9	10.8	35.2	74.5
Interval-mask regression	16.4	22.7	45.8	84.8
Shallow-first	11.9	8.6	35.0	79.8
Deep-first	8.2	12.3	35.4	80.6
Center-first	11.0	12.9	40.8	80.2
Deletion divergence	13.2	19.8	**50.1**	86.3
Policy divergence	10.7	11.5	35.9	79.3
Reward/value deviation	12.1	16.3	42.4	79.7
RewardSal	11.7	15.0	36.4	80.9
ValueSal	12.2	14.5	41.5	74.8
Retain/delete heuristic	8.8	9.7	32.9	91.3
Diversity heuristic	8.7	11.8	33.6	90.7
FutureCert (Surrogate-Greedy)	12.7	14.4	36.2	80.8
FutureCert (Surrogate-Search)	12.7	16.3	38.3	82.0
FutureCert (Beam-4)	-	-	-	-
Random	9.8	12.1	35.7	79.1
Exact fixed-size audit	26.6	43.6	69.3	95.5

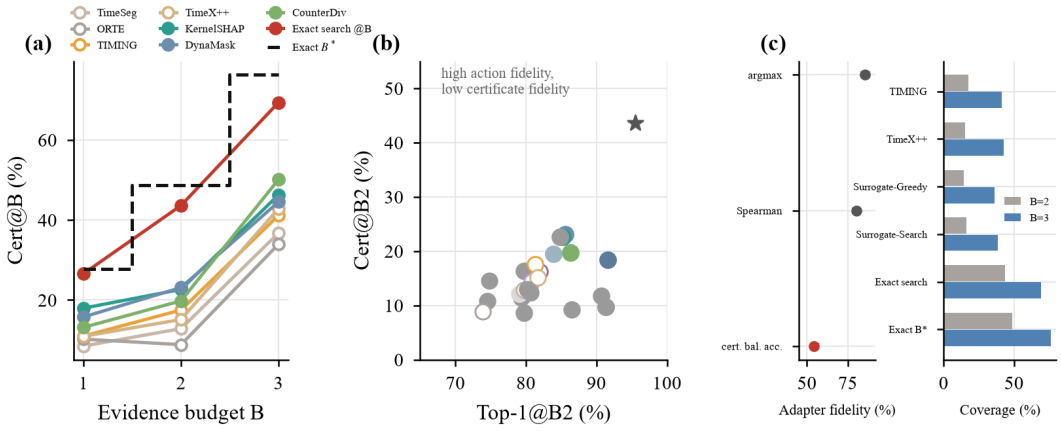


Figure 8: Action retention and preference-certificate pass rates at fixed evidence size. Each proposed set is tested by full planner re-execution; the exact reference checks every set in the same cardinality layer. Evidence size differs from the planner-call cap in Section 5.3. Labels $B^\star$ denote the minimum size $k^\star$. DynaMask, CounterDiv, and Exact search @B correspond to Mask regression, Deletion divergence, and Exact fixed-size audit in Table 34.

removed fine unit inherits its parent's reference, so the lift of every 12-unit set reproduces the coarse outcome bitwise, which we check on all 4,096 sets of the pilot inputs and on every lifted query of the full run. The fine search charges the lifted default FutureCert run, then applies deletion descent and the stochastic verified beam with the remaining budget; no 24-unit prior is trained. Table 35 shows that finer units retain 20.4% instead of 25.4% of the read groups within 384 queries, strictly fewer for 66% of inputs and fewer than the lifted exact 12-unit minimum for 59%. On a frozen subset enumerated exhaustively up to four fine units (Table 36), every uncensored non-empty minimum is smaller than the lifted coarse minimum and none of these is the lift of a coarse set; the search attains the fine minimum on 5 of 7 uncensored inputs at cap 384.

Table 35: **Finer evidence units on the same planning problem.** Each two-iteration stage of TD-MPC2 is split into single MPPI iterations (24 units); removed fine units inherit the parent reference, so a lifted 12-unit set reproduces the coarse outcome bitwise (117,482 online checks, no mismatch). The fine search starts from the lifted default FutureCert run, whose queries are charged, then applies deletion descent and the stochastic verified beam within the same cap. Fractions are retained fine read-groups; "strictly fewer" compares with the lifted FutureCert certificate and with the lifted exact 12-unit minimum. 120 inputs (60 per task); mean $\pm$ SD over five solver seeds. All returned sets are natively valid. Exact 24-unit optima are unavailable at this scale.

Cap	Task	Fine	Lifted FutureCert	Lifted exact	Fewer (%)	Fewer than exact (%)	Calls
384	Cheetah Run	0.173 ± 0.004	0.229 ± 0.005	0.218	73.0 ± 0.7	66.0 ± 1.5	220.3 ± 1.2
	Cup Catch	0.235 ± 0.004	0.278 ± 0.004	0.264	59.3 ± 1.9	52.3 ± 1.9	324.4 ± 1.0
	Macro	0.204 ± 0.004	0.254 ± 0.004	0.241	66.2 ± 0.7	59.2 ± 1.3	272.3 ± 1.0
768	Cheetah Run	0.143 ± 0.005	0.229 ± 0.005	0.218	83.3 ± 1.2	80.3 ± 1.8	291.2 ± 4.0
	Cup Catch	0.215 ± 0.005	0.278 ± 0.004	0.264	68.0 ± 1.8	61.0 ± 2.2	504.9 ± 4.9
	Macro	0.179 ± 0.005	0.254 ± 0.004	0.241	75.7 ± 1.4	70.7 ± 1.8	398.1 ± 3.6

Table 36: **Exhaustive 24-unit minima on a frozen subset.** First valid fine layer by complete enumeration up to four units (>4: censored), the exact 12-unit minimum $k^\star_{12}$ and its lifted size $2k^\star_{12}$, and the fine search's sizes for five seeds at caps 384 and 768. No fine minimum is the lift of a coarse set.

Input	Fine $k^\star$	$k^\star_{12}$	$2k^\star_{12}$	Search, cap 384	Search, cap 768
Cheetah Run-3	2	2	4	2, 2, 2, 2, 2	2, 2, 2, 2, 2
Cheetah Run-44	2	2	4	2, 2, 2, 2, 2	2, 2, 2, 2, 2
Cheetah Run-82	>4	4	8	7, 7, 7, 7, 7	7, 7, 7, 7, 7
Cheetah Run-122	3	2	4	3, 3, 3, 3, 3	3, 3, 3, 3, 3
Cheetah Run-161	>4	4	8	8, 8, 8, 8, 8	8, 8, 8, 6, 8
Cup Catch-2	4	3	6	5, 5, 5, 5, 5	5, 4, 4, 4, 5
Cup Catch-42	3	3	6	3, 3, 3, 3, 3	3, 3, 3, 3, 3
Cup Catch-84	2	2	4	2, 2, 2, 2, 2	2, 2, 2, 2, 2
Cup Catch-120	4	3	6	5, 5, 5, 5, 5	4, 4, 4, 4, 4
Cup Catch-170	>4	4	8	7, 7, 7, 7, 7	7, 7, 6, 6, 6

F Budgeted Certificate Recovery

This appendix expands Section 5.3 by comparing search policies within the fixed native verifier. Recovery quality, realized query cost, restart variability, and preparation costs are reported under their corresponding protocols.

F.1 Comparisons under Matched Query Budgets

Table 34 reports the complete fixed-size comparison. The following recovery table collects all nine settings under the matched online cap.

All following recovery comparisons use a 384-query cap; the FutureCert (Beam-4) variant retains its original stopping and deletion rules. Default FutureCert entries average ten solver seeds; parenthesized standard deviations measure variation of the episode-weighted mean across full solver executions, not across restarts within an execution. The nine-setting macro average is computed after

Table 37: **Fixed-size selection on TD-MPC2.** Full-fidelity pass rate (%, ↑) of a set with exactly B units, task macro over Cheetah Run, Cup Catch, and Finger Spin. The fixed-size variant of FutureCert searches the $|S| = B$ layer with the native verifier under the stated query cap. The exact fixed-size audit enumerates the whole layer and is a reference, not a method. Queries are mean online native queries per decision, averaged over the four sizes.

Method	$B = 1$	$B = 2$	$B = 3$	$B = 6$	Queries ↓
KernelSHAP	27.0	27.8	28.7	32.5	18.8
LIME	25.5	29.5	30.3	41.0	19.2
RISE	24.5	28.3	33.3	46.0	18.5
TIMING	24.3	29.3	36.2	48.0	1.0
TimeX++	18.5	19.7	20.7	21.7	1.0
TimeSeg	27.5	31.8	36.3	42.8	1.0
ORTE	21.8	21.5	21.5	25.7	1.0
FutureCert fixed-size, cap 8	35.3	48.3	58.3	56.2	4.9
FutureCert fixed-size, cap 32	36.0	54.5	70.0	60.8	13.0
FutureCert fixed-size, no cap	36.0	57.3	80.0	99.3	63.3
Exact fixed-size audit	*36.0*	*57.3*	*80.0*	*99.3*	*402.5*

equal episode weighting within each setting. Mean online calls are 36.23 for default FutureCert and 35.64 for its FutureCert (Beam-4) variant.

Table 38: **Matched-budget recovery across all nine settings.** Every solver uses the same native verifier and 384-query cap. Parentheses contain SD over ten default solver seeds; other solvers are deterministic. Recovery is in percent; Gap and Queries are counts. All point estimates are shown without significance ranking.

Setting	Method	Recovery (%)	Gap	Queries
MuZero Pong	FutureCert	99.21 (0.32)	0.015 (0.006)	24.77 (0.23)
MuZero Pong	FutureCert (Beam-4)	99.43	0.011	21.48
MuZero Pong	FutureCert (Native greedy)	96.31	0.054	13.44
MuZero Pong	FutureCert (Enumeration)	97.45	0.057	31.95
MuZero Pong	KernelSHAP	95.45	0.130	39.56
MuZero Breakout	FutureCert	100.00 (0.00)	0.000 (0.000)	12.68 (0.02)
MuZero Breakout	FutureCert (Beam-4)	98.64	0.027	14.62
MuZero Breakout	FutureCert (Native greedy)	95.73	0.096	10.62
MuZero Breakout	FutureCert (Enumeration)	100.00	0.000	13.47
MuZero Breakout	KernelSHAP	95.97	0.088	26.15
SEZ Pong	FutureCert	99.84 (0.14)	0.005 (0.004)	12.59 (0.02)
SEZ Pong	FutureCert (Beam-4)	99.23	0.021	14.91
SEZ Pong	FutureCert (Native greedy)	93.23	0.130	11.27
SEZ Pong	FutureCert (Enumeration)	100.00	0.000	12.73
SEZ Pong	KernelSHAP	94.29	0.146	36.97
Cheetah Run	FutureCert	93.05 (1.01)	0.103 (0.021)	81.64 (0.78)
Cheetah Run	FutureCert (Beam-4)	84.00	0.315	87.21
Cheetah Run	FutureCert (Native greedy)	52.50	1.400	46.17
Cheetah Run	FutureCert (Enumeration)	78.50	1.675	176.43
Cheetah Run	KernelSHAP	35.00	2.395	209.29
Cup Catch	FutureCert	86.10 (1.29)	0.315 (0.048)	122.02 (1.77)
Cup Catch	FutureCert (Beam-4)	75.50	0.605	108.47
Cup Catch	FutureCert (Native greedy)	51.00	1.525	55.21
Cup Catch	FutureCert (Enumeration)	67.00	2.545	218.84
Cup Catch	KernelSHAP	31.00	2.165	214.14
Finger Spin	FutureCert	99.75 (0.26)	0.003 (0.003)	5.19 (0.08)
Finger Spin	FutureCert (Beam-4)	100.00	0.000	12.65
Finger Spin	FutureCert (Native greedy)	98.00	0.020	9.05
Finger Spin	FutureCert (Enumeration)	100.00	0.000	10.71
Finger Spin	KernelSHAP	98.00	0.020	94.25

Continued on next page

Table 38: Matched-budget recovery, continued.

Setting	Method	Recovery (%)	Gap	Queries
MuZero MsPacman	FutureCert	99.10 (0.29)	0.030 (0.010)	34.53 (0.19)
MuZero MsPacman	FutureCert (Beam-4)	99.13	0.020	29.10
MuZero MsPacman	FutureCert (Native greedy)	99.14	0.020	16.83
MuZero MsPacman	FutureCert (Enumeration)	97.98	0.058	45.48
MuZero MsPacman	KernelSHAP	97.70	0.057	45.75
SEZ MsPacman	FutureCert	99.87 (0.14)	0.003 (0.003)	15.45 (0.03)
SEZ MsPacman	FutureCert (Beam-4)	99.74	0.005	15.84
SEZ MsPacman	FutureCert (Native greedy)	97.12	0.063	11.60
SEZ MsPacman	FutureCert (Enumeration)	100.00	0.000	15.57
SEZ MsPacman	KernelSHAP	97.88	0.048	21.63
EfficientZero MsPacman	FutureCert	100.00 (0.00)	0.000 (0.000)	17.16 (0.09)
EfficientZero MsPacman	FutureCert (Beam-4)	99.72	0.006	16.47
EfficientZero MsPacman	FutureCert (Native greedy)	97.24	0.063	11.67
EfficientZero MsPacman	FutureCert (Enumeration)	99.45	0.006	19.15
EfficientZero MsPacman	KernelSHAP	98.63	0.038	21.59

Table 39: **FutureCert minus FutureCert (Beam-4) at equal total cap: paired 95% intervals.** Recovery differences are percentage points; negative Gap and Calls favor FutureCert. Bootstrap crosses solver seeds and paired episode/decision resampling (3,000 replicates).

Setting	ΔRec. ↑	ΔGap ↓	ΔCalls ↓
MuZero Pong	-0.22 [-1.27, +0.95]	+0.00 [-0.02, +0.03]	+3.28 [+0.42, +6.35]
MuZero Breakout	+1.36 [+0.25, +2.99]	-0.03 [-0.06, -0.01]	-1.94 [-2.41, -1.45]
SEZ Pong	+0.61 [+0.00, +1.70]	-0.02 [-0.05, +0.00]	-2.32 [-2.99, -1.66]
Cheetah Run	+9.05 [+4.20, +14.45]	-0.21 [-0.35, -0.09]	-5.56 [-11.27, +0.15]
Cup Catch	+10.60 [+6.05, +15.80]	-0.29 [-0.47, -0.15]	+13.56 [+6.62, +21.29]
Finger Spin	-0.25 [-1.20, +0.00]	+0.00 [+0.00, +0.01]	-7.46 [-10.90, -5.04]
MuZero MsPacman	-0.03 [-1.18, +1.01]	+0.01 [-0.03, +0.06]	+5.44 [+2.94, +8.43]
SEZ MsPacman	+0.13 [+0.00, +0.62]	-0.00 [-0.01, +0.00]	-0.39 [-0.93, +0.22]
EfficientZero MsPacman	+0.28 [+0.00, +1.39]	-0.01 [-0.03, +0.00]	+0.70 [-0.17, +1.87]

F.2 Restart Sensitivity and Variance

Table 40 keeps the frozen stochastic-beam rule and 96 fresh queries per restart. Over the eight settings in the table, recovery means are 95.35/97.15/98.07% for $R = 2/4/8$, with Gap 0.154/0.059/0.035 and calls 34.26/40.11/44.11. Caps differ (192/384/768), so these are resource-quality measurements rather than equal-budget wins. FutureCert remains the default. A fixed shared restart prefix preserves the best verified incumbent: $|\widehat{S}_{R+1}| \leq |\widehat{S}_R|$. The property assumes that the earlier cache and incumbent trajectory are preserved and gives no global optimality or cross-planner-stream guarantee.

Same-resource decomposition. Table 41 crosses the development prior, one 384-query run versus four 96-query restarts, and the online additive ordering rule, all at the same 384-query cap and verifier on the 400 Cheetah Run and Cup Catch decisions. With the prior and ordering fixed, restarts add 4.55 and 6.10 recovery points on Cheetah Run and Cup Catch; with restarts and ordering fixed, the prior adds 1.05 and 1.60 points. The ordering rule changes recovery by $+0.05$ and -0.35 points, so we attribute no consistent quality gain to it. The full configuration reproduces the archived default outputs for all 400 decisions and ten seeds.

F.3 Candidate-policy Ablations

Source snapshot. This section preserves the candidate-ordering confirmation archive and its configuration-specific intervals. Main results use the unified re-evaluation of Appendix F.1; its source manifest records small numerical differences in the regression controls.

Table 40: **Restart sensitivity on eight settings.** Means (solver-seed SD) over ten executions; recovery and its SD are percentages and percentage points. Each restart permits 96 fresh queries; caps are $96R$. Finger Spin has only the default $R = 4$ result in the matched-budget study.

Setting	R	Cap	Rec. ↑	Gap ↓	Calls ↓
MuZero Pong	2	192	98.47 (0.19)	0.032 (0.006)	21.87 (0.15)
	4	384	99.21 (0.32)	0.015 (0.006)	24.77 (0.23)
	8	768	99.74 (0.21)	0.005 (0.004)	27.15 (0.35)
MuZero Breakout	2	192	100.00 (0.00)	0.000 (0.000)	12.56 (0.02)
	4	384	100.00 (0.00)	0.000 (0.000)	12.68 (0.02)
	8	768	100.00 (0.00)	0.000 (0.000)	12.78 (0.02)
SEZ Pong	2	192	99.76 (0.15)	0.007 (0.004)	12.43 (0.03)
	4	384	99.84 (0.14)	0.005 (0.004)	12.59 (0.02)
	8	768	99.92 (0.13)	0.002 (0.004)	12.70 (0.03)
Cheetah Run	2	192	88.70 (0.67)	0.394 (0.058)	68.81 (0.35)
	4	384	93.05 (1.01)	0.103 (0.021)	81.64 (0.78)
	8	768	95.50 (0.62)	0.055 (0.006)	90.37 (1.04)
Cup Catch	2	192	78.00 (1.60)	0.710 (0.055)	97.01 (0.95)
	4	384	86.10 (1.29)	0.315 (0.048)	122.02 (1.77)
	8	768	90.20 (1.23)	0.195 (0.035)	138.73 (2.62)
MuZero MsPacman	2	192	98.12 (0.28)	0.081 (0.006)	29.67 (0.07)
	4	384	99.10 (0.29)	0.030 (0.010)	34.53 (0.19)
	8	768	99.25 (0.28)	0.023 (0.009)	37.90 (0.31)
SEZ MsPacman	2	192	99.79 (0.11)	0.004 (0.002)	15.23 (0.03)
	4	384	99.87 (0.14)	0.003 (0.003)	15.45 (0.03)
	8	768	99.97 (0.08)	0.001 (0.002)	15.63 (0.03)
EfficientZero MsPacman	2	192	99.94 (0.12)	0.001 (0.001)	16.53 (0.07)
	4	384	100.00 (0.00)	0.000 (0.000)	17.16 (0.09)
	8	768	100.00 (0.00)	0.000 (0.000)	17.64 (0.05)
Eight-setting macro	2	192	95.35 (0.24)	0.154 (0.011)	34.26 (0.13)
	4	384	97.15 (0.16)	0.059 (0.006)	40.11 (0.26)
	8	768	98.07 (0.14)	0.035 (0.005)	44.11 (0.38)

Table 41: **Same-resource decomposition of the solver on Cheetah Run and Cup Catch.** Every configuration has the same 384-query cap and the same verifier. Recovery (%) is the mean over ten solver seeds (SD in parentheses); single-run configurations are deterministic. Calls are mean online queries for Cheetah Run / Cup Catch. The bold row is the default FutureCert. Ordering is the online additive single-unit ordering rule.

Prior	Restarts	Ordering	Cheetah Rec.↑	Cup Rec.↑	Calls↓
off	1×384	off	84.00 (0.00)	75.00 (0.00)	76.5 / 95.2
off	1×384	on	84.00 (0.00)	74.50 (0.00)	65.0 / 89.8
off	4×96	off	91.95 (0.60)	84.55 (1.67)	92.8 / 120.7
off	4×96	on	92.00 (0.62)	84.50 (1.62)	80.4 / 114.7
on	1×384	off	88.50 (0.00)	80.00 (0.00)	69.0 / 99.5
on	1×384	on	88.50 (0.00)	80.00 (0.00)	66.8 / 97.3
on	4×96	off	93.00 (0.58)	86.45 (1.19)	83.9 / 124.1
on	4×96	**on**	93.05 (1.01)	86.10 (1.29)	81.6 / 122.0

Candidate policies. This refinement changes TD-MPC2 candidate ordering while retaining the native verifier. The matched policies share the initial native path, candidate pool, cached-mask skipping, query cap, and final deletion stage. Size-first distance, fixed-random, and prototype-frequency orders isolate within-cardinality prioritization. Frequency counts distinct source decisions, merging duplicates across development libraries. The mixed queue preserves the original nearest-prefix and cardinality priorities under the shared queue handling. The prototype solver is reported separately to expose the effect of common cache and stopping changes.

Utility and queried feedback. For candidate S, training decisions yield the Laplace-smoothed validity prior $p_0(S) = (n_{\text{valid}}(S)+1)/(n_{\text{train}}+2)$. Static utility ranks by $p_0(S)\max(|\widehat{S}|-|S|,0)$. Feedback utility uses the same current incumbent size but updates its validity score from already queried masks. Observation Q receives Hamming-kernel weight $\exp(-|S \triangle Q|/2)$ multiplied by a violation weight $1+\min(\max(\max(d_a,d_\mu)/0.2-1,0),3)$. The score is the weighted mean of observed binary validity and the prior with weight 2. This is a ranking heuristic, not a calibrated probability. Feedback changes ordering without pruning unqueried subsets or supersets. All accepted sets pass the native verifier.

Development selection and retrospective evaluation. Each task has 50 development decisions in five episodes. Each fold rebuilds its library and validity priors from four episodes and evaluates the fifth. Macro gap, recovery, then policy simplicity determine the frozen choice: feedback at caps 16/32 and static utility at 96. Both utility policies use the same incumbent updates, isolating feedback in their comparison. The selected rule is replayed on the original 400 Cheetah Run and Cup Catch confirmation decisions and the existing 100 independent Cup Catch decisions; no new episodes are collected for this refinement. $k^\star$ is used only after search for scoring. All candidate policies and controls are retained in the report; the selected rule is not changed after seeing evaluation results.

Table 42: Complete candidate-ordering comparison on the 400 Cheetah Run and Cup Catch confirmation decisions. Rec. is exact-size recovery (%). The development-selected policies are feedback at cap 32 and static utility at cap 96. Bold marks the best unrounded column value.

	32-call cap			96-call cap		
Policy	Rec.↑	Gap↓	Calls↓	Rec.↑	Gap↓	Calls↓
Prototype solver	26.8	3.007	31.01	62.5	1.275	85.77
Matched mixed queue	26.8	3.010	31.20	62.7	1.280	86.04
Size + distance	26.7	3.120	31.20	62.7	1.305	86.04
Size + random	24.8	3.188	31.20	62.7	1.297	86.04
Size + frequency	27.0	3.093	31.20	63.0	1.290	86.04
Utility (static prior)	**33.5**	2.695	31.20	64.0	**1.113**	86.04
Utility + feedback	33.0	**2.672**	31.20	63.5	1.125	86.04
FutureCert (Enumeration)	15.0	7.520	**29.34**	40.0	5.037	75.72
FutureCert (Beam-4)	30.8	5.945	30.09	**66.5**	2.538	**73.82**
KernelSHAP	19.0	5.658	29.76	35.2	2.287	73.97

Results and incremental effects. At caps 16/32/96, the frozen selected rule recovers 11.8/33.0/64.0%, with gaps 4.527/2.672/1.113 and 15.67/31.20/86.04 calls. The prototype solver gives 10.5/26.8/62.5%, gaps 4.550/3.007/1.275, and 15.69/31.01/85.77 calls. The paired 32-call recovery gain is 6.2 points [1.5, 11.0]; paired gap differences at 32/96 are -0.335 $[-0.555, -0.125]$ and -0.163 $[-0.300, -0.023]$. At 32 calls static utility recovers 33.5% and feedback 33.0%, while feedback reduces gap from 2.695 to 2.672. Thus improved utility ordering and the incremental value of online feedback are separate comparisons.

On the existing independent Cup Catch sample, the frozen rule recovers 9/21/53% at caps 16/32/96, versus 9/20/51% for the prototype solver. Its gaps are 3.880/2.720/1.360, versus 3.810/3.000/1.480. At cap 96 it uses 92.71 calls versus 92.18. The paired gap difference is -0.120 $[-0.310, 0.010]$, so this sample gives a smaller and uncertain improvement. The cap-16 gap is slightly worse. All intervals use 2,000 hierarchical episode-decision bootstrap draws and are not corrected for multiple comparisons.

Offline and online cost. Fitting the prior across all 50 development decisions queries 81, 105, and 65 distinct candidate masks per decision, totaling 4,050/5,250/3,250 gross labels across the three tasks. These can overlap the original prototype preparation labels, so the totals are not additive marginal costs. Evaluation tables charge only newly proposed non-full masks. CPU replay-plus-ordering timing is reported separately from physical planner runtime. Query traces, frozen source snapshots, development folds, per-task/all-cap results, and paired comparisons are stored in the development-selection and confirmation records in the result archive.

F.4 Online and Offline Cost

The frozen source bank uses 16,995, 18,629, and 15,368 distinct non-full native queries for Cheetah Run, Cup Catch, and Finger Spin, including the union of source certificates and validity-prior fitting. Fifty shared full targets per task are additional if charged. Over N new decisions, the amortized count is $q_{\text{online}} + Q_{\text{prep}}/N$. These labels are available to default FutureCert; FutureCert (Beam-4) and FutureCert (Native greedy) use no source bank. Exploratory model-selection runs and exact evaluation audits are separate. All current online counts are in Appendix F.1.

Larger FutureCert (Beam-4) caps and break-even. Raising FutureCert (Beam-4)'s cap to 477 (384 plus the largest amortized bank cost) or 768 leaves its returned sets and calls unchanged on all 600 TD-MPC2 decisions, because its original stopping rule ends the search before either cap; we did not alter that rule, so this does not exclude better-performing beam variants with larger budgets. Table 43 counts offline and online queries with one rule: every distinct native execution counts once, and repeated identical queries are served from cache.

Table 43: **Offline preparation and query break-even.** Offline queries are distinct native executions for the frozen development bank, including validity-prior fitting (the same counting rule as online queries, where repeated identical queries are cached). Per decision amortizes over the 200 evaluation decisions per task. Break-even is the number of audited decisions after which the offline queries are repaid by online savings relative to FutureCert (Beam-4); "none" means FutureCert uses more online queries. It is a query-count ledger, not a time or equal-quality guarantee. FutureCert (Beam-4) returns identical sets and calls at caps 384, 477, and 768.

Here offline counts exclude the 50 shared full targets per task; Table 2 includes them in its preparation totals.

Task	Offline queries	Per decision	FutureCert online	FutureCert (Beam-4) online	Break-even N
Cheetah Run	16,995	85.0	81.6	87.2	3,056
Cup Catch	18,629	93.1	122.0	108.5	none
Finger Spin	15,368	76.8	5.2	12.6	2,061

Current realized calls accompany the matched-budget tables in Appendix F.1; pipeline and reuse timings are in Appendix D.

F.5 Per-setting Explanation Quality

Table 2 summarizes TD-MPC2 recovery and preparation; Appendix F.1 gives every setting, seed variability, and paired comparisons.

F.6 Independent Episodes and Input Deduplication

The FutureCert re-evaluation uses 96 eligible Breakout inputs and 100 Cup Catch inputs from ten previously collected episodes per task. Each complete solver execution has four 96-query restarts, shared cache, and total cap 384. Table 44 averages ten solver seeds. These samples are independent of the original evaluation episodes but were collected before FutureCert was frozen; post-freeze trajectories are in Table 46.

We recompute SHA256 hashes from the original observation arrays and exclude byte-identical development observations. Of 1,094 eligible core inputs, 931 remain: 297 Pong, 260 Breakout, and 374 SEZ Pong inputs. Completed FutureCert runs retain all ten solver seeds; every comparator retains its original 384-query implementation. This is a filtered reanalysis, with no new planner

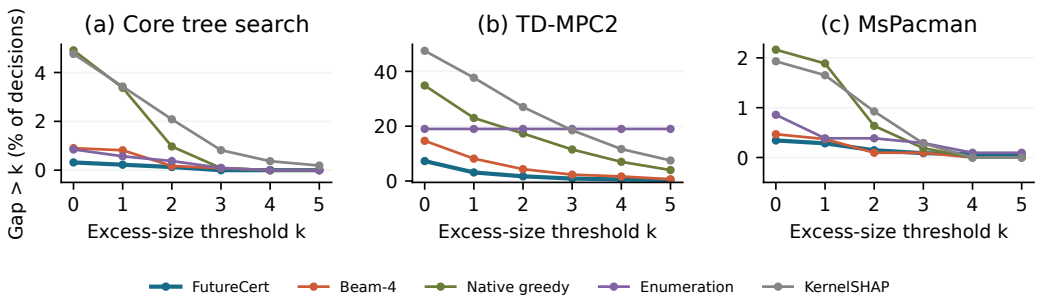


Figure 9: **Excess certificate-size error at equal total query cap.** Each curve reports the percentage with Gap greater than k (↓), weighting episodes and then tasks equally. FutureCert averages ten solver seeds; all methods use the same native verifier and 384-query cap.

Table 44: **Independent-episode FutureCert re-evaluation at cap 384.** Mean (SD across ten solver seeds), episode-equal aggregation. Rec. is percent and its SD is pp. Both rows describe FutureCert; different tasks are not ranked.

Task	Inputs	Rec. ↑	Gap ↓	Calls ↓
MuZero Breakout	96	100.00 (0.00)	0.000 (0.000)	10.03 (0.06)
TD-MPC2 Cup Catch	100	81.70 (2.26)	0.399 (0.042)	138.99 (1.21)

executions or policy changes. Episode weights are recomputed after filtering. It excludes byte-identical development observations, not all similar states or every possible dependence in planning context. FutureCert recovery changes from 99.683% to 99.668%, with Gap changing from 0.00670

Table 45: **Current FutureCert recovery after observation-collision exclusion.** Core three-setting macro means, before and after filtering. All methods use a 384-query cap; FutureCert averages ten solver seeds. Bold marks the best point estimate within each population.

	All eligible inputs ($n = 1,094$)			No development match ($n = 931$)		
Method	Rec.% ↑	Gap ↓	Calls ↓	Rec.% ↑	Gap ↓	Calls ↓
FutureCert	**99.68**	**0.00670**	16.68	**99.67**	**0.00703**	16.10
FutureCert (Beam-4)	99.10	0.01975	17.00	99.16	0.01849	16.51
FutureCert (Native greedy)	95.09	0.09337	**11.78**	95.23	0.09031	**11.50**
FutureCert (Enumeration)	99.15	0.01888	19.38	99.11	0.01970	18.88
KernelSHAP	95.24	0.12120	34.22	95.85	0.11248	35.16

to 0.00703. Against FutureCert (Beam-4) on the filtered subset, the paired recovery difference is $+0.509$ percentage points (95% interval $[-0.121, 1.226]$), and the Gap difference is -0.01146 ($[-0.02741, 0.00202]$). These intervals use 3,000 crossed solver-seed and paired episode-decision bootstrap draws. Both quality intervals against Beam and enumeration cross zero; those against FutureCert (Native greedy) and KernelSHAP favor FutureCert. The result checks exact-observation overlap in the existing archive rather than establishing prospective generalization.

F.7 Larger Evidence Spaces and New Tasks

Hopper Hop: FutureCert minus comparator size, with paired 95% intervals: FutureCert (Beam-4): -0.020 [-0.080, +0.000]; FutureCert (Native greedy): -0.020 [-0.080, +0.000]; KernelSHAP: -0.040 [-0.120, +0.000]. Quadruped Run: FutureCert minus comparator size, with paired 95% intervals: FutureCert (Beam-4): -0.332 [-0.788, -0.032]; FutureCert (Native greedy): -2.472 [-3.372, -1.568]; KernelSHAP: -3.512 [-4.420, -2.536].

For horizon h and six optimization iterations, unit (t, i) denotes reward at step t and iteration i, plus terminal-value groups, giving $6(h+1)$ units. Reward and value replacement preserve calls that consume random numbers. References average full-run predictions over all 50 development obser-

Table 46: Task transfer under three separate frozen protocols. The 50-input cohort uses the current default and cap 384, with five default solver seeds. The earlier 25- and 100-input cohorts use FutureCert (Utility-96) and cap 96. Cohorts are not pooled; all returned sets pass their native predicate. No exact minimum is available.

Cohort	Task	Method	Size	Non-full (%)	Queries
50 inputs, cap 384	Hopper Hop	FutureCert	0.760	100.0	7.04
50 inputs, cap 384	Hopper Hop	FutureCert (Beam-4)	0.780	100.0	19.50
50 inputs, cap 384	Hopper Hop	FutureCert (Native greedy)	0.780	100.0	11.38
50 inputs, cap 384	Hopper Hop	KernelSHAP	0.800	100.0	104.42
50 inputs, cap 384	Quadruped Run	FutureCert	4.468	99.6	185.79
50 inputs, cap 384	Quadruped Run	FutureCert (Beam-4)	4.800	98.0	142.52
50 inputs, cap 384	Quadruped Run	FutureCert (Native greedy)	6.940	92.0	70.96
50 inputs, cap 384	Quadruped Run	KernelSHAP	7.980	86.0	226.40
100 inputs, cap 96	Hopper Hop	FutureCert (Utility-96)	0.56	100.0	9.2
100 inputs, cap 96	Hopper Hop	FutureCert (Native greedy)	0.56	100.0	7.8
100 inputs, cap 96	Hopper Hop	KernelSHAP	0.60	100.0	25.1
100 inputs, cap 96	Quadruped Run	FutureCert (Utility-96)	7.59	92.0	95.1
100 inputs, cap 96	Quadruped Run	FutureCert (Native greedy)	7.18	92.0	71.8
100 inputs, cap 96	Quadruped Run	KernelSHAP	8.30	88.0	87.5
25 inputs, cap 96	Hopper Hop	FutureCert (Utility-96)	1.12	96.0	14.9
25 inputs, cap 96		FutureCert (Native greedy)	1.12	96.0	14.2
25 inputs, cap 96		KernelSHAP	1.12	96.0	29.5
25 inputs, cap 96	Quadruped Run	FutureCert (Utility-96)	7.68	88.0	96.0
25 inputs, cap 96		FutureCert (Native greedy)	7.36	92.0	72.0
25 inputs, cap 96		KernelSHAP	9.00	92.0	90.3

Table 47: Actual horizon extension with 24/54/102 reward/value groups; five inputs per task/horizon. Ret. is retained evidence (%, ↓); Q is queries (↓), capped at 96. All full endpoints match native execution and returned sets pass verification. Large spaces have no exact-optimum reference.

		Cheetah Run		Cup Catch	
Horizon	Units	Ret. ↓	Q ↓	Ret. ↓	Q ↓
3	24	36.7	48.4	54.2	46.6
8	54	3.0	51.0	67.4	90.4
16	102	27.1	77.0	60.4	77.0

Table 48: 102-unit confirmation on 20 inputs per task (40 unique inputs, including the earlier 10). Both deletion orders use the same native verifier and 96-query cap. Ret., Selected, and Matched are percentages; Ret. measures retained evidence. Matched random sets have exactly the trace-order output size (four draws per input). Every returned search set and all native full endpoints pass their checks.

Setting	Trace ret. ↓	Rand. ret. ↓	Selected ↑	Matched ↑	Q: trace/rand. ↓
Cheetah Run	**28.4**	30.9	**100.0**	30.0	86.5/86.5
Cup Catch	**50.6**	51.6	**100.0**	38.8	72.2/72.2

vations at each new horizon. All controlled calls use a zero initial mean, including the unmodified native comparison. Five evaluation inputs per task (one per episode) are selected before execution. The same deletion rule and 96-query cap apply throughout. Mean retained fractions and calls are reported for every task/horizon pair; no exact-size metric is inferred at 54 or 102 units. Per-decision timings include concurrent GPU workloads and are not used for method speed comparisons.

The source pool unions Cheetah Run, Cup Catch, and Finger Spin development masks. Frequencies count source decisions; each prior averages the three source validity priors, assigning $1/52$ to an unobserved mask in a source task. Static utility ordering is unchanged. The new checkpoints are the available Hopper Hop and Quadruped Run models; this is a new task/model pair rather than a second training seed of an existing task. Reference episodes use seeds 74001-74005; evaluation episodes use 75001-75005, with 200 decisions per episode and five uniformly spaced sampled decisions. Ref-

erence trajectories supply only means, not target-task certificate labels. FutureCert (Native greedy) and KernelSHAP receive the same native query cap and full-set fallback.

The 102-unit confirmation uses the first four archived inputs in each of five evaluation episodes. It reuses the same horizon-16 development reference calculation and zero warm start as the initial experiment. Search methods have separate logical caches and 96-query caps; a shared physical cache only avoids repeated execution. Every sampled random-control mask has exactly the corresponding trace-order output size. The five earlier inputs per task are counted once in the expanded population.

The prospective confirmation collects new episode seeds 76001-76005 and 20 sampled inputs per episode. The initial cohort uses 75001-75005 and five inputs per episode. Both cohorts use the original 74001-74005 reference episodes and the same frozen source bank. The additional cohort is reported separately, so five- and twenty-sample episodes are not silently pooled into an equally weighted decision mean. The checkpoints are unchanged from the first transfer experiment; additional trajectories extend episode coverage, not training-seed coverage.

Cheetah Run: random-minus-trace retained fraction is 2.50 percentage points, with a paired 95% episode-decision interval [-2.31,7.21].

Cup Catch: random-minus-trace retained fraction is 0.93 percentage points, with a paired 95% episode-decision interval [-1.23,3.48].

Hopper Hop: FutureCert (Native greedy) minus FutureCert (Utility-96) mean size is 0.000, with a paired 95% episode-decision interval [0.000,0.000].

Hopper Hop: KernelSHAP minus FutureCert (Utility-96) mean size is 0.040, with a paired 95% episode-decision interval [0.000,0.110].

Quadruped Run: FutureCert (Native greedy) minus FutureCert (Utility-96) mean size is -0.410, with a paired 95% episode-decision interval [-1.180,0.460].

Quadruped Run: KernelSHAP minus FutureCert (Utility-96) mean size is 0.710, with a paired 95% episode-decision interval [-0.220,1.750].

F.8 Evidence Sizes and Preparation Cost

Table 49: Core tree-search decisions by evidence size H (a property of the evaluation data, independent of the solver). 95.3% of decisions have $H \leq 7$.

H	3	4	5	6-7	8-12	Total
Decisions	224	288	314	217	51	1,094

Table 50 records the earlier 16-96-query development protocol, with gross calls and deduplicated labels of development preparation for prototypes, router, and frozen surrogate. Prototype and router calls overlap and cannot be added. The prototype pool used at the displayed 16-96 call caps is an index over stored development certificates rather than a separately trained model. Rebuilding that index from frozen artifacts takes 0.181, 0.205, and 0.212 seconds for the three tree settings and 1.114 and 1.893 seconds for Cheetah Run and Cup Catch on the evaluation host (means of ten CPU builds). For Cheetah Run and Cup Catch the development banks contain 16,052 and 16,997 distinct masks including the full mask (16,002 and 16,947 non-full labels). The older tree-search records do not retain mask identities for every preparation stage; their union can only be bounded by 9,988-14,256, 10,873-16,367, and 11,839-17,238 calls. These earlier label ledgers are separate from the complete current serving-bank union reported in Appendix F; they are not alternative estimates of the same preparation run.

F.9 Anytime Recovery and Difficulty Strata

Table 51 replays the logged query order of the default solver against the exhaustive Cheetah Run and Cup Catch archive: recovery rises from 22.8% after 16 queries to 68.3% after 96 and 89.6% after 384 (task macro), and the replay reproduces every logged final output. Table 52 separates tree-search decisions by the size of their minimum certificate. Empty and full minima, 29.8-51.8% of decisions,

Table 50: Offline labels and query records during development. Units are planner calls or deduplicated non-full mask labels.

Prototype and router preparation

Setting	Prototype gross calls ↓	Router-expert gross calls ↓
MuZero Pong	9988	4268
MuZero Breakout	10873	5494
SEZ Pong	11839	5399

Surrogate preparation

Setting	Random-label calls ↓	Reused-enumeration gross calls ↓	Distinct non-full labels ↓
MuZero Pong	4321	18782	19238
MuZero Breakout	7673	6528	8920
SEZ Pong	7628	7282	10061

are recovered by every method; on the remaining decisions FutureCert, FutureCert (Beam-4), and enumeration recover 95-100%. Tree search therefore measures the audit rather than the solver.

Table 51: **Anytime recovery of the default solver on Cheetah Run and Cup Catch.** Recovery (%) / Gap of the incumbent after the first c charged queries of the default four-restart run, averaged over ten solver seeds and replayed from the logged query order against the exhaustive archive. The $c = 384$ column reproduces the logged solver output for all 4,000 runs. Points below 96 lie inside the deterministic first restart; this is an anytime profile of one run, not a separate run per cap.

Task	$c = 16$	32	64	96	192	384
Cheetah Run	31.5 / 6.05	52.5 / 3.58	67.5 / 1.88	77.0 / 1.30	90.2 / 0.38	93.0 / 0.10
Cup Catch	14.0 / 7.44	25.0 / 4.64	41.0 / 2.97	59.5 / 1.84	79.5 / 0.69	86.1 / 0.32
Two-task macro	22.8 / 6.75	38.8 / 4.11	54.3 / 2.42	68.3 / 1.58	84.9 / 0.53	89.6 / 0.21

Table 52: **Tree-search recovery by stratum at the 384-query cap.** Recovery (%) on decisions whose minimum certificate is empty ($k^\star = 0$), partial ($0 < k^\star < H$), or the full set ($k^\star = H$). Every method recovers the trivial strata; on partial decisions FutureCert, FutureCert (Beam-4), and enumeration are within 95-100%, so the solver's advantage is specific to TD-MPC2. FutureCert averages ten solver seeds; baselines are the matched-cap reruns behind Table 2.

Setting	Stratum	n	FutureCert	FutureCert (Beam-4)	FutureCert (Enumeration)	FutureCert (Native greedy)	KernelSHAP
MuZero Pong	$k^\star = 0$	12	100.0	100.0	100.0	100.0	100.0
	$0 < k^\star < H$	163	98.7	99.1	95.2	93.1	91.6
	$k^\star = H$	163	100.0	100.0	100.0	100.0	100.0
MuZero Breakout	$k^\star = 0$	55	100.0	100.0	100.0	100.0	100.0
	$0 < k^\star < H$	262	100.0	98.1	100.0	93.8	94.0
	$k^\star = H$	56	100.0	100.0	100.0	100.0	100.0
SEZ Pong	$k^\star = 0$	36	100.0	100.0	100.0	100.0	100.0
	$0 < k^\star < H$	250	99.7	98.7	100.0	89.4	91.2
	$k^\star = H$	97	100.0	100.0	100.0	100.0	100.0

G Certificate Structure and Repair

This section details certificate structure and repair in Section 5.4.

G.1 Minimum Families, Core, and Witnesses

Minimum-certificate size and solution family Figure 10 summarizes the exhaustive family statistics used in Section 5.4.

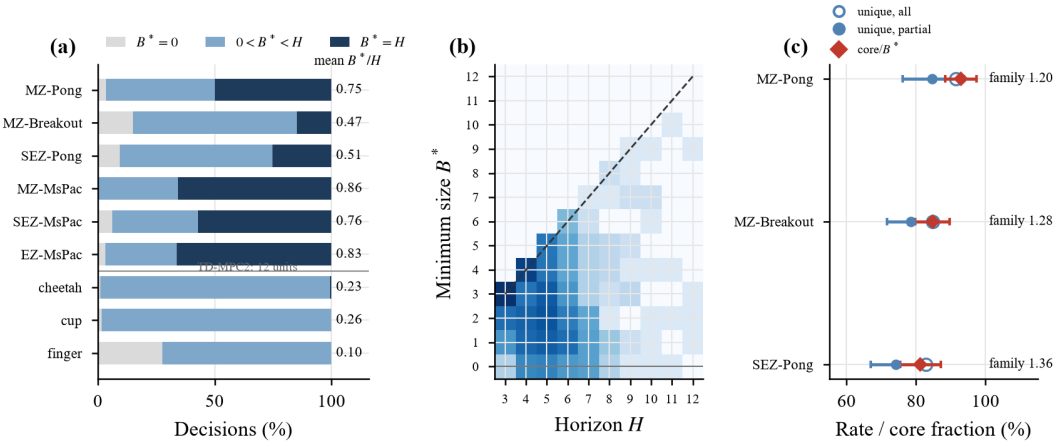


Figure 10: **Exact audits expose certificate size, alternatives, and shared evidence.** (a) Empty, partial, and full-retention decisions across nine settings; labels give mean $k^\star/H$. (b) Joint distribution of evidence-space size H and minimum size $k^\star$ in core tree search. (c) Unique-certificate rates among all and partial decisions, the Core fraction of a minimum certificate, and mean family size. Labels $B^\star$ denote the minimum size $k^\star$.

Fix a deterministic native predicate on a finite universe of H units. Suppose that the minimum cardinality $k^\star$ has already been proved independently, and a size-$k^\star$ certificate S_0 has been verified. Write $\mathcal{K} = \text{Core}$ and $c = |\mathcal{K}|$. The following statements require no monotonicity. They concern the size-$k^\star$ layer, not necessity across larger valid sets.

Proposition 5 (small intersection witness). There is a subfamily $\mathcal{W} \subseteq \mathcal{F}^\star$ containing S_0 such that $\bigcap_{S\in\mathcal{W}} S = \mathcal{K}$ and $|\mathcal{W}| \le k^\star - c + 1$. *Proof.* For each $u \in S_0 \setminus \mathcal{K}$, the definition of the intersection supplies some $S_u \in \mathcal{F}^\star$ excluding u. The family consisting of S_0 and these witnesses has intersection exactly $\mathcal{K}$. Duplicate witnesses can be removed. The bound can be attained: let $S_0 = \mathcal{K} \cup \{u_1, \dots, u_{k^\star - c}\}$ and supply, for each u_i, only the alternative replacing u_i by an outside unit. Every such alternative is then needed. This is a combinatorial representation result; it does not give a low-query way to find the witnesses.

Proposition 6 (minimum sufficient proof record). Let ρ_{S_0} be the smallest size of an intersection witness subfamily that contains S_0. In a membership-oracle model with no other known size-$k^\star$ labels, the smallest additional query record sufficient to determine $\mathcal{K}$ among all predicates consistent with that record has size

$$Q^*_{\text{record}} = \binom{H}{k^\star} - \binom{H-c}{k^\star-c} + \rho_{S_0} - 1. \tag{12}$$

Knowledge of $k^\star$ is a supplied premise; its proof cost is separate. The expression is an instance-specific minimum record length, not an algorithm that can discover this record without knowing the instance.

Proof. Every size-$k^\star$ set not containing $\mathcal{K}$ must have a recorded negative label. Otherwise changing its unqueried label to positive preserves all known labels and the proved minimum size but removes a member of $\mathcal{K}$. There are $\binom{H}{k^\star} - \binom{H-c}{k^\star-c}$ such sets. In addition, the recorded positive sets must intersect exactly in $\mathcal{K}$: if their intersection contained a further unit, setting all remaining unknown labels to negative would produce a consistent larger Core. This needs at least $\rho_{S_0} - 1$ additional positive labels. The required negative and positive labels are disjoint. Conversely, these two parts suffice: the negative labels force every possible optimum to contain $\mathcal{K}$, and the positive witnesses exclude every other unit from the intersection. This proves both bounds. If $\mathcal{K} = \varnothing$, the negative term is zero; if $k^\star = 0$, the known empty optimum already determines the Core.

Directed certification. Maintain a verified-witness intersection K_{possible} and a proved subset $K_{\text{confirmed}}$. For each unresolved unit u, query size-$k^\star$ sets excluding u. A valid counterexample shrinks K_{possible}; exhaustion of that restricted layer adds u to $K_{\text{confirmed}}$. A shared cache reuses earlier membership results, including the negatives obtained while finding the first optimum. At every cap,

$$K_{\text{confirmed}} \subseteq \mathcal{K} \subseteq K_{\text{possible}}.$$

Stopping before equality reports an interval, not a complete Core. This is related to counterexample-based backbone computation (Janota et al., 2015); our oracle is the native planning predicate on a proved optimum layer. It is an optional structural auditor, distinct from the FutureCert size-recovery solver. In particular, four stochastic restarts are not four distinct minimum witnesses.

Validation and cost. We enumerate every nonempty cardinality-layer family for $H \leq 4$ to check the witness bound, sufficient records against all compatible families, directed bounds under every cap, and single/pair unavailability answers. Native-archive replay then uses 1,094 core tree-search decisions. We distinguish the retrospectively identified time when an intersection first equals the true Core from the later time when the observed labels prove that equality. Only the latter is an operational stopping rule. Table 53 reports the 152 decisions with multiple minima; weights are equal across represented episodes. Full costs include discovering and proving $k^\star$. The additional directed column counts new queries after the first minimum, reusing that discovery cache; Equation 12 instead assumes only S_0 is known on its layer.

Table 53: **Few witnesses and proof costs measure different requirements.** Multiple-minimum cases only. Directed certification returns the exact Core in every case. Calls are logical native queries (↓); small cost reductions do not establish a general speedup.

Setting	n	Witnesses ρ_{S_0}	First correct	Directed added	Directed total	Full layer total
MuZero Pong	30	2.094	162.99	47.19	195.08	196.93
MuZero Breakout	56	2.000	13.31	6.22	14.32	15.56
SEZ Pong	66	2.026	13.79	7.65	16.92	19.57

These results support exact auditing as a completeness reference and targeted queries as an alternative when only a structural answer is needed. The complete family is still needed to list every alternative or compute exact family frequencies, but the Core alone need not require listing every optimum. The observed small witness counts do not make the proof costs disappear, and the measured directed savings are modest.

We enumerate in lexicographic order within increasing cardinality layers. A prefix stops after discovering k minimum certificates, or after exhausting the first valid layer if fewer exist. The full-output target is a shared zero-cost endpoint. Partial-family intersections are supersets of the complete Core and can only overstate shared membership. Exact-Core agreement is computed retrospectively using the completed archive; the prefix algorithm cannot infer completeness from this agreement. Cross-reference frequency MAE in Table 54 compares membership frequencies under the original reference bank with an independently recomputed family under a bank collected from separate reference episodes. Even the complete original family can therefore have nonzero error. This direct comparison differs from the fitted prediction errors in Table 80.

Unique-minimum cases number 308, 317, and 317 in MuZero Pong, MuZero Breakout, and SEZ Pong. Their first certificate equals the Core; further enumeration establishes uniqueness without changing its membership. The stratified source CSV retains all, unique, and multiple-minimum populations.

G.2 Certificates under Unavailable Evidence

Querying alternative explanations under unavailable evidence Suppose a previously audited minimum size is $k^\star$ and a set D of read groups is unavailable. The operational question is

$$A(D) = \mathbf{1}\{\exists S \in \mathcal{F}^\star : S \cap D = \varnothing\}. \tag{13}$$

For a single unavailable unit, $A(\{u\}) = 1$ exactly when $u \notin \text{Core}$. For several units, Core membership alone is insufficient: the family $\{\{1,2\},\{1,3\},\{2,3\}\}$ has empty Core but no certificate

Table 54: Few versus complete minimum certificates on decisions with multiple minima ($n = 30/56/66$). Prefixes follow lexicographic enumeration; exact Core is assessed afterward against the full audit. False shared counts excess units in the provisional intersection. Cross-reference frequency MAE compares with the family under a separate reference bank. Calls include the smaller failed layers and discovery of the displayed prefix. Recovering the true Core does not by itself prove completion.

Setting	Certificates	Exact Core ↑	False shared ↓	Cross-reference frequency MAE ↓	Calls ↓
MuZero Pong	1	0.0	1.300	0.214	145.3
	2	83.3	0.167	0.039	150.5
	4	**100.0**	0.000	0.004	178.6
	8	**100.0**	0.000	0.004	181.6
	All	**100.0**	0.000	0.004	181.6
MuZero Breakout	1	0.0	1.107	0.252	10.6
	2	96.4	0.036	0.161	12.9
	4	**100.0**	0.000	0.154	15.1
	8	**100.0**	0.000	0.154	15.1
	All	**100.0**	0.000	0.154	15.1
SEZ Pong	1	0.0	1.167	0.272	11.6
	2	90.9	0.091	0.198	13.8
	4	**100.0**	0.000	0.194	19.6
	8	**100.0**	0.000	0.193	19.7
	All	**100.0**	0.000	0.194	19.8

avoids $\{1, 2\}$. A query-directed audit checks only size-$k^\star$ subsets of $\mathcal{E}_x \setminus D$: the first passing set is a witness, while exhaustion proves absence at that size. It needs at most $\binom{H-|D|}{k^\star}$ additional membership tests from a fresh cache, given the minimum-size proof. An unfinished query returns unknown. This answers a requested structural question without always completing the entire family. The comparison below uses one, two, and four representative certificates.

Table 55: Unavailable-pair queries on decisions with multiple minima. Feasible queries admit an equally small certificate. Misses count feasible queries with no prefix witness (unknown, not proved absent). Core errors count infeasible pairs that avoid the Core. Q compares query-directed versus complete-layer membership tests given a proven minimum size, from a fresh cache; counts exclude the shared lower-layer proof.

Setting	Feas./all	Miss 1 ↓	Miss 2 ↓	Miss 4 ↓	Core err. ↓	Q: dir./all ↓
MuZero Pong	254/670	83	14	**0**	53	15.6/88.9
MuZero Breakout	388/610	146	13	**0**	65	2.5/9.4
SEZ Pong	686/961	236	33	**2**	77	2.9/13.7

The unavailable-evidence study uses the 152 multi-minimum inputs in the core tree-search archive (30 Pong, 56 Breakout, 66 SEZ Pong). Their evidence counts range from 3 to 12. We merge native query caches and physically complete every missing mask, then check that each original minimum size and complete minimum family is unchanged. For every single- and double-unit unavailable set U, we compute the constrained optimum over all valid sets avoiding U. There are 3,112 scenarios: 2,045 retain an original-size certificate, 106 require a larger one, and 961 have no valid set under the fixed intervention and predicate. Among the 106 enlarged cases, 92 need one additional unit, 12 need two, and two need three.

Representations and queries. Prefix-K stores the first up to K minima under lexicographic enumeration, for $K = 1, 2, 4, 8$. Full stores the complete minimum family. Diverse4 greedily maximizes distance within that known family and is an oracle representation reference. Found4 retains up to four diverse witnesses at the smallest size actually observed in the original FutureCert query trace (seed 61831); that size need not be globally minimum. Directed query controls start with Prefix4 and enumerate sets avoiding U, starting at the independently established original minimum size, under additional caps 16,96,384 or exhaustive completion. A returned set is verified; exhausting a finite cap yields unknown. Completing every eligible mask proves infeasibility under the fixed verifier.

Table 56: **Certificate substitution and constrained expansion.** Counts over 3,112 scenarios from 152 decisions. The columns count returned witnesses, not independent trials. Directed queries start from the first four enumerated minima and permit 96 additional queries per scenario.

Available certificates / query rule	Same-size found ↑	Larger found ↑
First enumerated minimum	1,402	0
Up to four enumerated minima	2,043	0
Complete minimum family (reference)	2,045	0
FutureCert queried portfolio (up to four)	1,400	10
Four minima + constrained queries	**2,045**	**88**

The family-size distribution is 119 inputs with two minima, 22 with three, ten with four, and one with eleven. Prefix4 is therefore already the complete family on 151/152 inputs. It recovers 2,043/2,045 same-size opportunities, versus 1,402 for one minimum; Found4 recovers 1,400 same-size and ten larger witnesses. Prefix4 acquisition averages 49.30 queries per input under the specified enumeration order, compared with 49.99 for the full minimum layer; these are offline acquisition counts, separate from per-request queries. Directed96 recovers all 2,045 same-size opportunities and 88/106 larger witnesses, leaving 86 of all 3,112 scenarios unknown. The remaining cases are proved infeasible within that cap.

Clustered summaries. The original summaries first average unavailable sets within an input, inputs within an episode, and episodes within each task; tasks receive equal weight, with singles and pairs reported separately. Episode-block percentile bootstrap intervals use 4,000 draws. A separate input-block analysis resamples 152 inputs together with all their associated single/double scenarios and recomputes pooled ratios. It gives same-size feasibility 65.71% [60.57,70.83], enlarged feasibility 3.41% [2.23,4.72], and infeasibility 30.88% [25.54,36.29]. Conditional same-size recall is 68.56% [66.60,70.47] for Prefix1 and 99.90% [99.70,100] for Prefix4; Directed96 enlarged-case recall is 83.02% [64.66,98.94]. The pooled input-block and balanced episode-block summaries have different weighting; neither treats the 3,112 scenarios as independent inputs. This is evidence-availability auditing, with no measured downstream replan or return benefit.

The analysis replays the completed exact-family records for all 1,094 original tree-search decisions. Every unavailable singleton and unordered pair is queried. Main-table results restrict to the multi-minimum stratum (30/56/66 decisions); the archive also reports all decisions. A prefix without a witness returns unknown. Core alone gives exact singleton availability, but pair availability depends on combinations of alternatives. Query-directed cost is the number of membership tests on the allowed minimum layer from a fresh cache, conditional on an already proven $k^\star$; it excludes the shared minimum-size proof and does not represent a new physical runtime benchmark.

G.3 Repair without the Exact Family

The counts above use exhaustive truth for evaluation; the directed-query controls receive a proven minimum size and a precomputed prefix of up to four minimum certificates. In Table 57, neither policy receives $k^\star$, the exact family, or the Core, and queries used to build a cache count against the cap. Constrained search from the empty set finds a verified witness in 96-100% of feasible scenarios within 384 queries and proves 919 of the 961 infeasible scenarios; within 96 queries it recovers 76.2% on MuZero Pong. Starting from the certificates FutureCert already queried does not raise recovery (it is 1.5 points lower on MuZero Pong within 96 queries), slightly lowers the share of constrained-optimal witnesses, and uses 30-60% more queries. In these settings, the certificate structure is an audit reference for repair rather than a source of query savings.

TD-MPC2. Tree-search decisions have few units, so enumeration from the empty set is cheap there. Table 58 repeats the audit on all 600 TD-MPC2 inputs and every single- and pair-unavailability scenario (46,800), where the 4,096-set space no longer fits small budgets. Restricting the default FutureCert solver to available units recovers 98.0%, 95.9%, and 99.9% of feasible scenarios within 384 queries, against 85.3%, 77.4%, and 99.8% for enumeration, with about half the queries; paired differences to enumeration are $+12.7$ $[9.5, 16.6]$ and $+18.5$ $[14.4, 22.8]$ points on

Table 57: **Certificate repair without the exact family.** 152 multi-minimum core decisions, 3,112 single- and pair-unavailability scenarios. Neither policy receives $k^\star$, the family, or the Core. *From empty set* runs constrained enumeration from size zero; *FutureCert queries first* charges the first (up to 96) queries of the default FutureCert run as a cache and then continues the same search. Recovery and constrained-optimal rates use feasible scenarios only (428/658/1,065); infeasible scenarios are reported as proved infeasible; *unknown* means the cap ended the search. Means are episode, then input, then scenario weighted.

Setting	Cap	Policy	Recovery (%)↑	Optimal (%)↑	Infeasible proved	Unknown	Calls↓
MuZero Pong	96	From empty set	76.2	76.2	206/442	347	39.8
	96	FutureCert queries first	74.7	74.5	143/442	416	51.5
	384	From empty set	96.0	96.0	400/442	57	70.3
	384	FutureCert queries first	96.0	95.7	400/442	57	91.6
MuZero Breakout	96	From empty set	100.0	100.0	241/241	0	7.2
	96	FutureCert queries first	100.0	98.9	241/241	0	11.4
	384	From empty set	100.0	100.0	241/241	0	7.2
	384	FutureCert queries first	100.0	98.9	241/241	0	11.4
SEZ Pong	96	From empty set	100.0	100.0	278/278	1	8.4
	96	FutureCert queries first	100.0	99.7	278/278	1	13.8
	384	From empty set	100.0	100.0	278/278	0	8.5
	384	FutureCert queries first	100.0	99.7	278/278	0	13.8

Cheetah Run and Cup Catch. Charging the default run's queries first and then enumerating is dominated by the restricted solver at every cap.

Table 58: **Repair without the exact family on TD-MPC2.** All 600 confirmatory inputs, every single- and pair-unavailability scenario (46,800), simulated from the exhaustive archive. Entries are feasible-scenario recovery (%) / mean queries; no method receives $k^\star$, the family, or the Core. FutureCert is the default solver restricted to available units (ten seeds); *Cache + enumeration* first charges the default FutureCert run's queries and then enumerates. Paired 95% intervals are in the released tables.

Task	Cap	FutureCert	FutureCert (Beam-4)	FutureCert (Enumeration)	Cache + enum.
Cheetah Run	32	57.1 / 25	36.3 / 30	17.8 / 29	44.6 / 26
	96	81.4 / 56	80.0 / 67	45.3 / 71	65.6 / 63
	384	98.0 / 87	95.1 / 80	85.3 / 171	83.4 / 148
Cup Catch	32	44.4 / 28	23.6 / 30	14.4 / 29	27.3 / 29
	96	71.1 / 70	66.8 / 75	32.6 / 78	51.4 / 75
	384	95.9 / 118	91.5 / 95	77.4 / 207	76.7 / 191
Finger Spin	32	97.1 / 4	95.4 / 8	92.7 / 6	95.9 / 4
	96	99.4 / 6	99.0 / 11	97.7 / 9	97.9 / 7
	384	99.9 / 7	99.7 / 12	99.8 / 12	99.7 / 10

Small budgets on tree search. Table 59 shows where the exact family matters: on MuZero Pong within 16-32 queries, a user given the complete minimum family repairs 96-97% of feasible scenarios, whereas every search method stays below 64%; by 384 queries all methods reach 96-100%.

Table 59: **Small-budget repair on MuZero Pong.** Feasible-scenario recovery (%) / mean queries on the multi-minimum Pong decisions. The exact-family reference is given the complete minimum family and is not a method. Breakout and SEZ Pong reach 99-100% for every method from 32 queries on.

Cap	Exact family (ref.)	FutureCert	FutureCert (Beam-4)	FutureCert (Enumeration)	Cache + enum.
16	96.1 / 8	45.7 / 12	45.0 / 12	43.0 / 12	35.5 / 13
32	96.9 / 13	59.5 / 20	63.9 / 20	62.8 / 20	45.0 / 23
96	98.0 / 27	88.2 / 38	95.4 / 36	76.2 / 40	74.7 / 52
384	100.0 / 51	99.8 / 46	98.8 / 37	96.0 / 70	96.0 / 92

Repeated requests. A decision may face several unavailability requests. Keeping verified query results across the requests of one decision reduces the queries per successful repair from 262 to 25 on Cheetah Run, 324 to 29 on Cup Catch, and 256 to 56 on MuZero Pong, and raises feasible recovery to 99-100% (Table 60). Pre-charging the default FutureCert run adds nothing measurable, so the saving comes from sharing verified queries rather than from a precomputed certificate structure.

Table 60: **Repeated repair requests on the same decision (cap 384 per request).** Each decision receives all its single- and pair-unavailability requests in a fixed random order. Entries are queries per successful repair (feasible recovery, %) after the last request. *Independent* starts every request from scratch; *Shared* keeps verified query results across requests of the same decision; *Prewarm* additionally charges the default FutureCert run once at the start.

Setting	Requests	Independent	Shared	Prewarm + shared
Cheetah Run	78	262.0 (85.3)	24.8 (99.2)	24.7 (99.2)
Cup Catch	78	323.9 (77.4)	28.7 (99.0)	28.6 (99.1)
Finger Spin	78	13.3 (99.8)	2.1 (100.0)	2.1 (100.0)
MuZero Pong	78	255.7 (100.0)	56.3 (100.0)	56.3 (100.0)
MuZero Breakout	21	15.2 (100.0)	2.4 (100.0)	2.4 (100.0)
SEZ Pong	55	24.2 (100.0)	4.4 (100.0)	4.5 (100.0)

H Cross-Stream Reliability

This section expands the cross-stream results in Section 5.5.

H.1 Reference and Random-stream Transfer

Sensitivity to references and fidelity conditions We keep an exact-audit partial certificate fixed and test it under 16 reference-episode bootstrap draws. Action retention remains 95.1-97.6%, whereas full-certificate retention is 57.8-67.0%. This evaluates transfer of fixed certificates under changed references independently of the budgeted solver. Relaxing only NME from 0.10 to 0.20 still leaves 27.1-43.7% of the original audited extension edges invalid (Table 32). In a same-state TD-MPC2 control, binding three optimizer stages into each coarse group increases minimum retention from 2.740 to 7.155 equivalent fine units on Cheetah Run and from 3.060 to 9.390 on Cup Catch (Table 28). Appendix A.3 defines the reference construction; Appendix E.1 details the threshold and granularity controls.

The selection and validation populations each use the first four archived decisions in each of five evaluation episodes. The deletion order is ascending full-trace deviation from the reference and is shared across conditions. A single-seed check uses the original planning seed; four-stream selection adds three fixed offsets. Every proposed set is tested under all training seeds against their respective full outputs. The query cap is 96 sets, with seed-specific executions counted separately. The final sets are written before accessing 32 held-out seed outputs. Mean-reference results are in Table 63. Paired 95% episode-decision bootstrap intervals for the four-minus-one-stream pass-rate gains are [30.15,60.94], [27.50,60.00], and [4.38,28.12] percentage points, respectively.

The original reference mean is supplemented with mean predictions from the first three development episodes, averaged over their ten full-trace sampled decisions. Selection checks the Cartesian product of these four reference banks and the four training seeds. Validation alternates the means from the two remaining development episodes over the same 32 held-out streams; neither reference episode participates in multi-context selection. Full outputs are reference-independent and are shared across these comparisons. Baseline one- and four-stream sets are frozen outputs from the preceding experiment. This control changes reference construction as well as reference episodes; it tests these mean-replacement banks, not a learned on-manifold counterfactual generator. For this TD-MPC2 adapter, the reference terms are shared additive constants and cancel in real arithmetic (Proposition 7). The fixed-mask audit in Section I.2 attributes observed reference-dependent outputs to finite-precision amplification; these rows do not identify a semantic benefit of changing the constants.

Table 61: Retention of fixed partial certificates under reference-episode bootstrap (%, ↑ for all pass rates). Each certificate is tested with 16 reference draws. Each column measures a separate fidelity condition.

Setting	Action ↑	Ranking ↑	Margin ↑	Full ↑
MuZero Pong	96.7	84.0	71.9	64.4
MuZero Breakout	97.6	78.8	69.2	57.8
SEZ Pong	95.1	88.4	72.9	67.0

Table 62: **Overall reliability can include full-set fallback.** Four-stream selection is tested on 32 held-out streams for 20 inputs per task. Size is retained evidence (↓); compact columns condition on non-full outputs.

Setting	Overall ↑	Full outputs ↓	Size ↓	Non-full pass ↑	Non-full: all 32 ↑
Cheetah Run	81.9	11/20	10.90	59.7	0/9
Cup Catch	83.3	11/20	11.20	62.8	0/9
Finger Spin	84.2	0/20	2.35	84.2	4/20

Table 63: Costs and conditional coverage for the seed-robustness experiment. Partial pass restricts to returned non-full sets; their populations differ across rows. All 32 is the percentage of selected sets passing every held-out seed. Selection calls count every seed-specific non-full execution; full targets and held-out validation are separate.

Setting	Seeds	Pass ↑	Partial pass ↑	All 32 ↑	Selection calls ↓
Cheetah Run	1	35.0	27.8	10.0	20.5
	4	81.9	59.7	55.0	61.0
Cup Catch	1	37.8	30.9	10.0	17.0
	4	83.3	62.8	55.0	54.8
Finger Spin	1	68.1	68.1	15.0	8.6
	4	84.2	84.2	20.0	54.8

Table 64: FP32 joint random-stream and shared-reference sensitivity on the same 20 decisions per task. Every row is tested on 32 new seeds and two held-out reference-episode means. Pass averages all outputs, including full sets; Partial restricts to non-full sets. Size counts retained units; Cover. is non-full coverage (%). Calls count all seed-reference-specific selection executions.

Setting	Selection	Pass ↑	Partial ↑	Size ↓	Cover. ↑	Calls ↓
Cheetah Run	One seed	34.5	27.3	5.65	90	20.5
	Four seeds	81.4	58.7	10.90	45	61.0
	Four seeds + four refs.	**84.7**	56.2	11.20	35	228.8
Cup Catch	One seed	40.0	33.3	8.35	90	17.0
	Four seeds	83.6	63.5	11.20	45	54.8
	Four seeds + four refs.	**91.9**	67.5	11.70	25	209.6

The additional joint check changes all-output validity by +3.28 and +8.28 points relative to four-stream selection, with paired intervals $[-1.25, 10.32]$ and $[-0.47, 19.53]$. Full-set returns increase in Cheetah Run and Cup Catch, so Table 64 also reports coverage and validity among non-full sets. Held-out empirical validity is not an exact probability certificate, and the five-episode sample limits precision.

The first reliability experiment uses only the two frozen one/four-stream proposals. Distinct non-full masks define the multiplicity correction; the full set is excluded. Calibration and test seeds are independently sampled integers from $[1, 2^{31} - 1)$ and written before accessing responses. The selected mask is saved before test evaluation. Empty candidate pools or failed bounds lead to abstention, not a full-set success. Table 66 reports the completed first experiment.

The second experiment freezes the union of those proposals and all twelve single-unit deletions. Its calibration and test streams are new, and its primary threshold remains $\tau = 0.8$, $\alpha = 0.05$. The

Table 65: Compact-only certification with a broader frozen candidate pool. Base uses the original two proposals; Broad adds every one-unit deletion before sampling fresh streams. Both use 64 calibration and 64 test streams, an 80% target, and 95% per-input simultaneous confidence. Accepted/20 excludes full sets and abstentions. Size and test validity condition on acceptance; the last column requires at least 25% evidence removal.

Setting	Base ↑	Broad ↑	Size ↓	Test (%) ↑	$\lvert S\rvert \leq 9$ ↑
Cheetah Run	0/20	**1/20**	11.00	90.6	0/20
Cup Catch	0/20	0/20	-	-	0/20
Finger Spin	10/20	**18/20**	6.22	97.1	9/20

Table 66: Compact-only reliability certificates: two frozen proposal sources, 64 new calibration streams, and 64 independent test streams. Target pass probability is 80% with 95% confidence per input, simultaneous over its candidates. Coverage excludes full sets and abstentions. Size, bound and test pass condition on acceptance; - denotes no accepted output. Cal. counts non-full calibration executions, additional to proposal search and shared full targets.

Setting	Accepted/inputs ↑	Size ↓	Bound (%) ↑	Test (%) ↑	Cal. ↓
Cheetah Run	0/20	-	-	-	86.4
Cup Catch	0/20	-	-	-	86.4

restricted-pool baseline is recalibrated on this same new stream sample with its own multiplicity correction. Single-deletion candidates are stopped only when their largest possible final bound is below 0.8; the original candidates receive all 64 draws so both pool definitions can be compared. Acceptance is identical to completing all fixed-sample trials. The procedure reduces query cost by early rejection; it never accepts early. Test outcomes do not alter either candidate pool or threshold. Mean calibration costs in Table 67 exclude source proposal search and list shared target runs separately.

Table 67: Broader-pool calibration cost. Early rejection stops only when passing all remaining draws cannot reach the fixed confidence gate; accepted masks always receive all 64 draws. Thus it preserves the accept/reject result of full fixed-sample evaluation. Bounds condition on accepted outputs. Shared full targets and proposal search are accounted for separately.

Setting	Mean bound (%) ↑	Early-reject calls ↓	All-64 calls ↓	Full targets ↓
Cheetah Run	81.0	**208.7**	841.6	58.6
Cup Catch	-	**192.7**	841.6	60.2
Finger Spin	87.4	**757.5**	870.4	64.0

H.2 Exhaustive Stream Analysis and Calibration

Table 68: **Small fixed-stream certificates can require enlargement across streams** ($H = 12$). Pass and LCB refer to the k_{16} optimum; k_{cal} is the smallest certified of 26 frozen candidates (LCB ≥ 0.8).

	Cheetah-43	Cheetah-83	Cheetah-122	Cup-121	Cup-2	Cup-80
Single-stream median ↓	1.0	4.0	2.0	2.0	4.0	1.5
k_{16} ↓	1	12	12	10	11	10
Pass/200 ↑	164	200	200	116	154	132
LCB ↑	0.730	0.969	0.969	0.475	0.674	0.557
k_{cal} ↓	12	12	12	12	12	12

The exhaustive stream study fixes three inputs each on Cheetah Run and Cup Catch using earlier reliability results: Cheetah-43/83/122 and Cup-121/2/80. They are diagnostic cases selected to span previous outcomes. Each input receives two disjoint discovery groups A and B of eight random streams, plus 200 independent calibration streams. All $2^{12} = 4096$ masks are physically re-executed under every discovery stream, for 393,216 mask-stream evaluations. The full target and intervened

output share that stream, original input, reference vector, and planning state. Action and full (action plus optimized mean) tolerances are both 0.20. References are explicitly FP32, matching the original enumeration; eight original-seed masks per input are checked against that archive before the new runs.

For predicate V, we compute

$$k^{\star}_{\mathrm{path}}(\xi) = \min_{S: V(S,\xi)=1} |S|, \quad k_{16} = \min_{S:\, 16^{-1}\sum_{\xi} V(S,\xi) \geq 0.8} |S|, \quad \Delta_{\mathrm{stab}} = k_{16} - \mathrm{median}_{\xi}\, k^{\star}_{\mathrm{path}}(\xi).$$

The 0.8 threshold requires 13 successes, not unanimity. A representative is the smallest integer mask among all minima on its stream; transfer averages its validity on the other 15 streams. Unit-frequency heatmaps average over the complete tied minimum family, avoiding arbitrary representative tie-breaking. A/B cross-selection chooses a maximum-pass mask separately at each size in one eight-stream group and evaluates it on the other, then reverses the roles.

Table 69: **Execution-conditioned compactness and cross-stream certification.** Six diagnostic inputs, with action/full entries. k_{16} is the exact minimum passing at least 13 of 16 discovery streams; $\Delta = k_{16} - \mathrm{median}\, k^{\star}_{\mathrm{path}}$. Transfer tests a deterministic single-stream minimum on the other 15 streams. k_{cal} is the smallest pre-frozen candidate certified on 200 fresh streams with simultaneous 95% lower bound at least 0.8.

Input	Path size ↓	k_{16} ↓	Δ ↓	Transfer (%) ↑	k_{cal} ↓
Cheetah-43	1/1	1/1	+0/+0	47.9/52.9	12/12
Cheetah-83	3/4	12/12	+9/+8	8.8/14.6	12/12
Cheetah-122	2/2	12/12	+10/+10	10.4/22.5	12/12
Cup-121	1.5/2	10/10	+8.5/+8	8.8/2.9	12/12
Cup-2	3/4	11/11	+8/+7	2.9/1.2	12/12
Cup-80	1/1.5	10/10	+9/+8.5	12.5/4.6	12/12

Calibration. Before inspecting any calibration outcomes, we select one maximum-discovery-pass mask at each exact size 0-12 for each predicate, breaking ties by integer mask. For each of these 26 candidates and each input, let m_j be its successes on $n = 200$ fresh streams. The simultaneous one-sided lower bound is $\mathrm{Beta}^{-1}(0.05/26; m_j, n-m_j+1)$ for $m_j > 0$, and zero otherwise. A candidate is certified if this bound is at least 0.8. The guarantee covers both predicates and all frozen candidates for one input. Full retention passes 200/200 and has lower bound 0.969; it remains identifiable as the trivial complete-evidence outcome. Table 70 reports both the discovery-optimum candidate and the smallest independently certified frozen candidate. None of the fixed non-full candidates is certified on the six cases, whereas full retention is certified on all of them. The 156 candidate records are included in the result archive.

Search versus existence. A separate offline diagnostic exposes the same 16-stream aggregate predicate to $R = 1/4/16$ (96 new masks per restart) and unchanged FutureCert (Beam-4) (384 masks). Each aggregate query evaluates one set on all 16 discovery streams. Candidate priors and native ordering remain those frozen from the original development protocol. Proposal scoring uses the 13th ordered per-stream error, with joint action/mean error for the full predicate. FutureCert reaches the empirical optimum in five of six full-predicate cases, with mean Gap 1 on Cup-2 across five fixed solver seeds. The exact oracle, rather than solver success, establishes the empirical stability cost.

The two-mode study follows up Cheetah-83/122 and Cup-121/2/80, the five cases from the exhaustive stream study with compact individual-stream minima and shared optima of 10-12. For each input, 64 new full-retention streams fit a two-cluster router on standardized action and optimized mean (KMeans with ten initializations and fixed seed 9242010). The router assigns each discovery stream from that study to a mode. For each predicate and size 0-12, each mode receives a maximum-pass candidate selected only from its discovery streams, with integer-mask tie-breaking; a mode with fewer than four discovery streams receives the full set. Global candidates use all 16 discovery streams. The complete policy is frozen before a further 200 independent streams are observed. Routing uses the full planner output and therefore describes a post-hoc explanation, not a planning speedup.

Table 70: **Independent calibration under both predicates.** The reported sizes, pass counts, and bounds coincide for the action and full predicates on these six inputs. S_{16} is the frozen candidate at the empirical optimum size. The right columns give the smallest certified candidate among all 13 sizes; multiplicity covers both predicates (26 candidates). LCB is the one-sided simultaneous lower bound.

	Empirical-optimum candidate S_{16}				Smallest certified candidate		
Input	Size	Pass/200	LCB ↑	Certified	Size	Pass/200	LCB ↑
Cheetah-43	1	164/200	0.730	No	12	200/200	0.969
Cheetah-83	12	200/200	0.969	Yes	12	200/200	0.969
Cheetah-122	12	200/200	0.969	Yes	12	200/200	0.969
Cup-121	10	116/200	0.475	No	12	200/200	0.969
Cup-2	11	154/200	0.674	No	12	200/200	0.969
Cup-80	10	132/200	0.557	No	12	200/200	0.969

Table 71: **Two-mode routing does not yield a certified non-full policy in these five cases.** Full predicate; the strongest non-full routed LCB is shown among the simultaneously calibrated frozen policies. Both global and routed strategies certify only full retention. Action-only calibration gives the same certified sizes.

Input	Mode counts (discovery)	Best non-full pass	LCB ↑	Certified size: global/routed
Cheetah-83	13/3	140/200	0.591	12 / 12
Cheetah-122	8/8	107/200	0.423	12 / 12
Cup-121	10/6	156/200	0.678	12 / 12
Cup-2	8/8	157/200	0.684	12 / 12
Cup-80	7/9	156/200	0.678	12 / 12

The two policy types, 13 sizes, and two predicates give 52 frozen policies per input. We simultaneously calibrate their unconditional routed/global success probabilities with one-sided CP bounds at $\alpha = 0.05/52$. Mode-specific sample counts and pass rates are descriptive. This is a prespecified two-mode diagnostic, with no selection among alternative routers using calibration outcomes.

The two-mode grouping does not convert the observed discovery-stream heterogeneity into independently certified compact portfolios. These results restrict this fixed router and candidate construction; the stability costs measured by the exhaustive stream study remain independent of that grouping.

H.3 Calibration on 50 Inputs per Task

Table 72 extends the cohort calibration from 20 to 50 inputs per task. The original 20 inputs keep their archived positions and seeds; on Cheetah Run and Cup Catch, re-executing them reproduces every discovery proposal, held-out rate, calibration indicator, bound, and decision (1/20 and 0/20), and Finger Spin certifies 18/20. The 30 added inputs per task were drawn six per evaluation episode from decisions after the original first four, frozen before any outcome, and run with fresh seeds disjoint from every earlier seed. They give 0/30, 0/30, and 30/30, and the pooled cohorts give 1/50, 0/50, and 48/50. Because the added inputs come from later decisions, the pooled counts are not a simple random sample of each task's decisions; the guarantee is per input and simultaneous over its frozen pool, and the counts over inputs are descriptive.

H.4 Paired Constant-free Calibration

Table 73 asks whether the cross-stream result is an artifact of FP32 arithmetic. The original non-full discovery proposals of the first 20 inputs per task are re-calibrated on 100 new streams under FP32 and the constant-free implementation, with the bound corrected jointly over both implementations and predicates. The two implementations select the same set, or abstain, on all 40 inputs; compact certificates remain absent for Cheetah Run and Cup Catch. The pool excludes the one-unit deletions of the main-text protocol and the correction is stricter, so the counts are not a replacement for the main-text cohort result. On Finger Spin, the full 50-input cohort under the main-text protocol cer-

Table 72: **Stream-reliability calibration on 50 inputs per task.** The 20 original inputs keep their archived positions and seeds; 30 further inputs per task (six per evaluation episode, later decisions than the original first four) were drawn and frozen before any outcome, with fresh seeds. Protocol as in the main text: proposals from one and four discovery streams plus all one-unit deletions, 64 calibration and 64 test streams, per-input Bonferroni correction, $\tau = 0.8$, $\alpha = 0.05$. CI is the exact 95% interval over inputs. *Two-proposal* counts certification from the two discovery proposals alone. Test pass is the held-out pass rate of certified sets; the last two columns give 32-stream held-out pass rates of four-stream selections, all outputs and compact outputs.

Task	Cohort	Certified	95% CI	Two-proposal	Test pass	4→32 all	4→32 compact
Cheetah Run	Original 20	1/20	[0.00, 0.25]	0/20	90.6	81.9	59.7
	Added 30	0/30	[0.00, 0.12]	0/30	-	93.6	68.2
	All 50	1/50	[0.00, 0.11]	0/50	90.6	88.9	63.1
Cup Catch	Original 20	0/20	[0.00, 0.17]	0/20	-	83.3	62.8
	Added 30	0/30	[0.00, 0.12]	1/30	-	86.7	66.7
	All 50	0/50	[0.00, 0.07]	1/50	-	85.3	65.0
Finger Spin	Original 20	18/20	[0.68, 0.99]	10/20	97.1	84.2	84.2
	Added 30	30/30	[0.88, 1.00]	12/30	96.3	82.3	82.3
	All 50	48/50	[0.86, 1.00]	22/50	96.6	83.1	83.1

tifies compact sets for 48/50 inputs under both FP32 and the constant-free implementation, on the same inputs and streams.

Table 73: **Paired FP32/constant-free calibration of the original frozen candidates.** For each of the 20 cohort inputs per task, the originally frozen non-full candidates are calibrated on 100 new streams under both implementations, with the one-sided bound corrected jointly over two implementations, two predicates, and all candidates ($4m$). Entries count inputs with a certified compact set. The two implementations select identical sets or abstain identically on all 40 inputs under both predicates. The candidate pool here holds only the original non-full proposals (the main-text protocol adds all twelve one-unit deletions), and the correction and stream counts differ, so these counts are not comparable with the main-text cohort result.

Task	FP32 action	FP32 full	CF action	CF full
Cheetah Run	0/20	0/20	0/20	0/20
Cup Catch	0/20	0/20	0/20	0/20

I Reference and Implementation Stability

For Section 5.6, reference robustness of fixed certificates and minimum families is reported with the fidelity conditions in Appendix E; the constant-free re-enumeration is in Appendix D. This appendix gives the shared-reference invariance result and the FP32 drift analysis.

I.1 Certificate Stability across Audit Conditions

Table 74 extends the constant-free comparison to 34 Cheetah Run and 33 Cup Catch inputs and separates three factors. The minimum size is stable under changed arithmetic and reference banks for Cheetah Run (94-95%) but not for Cup Catch (50.8% and 56.1%). A new planning stream changes the minimum size for 56-69% of inputs, and it does so equally under FP32 and constant-free arithmetic, so stream dependence is not a floating-point artifact. The full-predicate arrays of all physical cells were recomputed and every first valid layer and all smaller layers were verified complete.

I.2 Shared-reference Invariance and Numerical Drift

For the TD-MPC2 adapter, every replaced quantity is shared across candidates and has a candidate-independent discount. At a fixed iteration and candidate pool,

$$G_i(S, \bar{z}) = \sum_{j \in S} \alpha_j z_{ij} + c(S, \bar{z}), \qquad c(S, \bar{z}) = \sum_{j \notin S} \alpha_j \bar{z}_j. \tag{14}$$

Table 74: **Certificate structure across arithmetic, reference banks, and planning streams (full predicate).** 67 TD-MPC2 inputs (34 Cheetah Run, 33 Cup Catch), three development reference banks, FP32 and constant-free (CF) implementations, and four planning streams; each cell completes the first valid cardinality layer, giving exact $k^\star$, family, and Core. Pairs change one factor at a time; metrics are averaged within input, then over inputs. Empty/empty Jaccard is 1 (last column: pairs with both Cores empty). Family Jaccard is 0 whenever $k^\star$ differs, so the equal-$k^\star$ column isolates composition changes. Under CF, reference banks give identical results by the implementation's reference invariance, verified by per-cell gates, so that axis is not an independent measurement and is omitted.

Factor	Task	Pairs	$k^\star$ kept (%)	$\lvert\Delta k^\star\rvert$	Core J.	Fam. J.	Fam. J. ($= k^\star$)	Both $\varnothing$ (%)
Arithmetic	Cheetah Run	408	94.9	0.06	0.900	0.859	0.905	24.0
	Cup Catch	396	50.8	0.56	0.365	0.179	0.354	10.4
Reference (FP32)	Cheetah Run	408	94.1	0.06	0.888	0.852	0.905	23.3
	Cup Catch	396	56.1	0.50	0.383	0.197	0.352	11.1
Stream (FP32)	Cheetah Run	612	41.3	0.76	0.391	0.058	0.139	11.6
	Cup Catch	594	33.3	0.96	0.213	0.025	0.075	7.2
Stream (CF)	Cheetah Run	612	43.6	0.74	0.406	0.059	0.135	13.2
	Cup Catch	594	31.3	1.00	0.177	0.014	0.045	5.1

Proposition 7 (shared-reference invariance). With exact real arithmetic, finite returns, identical random draws, candidate-independent discounts and termination, and translation-invariant tie breaking, changing these shared constants leaves the entire MPPI output unchanged for a fixed mask. The common shift preserves elite indices and the centered exponential weights; induction then preserves every subsequent candidate pool. This property need not hold for candidate-specific references or termination.

Our archived interpreter accumulates returns in FP32. The optional constant-free interpreter omits shared replacement terms before summation on partial masks, while retaining the model calls and random draws. At full retention it executes the original arithmetic verbatim. It implements the same real-arithmetic intervention and makes fixed-mask outputs independent of reference constants by construction; partial outputs are reverified under this interpreter rather than read from the FP32 archive. Section I.2 tests both implementations and isolates rounding amplification.

Table 75: Fixed-mask reference changes on ten inputs per task. Counts are over two alternative episode references and distinct nonempty partial masks. Diff. means any action/mean bit difference; Act. exceeds the 0.20 action tolerance; Flip changes declared validity. CF is the constant-free interpreter. All random-number digests and full endpoints agree.

Setting	Pairs	FP32 diff. ↓	Act. ↓	Flip ↓	CF diff. ↓
Cheetah Run	106	101	10	2	**0**
Cup Catch	102	98	52	9	**0**

For Proposition 7, fix the planner input, initial optimizer state, mask, and random stream. Before the first update, candidate action sequences and latent rollouts coincide. By Equation 14, every candidate return changes by the same scalar when the reference changes. Elite order is unchanged under translation-invariant tie breaking, and the scalar cancels from $G_i - \max_{e\in\text{elite}} G_e$. Consequently both weighted means and variances agree. The next pool, constructed from these moments and the same draws, also agrees. Induction establishes agreement across all six iterations and the final weighted random selection. The proof assumes finite unclipped real returns; FP32 rounding and overflow handling are outside it. The optional implementation cancels only constants that meet these conditions. Its full-mask branch calls the original estimator, so native full-output equivalence remains an executable assertion.

The reference audit uses the first two of the four previously sampled seed-study decisions in each evaluation episode. For each input we deduplicate full/empty masks, the one/four-stream submissions, and four deterministic random masks. Every mask is replayed under the original reference,

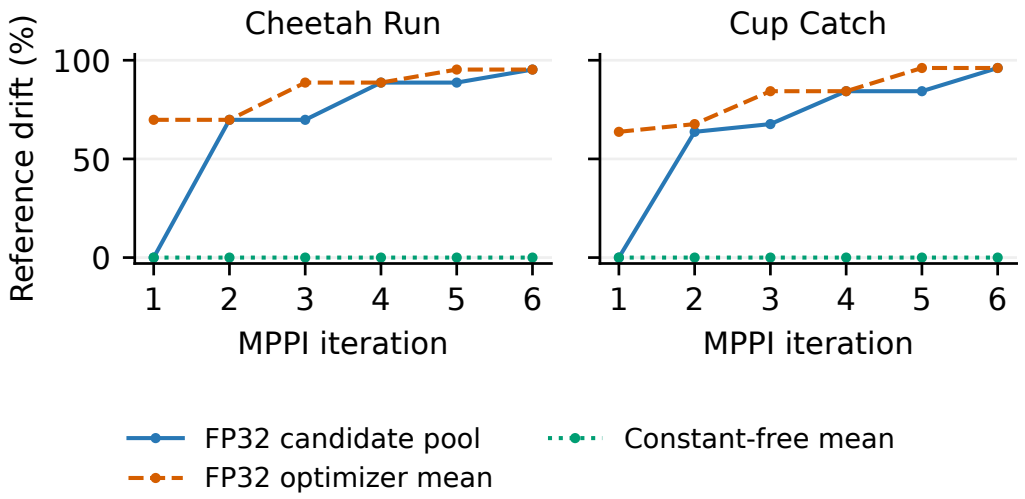


Figure 11: **Reference-dependent numerical drift across six optimizer iterations.** Percentages use the paired nonempty partial-mask comparisons in Table 75. The panels show Cheetah Run and Cup Catch. FP32 begins from identical candidates, but altered optimizer moments change later pools; the constant-free interpreter has no reference-dependent mean differences.

two held-out development-episode means, and zero. The main paired table restricts to nonempty partial masks and the two episode means. Instrumented outputs match the uninstrumented FP32 interpreter; a separate 120-query replay matches the original enumeration archive bit for bit. The standalone constant-free implementation matches the instrumented implementation in 48 checks over three tasks, two inputs, four masks, and two references. Per-round candidate, mean, score, and RNG records accompany the results. Same-pool FP64 reconstruction changes accumulation precision, not model tensors or candidate generation.

J Qualitative Examples

Figure 12 shows one real decision for each of four phenomena in the main text: a frozen search tree accepting a set that native search rejects, a same-size substitute that only the complete minimum family reveals, unavailable evidence that forces a larger certificate, and a set that is minimum on individual streams but not certified across streams. Every case is selected by a rule fixed before any candidate was inspected, and every value is read from the archived audits.

(a) MuZero Pong, decision 1 ($H = 4$; $k^\star = 4$, so the exact minimum is the full set). KernelSHAP, LIME, and RISE all commit $S = \{2, 3\}$ through the frozen tree (F1), whose replayed root visits (7, 9, 16, 5, 9, 4) keep the top action with Spearman $\rho = 0.971$ and normalized margin error (NME) 0.071 against the original (6, 10, 17, 3, 11, 3). Re-executed natively, the search expands 9 nodes absent from the recorded tree and returns (6, 9, 14, 7, 9, 5): the action is kept and NME is 0.071, but $\rho = 0.882 < 0.95$, so the set is rejected. (b) MuZero Pong, decision 81 ($H = 6$, $k^\star = 3$): $\mathcal{F}^\star = \{\{1, 4, 6\}, \{3, 4, 5\}\}$ and Core $= \{4\}$. With unit 1 unavailable, a user holding only the first minimum has no witness, whereas the complete family supplies the same-size substitute $\{3, 4, 5\}$. (c) The same decision with the Core unit 4 unavailable: both minima use it, all 26 sets of size at most 3 that avoid it fail, and the smallest valid set is $\hat{S} = \{1, 3, 5, 6\}$ of size 4, the only valid one among the 5 size-4 sets that avoid unit 4. (d) TD-MPC2 Cheetah-43 (Table 68), full predicate. Minimum certificates on the 16 individual discovery streams have size 0 on 6, 1 on 8, and 2 on 2 streams (median 1). The 16-stream optimum $S_{16} = \{10\}$ (terminal value, iterations 1-2) passes 13/16 discovery streams but only 164/200 fresh streams, a simultaneous lower bound (LCB) of 0.730$<$ 0.8. No non-full frozen candidate is certified (highest LCB 0.794, size 11, 175/200), so the smallest certified candidate is full retention (200/200, LCB 0.969). TD-MPC2 grids: quantities r_0, r_1, r_2, V (rows) by MPPI iterations (columns); tree-search units are search depths; ✓/✗ mark passing/failing the fidelity predicate in (a)-(c) and certification (LCB $\geq$ 0.8) in (d). Each case is the first eligible one under a rule fixed before any candidate was inspected, with no case skipped or replaced: scanning MuZero Pong, MuZero Breakout, then SEZ Pong by state index, (a) is the first F1-committed nonempty, non-full set that keeps the native action but fails native fidelity, (b) the first decision with at least two

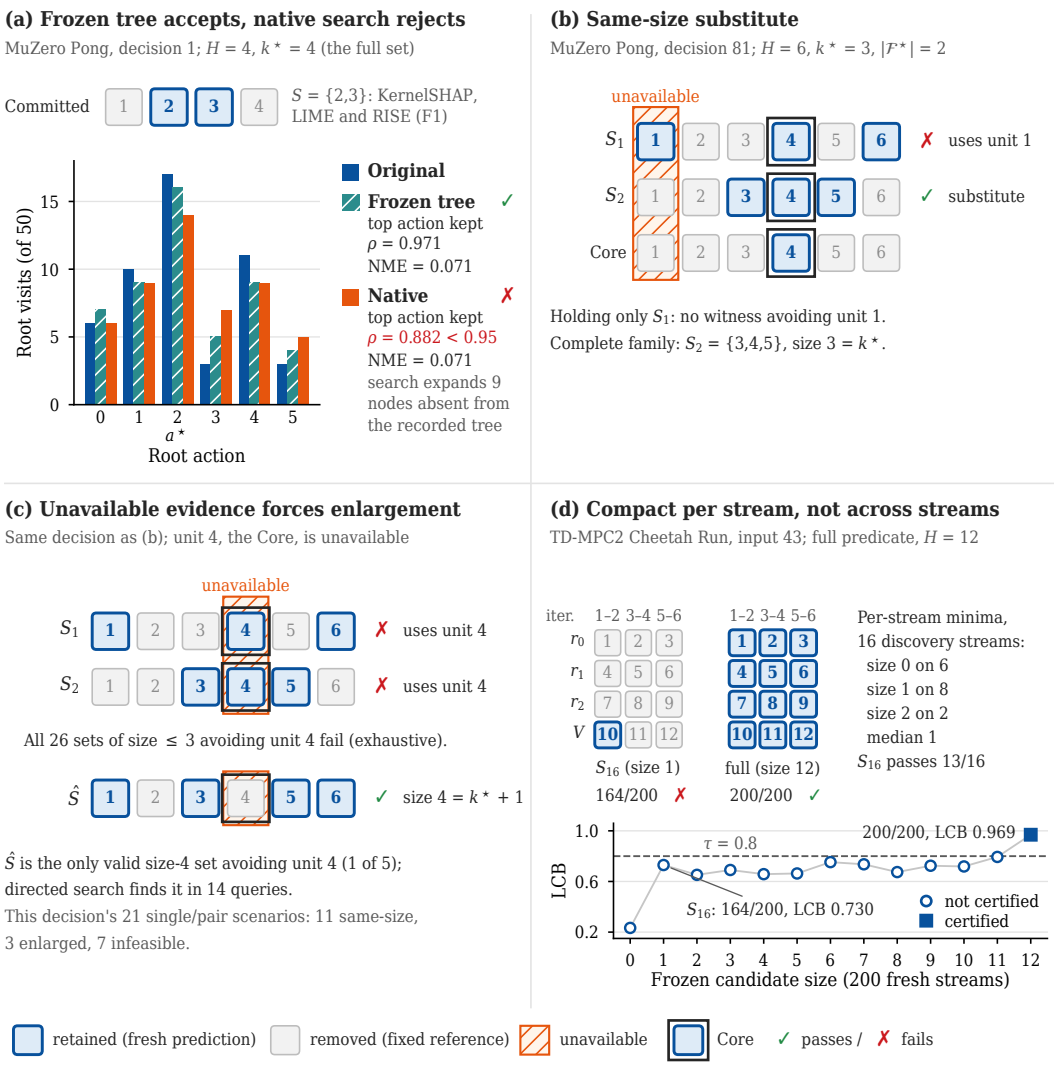


Figure 12: **On real decisions, whether an evidence set suffices changes under native re-execution, under unavailable evidence, and across planning streams.**

minima, with the smallest non-Core unit of its first minimum made unavailable, and (c) the first single- or pair-unavailability scenario whose constrained minimum exceeds $k^\star$; (d) is the first input in Table 68 order whose non-full 16-stream optimum is not certified while a larger frozen candidate is.

K Theory and Proofs

The following analyses separate intervention semantics, fidelity conditions, certificate structure, and iteration. Propositions 1 and 2 characterize information and randomness boundaries. The remaining propositions concern conditional reliability, intersection representation, proof records, arithmetic semantics, and exact reuse. Table 76 links each statement to its assumptions, proof, and experimental test. FutureCert's incumbent property is stated separately from these propositions. For Proposition 6, the instance-specific additional record has length $Q^*_{\text{record}} = \binom{H}{k^\star} - \binom{H-c}{k^\star-c} + \rho_{S_0} - 1$. The exhaustive stream study's k_{16} and the unavailable-evidence study's $k_{\neg U}$ are constrained-problem definitions (Appendices H.2 and G.2), rather than additional propositions.

Table 76: **Theory-experiment map.** $k^\star$ is independently proved minimum size, $c = |\mathcal{K}|$, and ρ_{S_0} is the smallest witness subfamily containing a known minimum S_0. Calibration confidence is conditional on one planning input and its frozen candidates.

Statement	Premise / question	Conclusion	Proof	Evidence	Scope
Prop. 7: shared references	Exact real arithmetic; shared additive constants; identical draws	Complete MPPI output invariant to these shifts	App. 7	I.2	Excludes FP32 rounding and candidate-dependent terms
Prop. 3: calibration	Frozen candidates, IID streams from $\mathcal{Q}$	$\Pr(\text{any false acceptance}) \leq \alpha$	App. K.2.1	H.2	Per-input reliability; neither existence nor compact coverage
Prop. 5: witnesses	Proved $k^\star$; known minimum S_0	$\lvert\mathcal{W}\rvert \leq k^\star - c + 1, \bigcap \mathcal{W} = \mathcal{K}$	App. G.1	G.1	Representation size, not discovery cost
Prop. 6: proof record	Membership oracle; no other known size-$k^\star$ labels	Q^*_{record} in Eq. 12; directed $K_{\text{confirmed}} \subseteq \mathcal{K} \subseteq K_{\text{possible}}$	App. G.1	G.2	Instance record length; minimum-size proof cost separate
Prop. 4: exact reuse	Bitwise determining inputs; deterministic operators; native order; RNG restoration	Intervened output bitwise equal to native re-execution	App. B.6	B.4	Equivalence, not acceleration
Prop. 8: prefix reuse	Bitwise determining inputs; deterministic transitions; RNG restoration	Exact output and $L_S - p$ fresh model evaluations	App. K.4	K.4	Counts exclude checks and cache costs
Prop. 9: record sufficiency	Path-determined predictions; deterministic control; record check completes	Record judgment equals native judgment, in any expansion order	App. K.5	Table 77	Deterministic tree search
FutureCert property	Shared restart prefix; verified incumbent; strict size improvement	$\lvert\widehat{S}_{R+1}\rvert \leq \lvert\widehat{S}_R\rvert$, fresh queries $\leq 96R$	App. B.3	F.1	No exact-optimum or cross-stream guarantee

K.1 Intervention Semantics and Arithmetic

We first characterize what an original execution transcript identifies, then analyze shared-reference arithmetic and finite-precision effects.

K.1.1 Fixed-trajectory information and native interventions

Consider a controller that adaptively queries a prediction oracle M at read sites. The controller code, input x, budget, random stream ξ, evidence partition, replacement values, and output predicate are known. A fixed-trajectory transcript $T_{\text{fixed}}(M)$ contains the complete original execution: queried sites and candidates, their prediction values, intermediate controller states, and final output. It can be reprocessed arbitrarily, but supplies neither oracle evaluations at unvisited sites nor parameters from which those values can be computed. The restriction concerns information access, even when controller code and RNG are known.

Proof of Proposition 1. There exist a controller, two prediction oracles M_0, M_1, and a retained set S such that $T_{\text{fixed}}(M_0) = T_{\text{fixed}}(M_1)$ but $P_{S,M_0}(x) \neq P_{S,M_1}(x)$. Here $P_{S,M}$ denotes the intervened planner using oracle M. Their original outputs agree, and their native action-preservation predicates for S differ. Hence no function of this transcript and the common audit specification can identify the native intervention output, or decide its sufficiency correctly, for every oracle in this class.

Proof. Use two read sites u, v, in separate evidence units, and the same deterministic controller for both oracles. It reads u; if the value is one, it outputs action zero and stops. Otherwise it reads

v and outputs its value as a binary action. Set $M_0(u) = M_1(u) = 1$, $M_0(v) = 0$, and $M_1(v) = 1$. The original executions both observe only $u = 1$ and output zero, so their complete fixed transcripts coincide. Choose replacement zero and retain $S = \{v\}$. Intervened execution reads the replacement at u, visits v, and outputs zero for M_0 and one for M_1. Exact action preservation therefore passes only for M_0. An estimator with identical observable inputs must give the same answer in both cases, so it cannot be correct in both. A randomized estimator likewise cannot be correct with probability one in both; under an equal mixture its binary success probability is at most one half. All computations use exact binary values. □

The result is a worst-case information statement for a trace-restricted interface. Model access, additional structural constraints, or certified state equivalence can reveal the missing information. In particular, the exact-reuse conditions of Proposition 4 are compatible with the proposition. The construction establishes neither a lower bound on the number of model evaluations nor a claim that every intervention changes the original path. Our final-set experiments test specific fixed-candidate and partially refreshed interfaces beyond this construction.

K.2 Independent Reliability Calibration

A shared reliable set is selected before the planning stream is drawn. We give a size-separation construction and then state the independent calibration guarantee.

K.2.1 Execution-conditioned and stochastic certificate sizes

For a fixed input and audit specification, define $k^\star_{\mathrm{path}}(\xi) = \min\{|S| : C_{\Gamma,\xi}(S) = 1\}$ and $k^\star_\tau = \min\{|S| : p_\Gamma(S) \geq \tau\}$. The latter chooses one shared set before drawing the planning stream. Each stream uses its own full-retention output as the fidelity target.

Proof of Proposition 2. For every integer $H \geq 1$ and threshold $\tau \in (0, 1]$, there exists a stochastic read-site planner with H evidence units such that $k^\star_{\mathrm{path}}(\xi) = 1$ on every stream, whereas $k^\star_\tau = \lceil \tau H \rceil$.

Proof. Let the stream select J uniformly from $\{1, \ldots, H\}$. The planner reads unit J, whose fresh value is one, and returns that value as its binary action. Every replacement is zero, and the predicate requires exact equality to the same-stream full-retention action, which is always one. Thus $C_{\Gamma,J}(S) = \mathbf{1}[J \in S]$. On each stream, the singleton $\{J\}$ passes and the empty set fails, giving pathwise minimum one. For a set chosen independently of J, $p_\Gamma(S) = |S|/H$. The reliability constraint is therefore equivalent to $|S| \geq \lceil \tau H \rceil$, and any set of that size attains it. □

The separation comes from the order of selection and randomization: $\forall \xi \, \exists S_\xi$ does not provide a small shared S. A stream-adaptive selection policy is a different object. At $\tau = 1$ the shared minimum is H; at $\tau = 0.8$ it is $\lceil 0.8H \rceil$. The result is an existence construction, whereas the finite-stream exhaustive experiment measures actual planners and independent calibration evaluates a frozen candidate pool. Calibration can fail to certify a candidate with adequate true reliability because it uses a finite-sample lower bound.

Compact certificates with a finite-sample reliability bound. A set can be verified for a fixed stream yet fail under a new one. We therefore freeze M distinct non-full candidates before drawing n independent calibration streams from a specified distribution $\mathcal{Q}$. Each stream compares the intervened output against its own full-retention output. For m_j passes of candidate S_j, define the one-sided Clopper-Pearson bound (Clopper & Pearson, 1934)

$$L_j = \begin{cases} 0, & m_j = 0, \\ \mathrm{Beta}^{-1}(\alpha/M; m_j, n - m_j + 1), & m_j > 0. \end{cases} \tag{15}$$

We return the smallest S_j with $L_j \geq \tau$, or report that no compact certificate was obtained. Full retention is excluded from this output class.

Guarantee of Proposition 3. Conditional on the frozen candidates and planning input, the probability over calibration that any reported candidate has true $\mathcal{Q}$-pass probability below τ is at most α.

Each exact lower bound has error probability at most α/M; a union bound covers selection among the candidates. This is a standard binomial construction applied to the native verifier. Its guarantee concerns random streams for that input, reference, and predicate. Calibration calls are additional to candidate search, and independent test streams evaluate coverage and compactness (Section 5.5).

For Proposition 3, condition on all data used to choose $S_1, \ldots, S_M$ and let $p_j = \Pr_{\xi \sim \mathcal{Q}}(C_\xi(S_j) = 1)$. For fixed j, its n fresh indicators are IID Bernoulli(p_j), and exact binomial inversion gives $\Pr(L_j > p_j) \le \alpha/M$. Dependence across candidates is allowed, including sharing streams and full targets. Thus $\Pr(\exists j : L_j > p_j) \le \alpha$. On the complementary event, selecting any candidate with $L_j \ge \tau$ implies $p_j \ge \tau$. If $M = 0$, the procedure abstains. Candidates, multiplicity, and sample size are fixed before calibration. Accepted candidates receive the full declared sample: 64 streams in the compact studies and 200 in the exhaustive-stream and two-mode studies. Early rejection is permitted when even success on every remaining stream cannot reach the fixed-sample confidence gate; it preserves the final decision and never permits early acceptance. Changing candidates or accepting at a data-dependent sample size requires a bound valid for that procedure. Confidence is per input and simultaneous over its declared pool. The compact studies exclude full retention, whereas the exhaustive-stream and two-mode studies include it as an explicit endpoint.

For Equation 13, every globally minimum certificate is on the size-$k^\star$ layer. Removing all sets that intersect D leaves exactly $\binom{H-|D|}{k^\star}$ potential witnesses. Direct verification of each remaining set is sufficient for an exact existence answer, without assuming monotonicity. This query asks for an equally small alternative: a negative answer leaves open whether a larger certificate avoids D. Prefix families can establish existence with a witness; lack of a prefix witness is unknown, unless the restricted layer has separately been exhausted.

K.3 Intersection Witnesses and Proof Records

Propositions 5 and 6 state the intersection-witness and proof-record bounds. Their statements and proofs appear together with the family experiments in Appendix G.1.

K.4 Certified-prefix Reuse

Proposition 1 says that the original transcript cannot decide sufficiency, not that every evaluation must be recomputed. This subsection makes precise which evaluations of an intervened execution can be served from the full-retention run and measures how many that is for TD-MPC2.

Setting. Fix x, θ, b, ξ, Γ and a retained set $S \subseteq \mathcal{E}_x$. We view the native intervened execution $P_S(x; \theta, b, \xi, \bar{z})$ as a sequence of transitions of an execution state σ that contains the latents, the search or optimizer state, read-site bookkeeping, the warm start, all mutable buffers, and the complete random-number state ρ (every generator the program may touch). Transitions are of two kinds. A *model-evaluation event* $e = (o, u)$ applies an operator o to input tensors u. The operator is a network head with parameters θ together with its fixed post-processing and numerical configuration (device, dtype, kernel and determinism settings). Started in random state ρ^-, the event returns a *raw* output y and leaves random state ρ^+. Every other transition is a *planner transition*. In particular, serving a read of unit j by $\tilde{z}_j(S)$ in Eq. (3) is a planner transition applied to the raw output of the event at that read site. As Γ requires, the raw evaluation still runs at replaced sites. Let $e_1^S, \ldots, e_{L_S}^S$ be the model-evaluation events of P_S in program order. Thus L_S counts the events of the *intervened* execution, not of the full-retention execution. The *determining inputs* of an event are $d(e) = (o, u, \rho^-)$, and all comparisons of determining inputs are bitwise (dtype, shape and bytes). A cache is a finite list of records (d_k, y_k, ρ_k^+) of events executed earlier. An example is the full-retention execution that produced $y_\xi^\star$ from the same x, warm start, b and ξ. A fixed map c assigns to each event index j a candidate record $c(j)$, for example the record at the same position.

Assumptions. (A1) *Reproducible operators.* In any execution, applying o to bitwise-equal (u, ρ^-) under the same numerical configuration returns a bitwise-equal raw output and post-state ρ^+, and changes no other part of σ. (A2) *Deterministic planner.* Every planner transition, including the choice of the next transition and termination, is a deterministic function of (σ, S, Γ).

Certified-prefix reuse. The procedure $\widehat{P}_S$ runs the program of P_S and executes every planner transition natively. At its j-th model-evaluation event, it checks two conditions: events $1, \dots, j-1$ were all reused, and the current determining inputs equal $d_{c(j)}$. If both hold, it returns $y_{c(j)}$ and sets the random state to $\rho^+_{c(j)}$ without evaluating o. Otherwise it evaluates this event and every later event natively. Let p denote the number of reused events.

Proposition 8 (certified-prefix reuse). Under (A1) and (A2) the following hold. (i) $\widehat{P}_S$ returns $y_{S,\xi}$ bit for bit. Its final execution state also equals that of P_S, including the random state and the warm start handed to the next decision. (ii) $p = \max\{q \in \{0, \dots, L_S\} : d(e^S_j) = d_{c(j)} \text{ for all } j \le q\}$. So p is the length of the leading run of events of the native intervened execution whose determining inputs match the cache. (iii) $\widehat{P}_S$ performs exactly $L_S - p$ model evaluations. It additionally performs $\min(p+1, L_S)$ equality checks, and the cache must be built and stored. If reuse is deliberately truncated after $p' \le p$ successful checks, output equivalence is preserved and $L_S - p'$ model evaluations are performed. The equality-check count follows the stopping rule: stopping immediately after the p'-th success performs p' checks, with no failing check. Consequently, $L_S - p$ is a *conditional upper bound* on the number of new model evaluations needed to reproduce $y_{S,\xi}$ from this cache. It is conditional on (A1), (A2), the cache and the event granularity. It is neither a lower bound on necessary re-execution nor a wall-clock speed-up guarantee.

Proof. Let σ_t and $\widehat{\sigma}_t$ be the states of P_S and $\widehat{P}_S$ after t transitions. They are equal at $t = 0$, because both restore x, the warm start, b and ξ and use the same S and Γ. Suppose $\sigma_t = \widehat{\sigma}_t$. By (A2), both executions take the same next transition. If it is a planner transition, then $\widehat{P}_S$ executes it natively and (A2) gives $\sigma_{t+1} = \widehat{\sigma}_{t+1}$. If it is the j-th model-evaluation event, then $\widehat{P}_S$ reads its determining inputs from $\widehat{\sigma}_t = \sigma_t$, so they equal $d(e^S_j)$. Suppose $\widehat{P}_S$ reuses the event. Then $d(e^S_j) = d_{c(j)}$. The record $c(j)$ was produced by evaluating the same operator on the same inputs from the same random state, so by (A1) the native evaluation in P_S returns $y_{c(j)}$ and leaves $\rho^+_{c(j)}$ with no other effect. This is exactly the update that $\widehat{P}_S$ installs. Suppose instead that $\widehat{P}_S$ evaluates the event natively. Then both executions apply the same operator to the same inputs and random state, and (A1) again gives equal results. In both cases $\sigma_{t+1} = \widehat{\sigma}_{t+1}$. By induction, the two executions take identical transition sequences, terminate at the same step and have equal final states, which gives (i). The check at event j compares $d(e^S_j)$ with $d_{c(j)}$, and reuse stops at the first failure. So exactly the events $1, \dots, p$ of (ii) are reused. Both executions contain the same L_S events, so $L_S - p$ of them are evaluated. There are p successful checks, plus one failing check if $p < L_S$, which gives (iii). Stopping earlier at $p' \le p$ changes only which events take the native branch of the induction. □

Remarks. *Raw versus served values.* The records hold raw evaluations, and the reuse procedure re-applies the replacement rule natively. Hence retained or removed status is not part of $d(e)$, and a record from a site that was retained in the full-retention run can be reused at the same site after removal. A cache that stores served values $\tilde{z}_j(S')$ must add $\mathbf{1}[j \in S]$ to $d(e)$. *Randomness at replaced sites.* The premise needs the random state before each event, and Γ runs randomness-consuming computation at replaced sites. An implementation that skipped those evaluations would be a different program from P_S and is not covered. *Online identification.* The check uses the reuse execution's own state, which equals the native state by the induction. No future divergence point has to be known, and a single failing comparison ends reuse. Outputs are never compared. *Not a lower bound.* The bound can be loose for two reasons. First, later events may match cached records again, and the general exact-replay cache of Proposition 4 would reuse them. Second, a suffix event can be dead: at a read site of a removed unit the raw output is discarded, so only its random-state advancement affects the output. *Not a speed-up.* The count excludes checks, cache construction and memory traffic. *Path length.* In tree search, S changes which nodes are expanded, so in general $L_S \neq L_{\mathcal{E}_x}$. The number of new evaluations is therefore not "original length minus first divergence".

MPPI instance. In the TD-MPC2 MPPI planner, horizon 3 and 6 iterations give $L_S = 6 + 8 \cdot 6 = 54$ events for every S. The first six events are the encoder and the policy-prior rollout. Each iteration then contains $R_0, D_0, R_1, D_1, R_2, D_2, \pi, Q$. A replaced value enters only the candidate returns, and these affect later event inputs only through the elite update at the end of the iteration. Let $\varphi_{\min}(S)$ be the earliest phase (pair of iterations) that contains a removed unit, and take the full-retention

cache with $c(j) = j$. Then every event through the first iteration of that phase has S-independent determining inputs. With phases numbered $0, 1, 2$ and iterations numbered $1, \ldots, 6$, this is iteration $2\varphi_{\min}(S) + 1$, giving $p \geq 6 + 8(2\varphi_{\min}(S) + 1) = 14 + 16\,\varphi_{\min}(S)$ from the program structure. In every query below, the first update after a removal changes the state bitwise and the bound holds with equality.

Measurement. We re-executed all 40,277 charged queries of the 400 default FutureCert runs on Cheetah Run and Cup Catch natively with per-event instrumentation. The cache was the full-retention execution with $c(j) = j$. Every query had $L_S = 54$, and $p = 0$ never occurred. The ratio p/L_S had mean 0.264, median 0.259 ($= 14/54$), 5th and 95th percentiles both 0.259, and maximum 0.852 ($= 46/54$). The suffix fraction $(L_S - p)/L_S$ had mean 0.736. The prefix is long only when every removed unit belongs to a late phase: $p/L_S = 0.259$, 0.556 and 0.852 for $\varphi_{\min} = 0, 1, 2$. These phases account for 98.5%, 1.3% and 0.2% of the queries. After the first mismatch no event matched any cached record again, whether matched by position or by content. The reuse of the exact-replay backend of Appendix B.4 is therefore exactly this certified prefix, at the coarser granularity of MPPI iterations. Reusing exactly p events and executing the suffix natively reproduced, bit for bit in every query, the action, the optimized mean (the warm start), the final standard deviation, the selected elite and the CPU and CUDA random states. Forward hooks counted exactly $L_S - p$ model evaluations. As a negative control, we forced reuse of one further event. The output changed in 919 of 922 controls where that event was live, and in none of 1,174 where it was a read site of a removed unit.

K.5 Record Sufficiency under Path-keyed Replay

Proposition 8 restricts reuse to a matching prefix, whereas Proposition 4 permits reuse whenever a cached evaluation has matching determining inputs, including after an earlier miss. In tree search, an intervention can expand the same nodes in a different order. The path-keyed check below uses node identity to establish the required match and agrees with native re-execution whenever it completes.

Setting. Consider deterministic tree search as in MuZero. Fix x, θ, b, ξ, Γ and a retained set S. A node is identified by its action path $\pi = (a_1, \ldots, a_d)$ from the root; the implementation obtains its latent by applying the dynamics network along π to the root representation and reads the raw reward $r(\pi)$, value $v(\pi)$, and policy logits $\ell(\pi)$. The record $\mathcal{R}$ of the full-retention execution stores the root outputs, the search seed, and $(r(\pi), v(\pi), \ell(\pi))$ for every node π it expanded.

Assumptions. (B1) *Path-determined predictions.* Under the fixed numerical configuration, the raw outputs at a node are a deterministic function of its action path: they are bitwise equal whenever the path is equal, regardless of when the node is expanded or which other nodes exist. (B2) *Deterministic control.* Selection, including tie breaking by the seeded search generator, expansion, and backup are deterministic functions of the tree statistics, the served reward and value reads, the policy logits, and the generator state; model evaluations consume no randomness. The intervention changes only served reward and value reads through Eq. (3) and leaves the policy logits unchanged.

Record check. The check R_S runs the planner's own selection, expansion, and backup with the original seed. When a simulation expands node π, R_S looks up π in $\mathcal{R}$. If π is absent, it stops and returns *unknown*. Otherwise it serves the read rule of π's unit applied to the recorded raw reward and value, together with the recorded logits. If all b simulations complete, it returns the root visit distribution and the selected action. The check makes no model call.

Proposition 9 (record sufficiency). Under (B1) and (B2), if R_S completes, its output equals the native intervened output $y_{S,\xi}$, and therefore Φ applied to it equals $C_{\Gamma,\xi}(S)$. This holds even when R_S expands nodes in a different order from the full-retention execution.

Proof. Let T_t and $\widehat{T}_t$ denote the complete search states of P_S and R_S after t simulations: expanded nodes, their statistics, the min-max normalization, and the generator state. Both prepare the root from the same outputs with the same seed; the full-retention replay check below confirms that the recorded root equals the native one. Hence $T_0 = \widehat{T}_0$. Suppose $T_t = \widehat{T}_t$. By (B2), simulation

$t+1$ traverses the same path in both executions, because selection depends only on the search state, and both reach the same unexpanded node π. Because R_S completed, $\pi \in \mathcal{R}$, and by (B1) the native evaluation at π returns exactly the recorded raw outputs. The read rule of π's unit then serves identical reward and value in both executions, and the logits coincide. Expansion and backup apply the same deterministic operations to equal inputs, so $T_{t+1} = \widehat{T}_{t+1}$. After b simulations the states agree, and so do the root visit distributions, the selected action, and the fidelity judgment. The induction never compares simulation indices, so a node expanded at a different simulation than in the full-retention execution is still served correctly. □

Remarks. *Sufficient, not necessary.* An *unknown* outcome carries no information about sufficiency; the verifier then re-executes natively. *Relation to exact reuse.* Proposition 9 identifies an evaluation by its action path. Under (B1), this identity establishes matching predictions regardless of expansion order; unlike Proposition 8, it does not stop at the first change in ordering. *Scope.* (B1) applies to MuZero-style search with a deterministic dynamics network; planners that draw their candidates from the random stream, such as Sampled EfficientZero during expansion and MPPI, use the native verifier instead.

Measurement. For every decision of MuZero Pong, Breakout, and MsPacman, we first replay full retention from the record and require bitwise-equal root visits and tree state; this holds on all 1,062 inputs. We then probe the empty set, two random masks per input, and, on Pong and Breakout, every set committed by the frozen-tree pipelines of Appendix D.4, and re-execute each probe natively. Table 77 compares the two judgments.

Table 77: **Record check versus native re-execution on MuZero.** Non-full probes per setting; *Completed* counts probes on which the record check returns an output, regardless of whether it satisfies fidelity; *Reordered* counts completed probes with a different expansion order from full retention; *Wrong* counts completed probes whose root visits differ from native re-execution.

Setting	Inputs	Non-full probes	Completed	Reordered	Wrong
MuZero Pong	338	1,609	19	17	0
MuZero Breakout	373	2,003	89	76	0
MuZero MsPacman	351	906	1	0	0

L Diagnosis and Fault Repair

These additional experiments evaluate diagnostic features and intervention-based repairs against independent targets. Repair of injected prediction faults is separate from the certificate-availability queries in Appendix G.2.

L.1 Diagnostic Targets and Protocols

The diagnostic target is action change after reducing the planning budget from 50 to 30 simulations. Episode-held-out models compare original planner statistics, inexpensive native probes, and certificate features; the following tables specify the feature sets and sample filters.

L.2 Incremental Diagnostic Information

Independent planning-budget diagnostic Before collecting any 30-simulation outcomes, we froze the population, target, initial feature groups, model, and split. The population contains every unique-action decision in the three core exact-family archives. The target is whether the deployed action changes when the same observation, model, search seed, and random stream are rerun with 30 rather than 50 simulations. The subsequent cheap-probe decomposition was separately frozen before its results were computed. Predictors use only 50-simulation outputs and interventions. We fit logistic regression with fixed inverse regularization strength $C = 1$ and leave one complete episode out at a time.

For every held-out episode, we hash the exact observation tensor and remove training rows with a matching hash. The three archives contain 320/338, 178/373, and 381/383 unique observations; 9, 168, and 4 rows have a hash that occurs in another episode. This purge changes the raw Breakout area under the precision-recall curve (AUPRC) from 0.272 to 0.252 and is used for all reported nested comparisons. Paired intervals resample the five held-out episode blocks 3,000 times.

The raw-output model uses the original visit margin, visit entropy, realized horizon, and top-visit share. The empty response contains action retention, declared-predicate acceptance, rank correlation, normalized margin error, and visit total-variation distance. If the empty set fails, the cheap policy queries every singleton and summarizes the corresponding action, predicate, rank, margin, and visit responses; if it passes, the policy stops after one query. Certificate variables are then added one at a time as shown in Table 79.

Table 78: Selected stages of the hash-purged planning-budget diagnostic. AUPRC and AUROC are higher-is-better (↑); Brier score is lower-is-better (↓). The detailed diagnostic tables report every intervening certificate stage.

MuZero Pong

Stage	Added information	AUPRC ↑	AUROC ↑	Brier ↓
Raw	Planner outputs	0.247	0.764	0.089
+ Empty action	One binary response	0.297	0.780	0.088
+ Empty response	Same one query	0.404	0.820	0.082
+ Singletons	Adaptive cheap probes	0.483	0.875	0.076
+ All structure	Through preference Core	0.471	0.866	0.076

MuZero Breakout

Stage	Added information	AUPRC ↑	AUROC ↑	Brier ↓
Raw	Planner outputs	0.252	0.816	0.098
+ Empty action	One binary response	0.318	0.853	0.091
+ Empty response	Same one query	0.248	0.767	0.105
+ Singletons	Adaptive cheap probes	0.234	0.733	0.120
+ All structure	Through preference Core	0.247	0.729	0.116

SEZ Pong

Stage	Added information	AUPRC ↑	AUROC ↑	Brier ↓
Raw	Planner outputs	0.259	0.834	0.077
+ Empty action	One binary response	0.384	0.863	0.071
+ Empty response	Same one query	0.365	0.867	0.073
+ Singletons	Adaptive cheap probes	0.407	0.885	0.070
+ All structure	Through preference Core	0.395	0.890	0.070

Independent-reference family and Core transfer The independent reference bank was collected from separate reference episodes, after which every minimum family was recomputed exactly for the same evaluation decisions. This yields an external target for the original family: which units remain shared by all minima, and how frequently each unit appears among the new minima. Unit-level models hold out complete episodes and apply the same observation-hash purge. Core membership uses fixed-C logistic regression; family frequency uses ridge regression with coefficient 1.0. No model or feature choice is tuned from transfer outcomes.

The comparison proceeds from unit position, to empty/singleton native responses, to one lexicographically chosen minimum certificate, and finally to unit frequency across the complete original family. Complete-family minus one-certificate AUPRC differences are +0.008 $[0.002, 0.016]$, +0.005 $[-0.002, 0.011]$, and +0.008 $[0.002, 0.015]$. The corresponding family-frequency MAE differences are -0.043 $[-0.052, -0.029]$, -0.030 $[-0.036, -0.024]$, and -0.014 $[-0.018, -0.011]$. Intervals use 3,000 episode-block resamples and keep every unit of a decision together. Table 80 gives the point estimates and direct Core transfer.

Table 79: Held-out AUPRC (↑) for the independently collected 50-to-30-simulation action change. Rows are strictly nested. $k^\star$, family, and Core refer first to the action predicate and then to the declared preference predicate. Bold marks the best unrounded value per setting.

Added information	MuZero Pong	MuZero Breakout	SEZ Pong
Raw planner outputs	0.247	0.252	0.259
+ Empty-set action indicator	0.297	**0.318**	0.384
+ Remaining empty-set response	0.404	0.248	0.365
+ Adaptive singleton probes	**0.483**	0.234	**0.407**
+ Action $k^\star$	0.478	0.236	0.407
+ Action family count	0.466	0.239	0.394
+ Action Core	0.469	0.236	0.394
+ Preference $k^\star$	0.483	0.244	0.389
+ Preference family count	0.477	0.243	0.387
+ Preference Core	0.471	0.247	0.395

Table 80: Transfer to an independently constructed reference bank. AUPRC predicts membership in the new Core (↑). Frequency MAE predicts membership frequency in the new minimum family (↓). Core precision/recall directly compares the original and new Cores.

(a) Independent Core-membership prediction

Setting	Cheap AUPRC ↑	One-cert. AUPRC ↑	Full-family AUPRC ↑	Family Δ ↑ [95% CI]
MuZero Pong	0.820	0.991	**0.999**	+0.008 [0.002,0.016]
MuZero Breakout	0.657	0.889	**0.894**	+0.005 [−0.002,0.011]
SEZ Pong	0.719	0.884	**0.891**	+0.008 [0.002,0.015]

(b) Independent family-frequency and Core transfer

Setting	One-cert. MAE ↓	Full-family MAE ↓	Core precision ↑	Core recall ↑
MuZero Pong	0.049	**0.007**	0.997	0.999
MuZero Breakout	0.165	**0.134**	0.937	0.917
SEZ Pong	0.251	**0.237**	0.846	0.822

L.3 Fault Localization and Repair

L.3.1 Samples, ordering, and repair operators

The analysis uses 201, 347, and 384 single-depth faults, totaling 932. For every second unique-action state with realized horizon at least three, we perturb the clean runner-up root branch at each depth by adding the smallest value offset in $\{0.5, 1, 2, 4\}\sigma_d$ that produces a unique action flip; σ_d is the reference-value standard deviation at depth d. The interval extension samples clean states with $H \geq 8$, uses widths two and three at frozen early, middle, and late positions, and applies positive offsets in $\{1, 2, 4\}\sigma_d$ at every depth in the window. It adds 368 MuZero Pong and 78 SEZ Pong instances, for 1,378 faults overall. All planner executions use 50 simulations. Instances are retained after injection only when the deployed action changes to a unique top action, so recovery rates are conditional on a successful controlled intervention. We average instances within a decision, decisions within an episode, and then weight episodes equally.

For a faulted decision, FutureCert probes each depth in the realized search: it replaces only that depth's reward and value predictions with locked reference means and fully re-executes the planner. Let a_{-d}, v_{-d} be the resulting action and root-visit vector, and $a_{\text{fault}}, v_{\text{fault}}$ the faulted output. The depth score is

$$q_d = 0.5\,\mathbf{1}[a_{-d} \neq a_{\text{fault}}] + 0.5\frac{\|v_{-d} - v_{\text{fault}}\|_1}{\|v_{\text{fault}}\|_1 + \epsilon}. \tag{16}$$

Depths are ranked by q_d; ties use minimum-certificate-family support, then the Core, then shallower depth. We jointly repair the top k depths and re-execute. Every method is given $k = |W|$, the true fault-window width. FutureCert ranks using only the faulted plan and probe outputs; action recovery compares with the unique pre-fault deployed action. A negative control removes the certificate-family and Core tie-breaks. Across 1,378 faults this changes the selected set in 9 cases and changes

the action outcome in 2 cases, once in each direction; aggregate action-recovery rates are unchanged. The repair results therefore support the intervention probe, not a causal claim about the minimum-certificate family.

R-ref replaces faulty predictions at selected depths with reference means. R-undo reverses the injected offset on the intersection of selected depths with the true fault window. The tables report action and preference recovery for single-depth and interval faults; preference recovery also requires pre-fault visit ranking $\rho \geq 0.95$ and root visit-count margin error $\mathrm{NME} \leq 0.10$.

L.3.2 Full recovery under reference replacement

FutureCert reaches 80.4%, 90.1%, and 82.2% on the three single-depth settings in Table 81. On the interval-fault settings, KernelSHAP reaches 92.8% and 88.4% action recovery, compared with FutureCert's 91.1% and 84.2%. The interval-fault comparison complements the single-depth study in Table 81.

Table 81: Action and preference recovery under reference replacement (%). FL-1 has single-depth and FL-2 interval faults; all methods select $|W|$ depths.

Single-depth faults: MuZero

	FL-1 MuZero Pong		FL-1 MuZero Breakout	
Method	Action ↑	Preference ↑	Action ↑	Preference ↑
FutureCert	**80.4**	**7.5**	**90.1**	**24.9**
KernelSHAP	69.2	4.5	82.8	23.8
LIME	70.8	5.0	80.8	21.5
RISE	76.5	5.2	88.7	22.9

SEZ single-depth and MuZero interval faults

	FL-1 SEZ Pong		FL-2 MuZero Pong	
Method	Action ↑	Preference ↑	Action ↑	Preference ↑
FutureCert	**82.2**	**14.6**	91.1	**7.1**
KernelSHAP	71.5	12.4	**92.8**	6.9
LIME	73.7	12.8	92.0	6.7
RISE	79.1	14.0	92.2	6.3

SEZ interval faults

	FL-2 SEZ Pong	
Method	Action ↑	Preference ↑
FutureCert	84.2	7.8
KernelSHAP	**88.4**	6.1
LIME	84.9	**8.7**
RISE	84.4	8.6

L.3.3 Reversing the injected offset

Table 82 keeps the same faults and depth rankings but switches the operator to R-undo. It changes predictions only when a selected depth intersects the true window, so it measures coverage of the injected offset by the ranking. Samples, budgets, and statistical weights match Table 81.

L.3.4 Cases and paired differences

Table 83 shows cases from MuZero Pong, MuZero Breakout, and SEZ Pong. FutureCert selects the fault depth in all three; RISE also recovers the action in the Breakout and SEZ Pong cases. Figure 13 gives paired action-recovery differences; all intervals against KernelSHAP lie above zero, while the Breakout interval against RISE crosses zero.

Table 82: Action and preference recovery under injection-offset reversal (%). Samples, budgets, and weights match Table 81.

Single-depth faults: MuZero

Method	FL-1 MuZero Pong Action ↑	FL-1 MuZero Pong Preference ↑	FL-1 MuZero Breakout Action ↑	FL-1 MuZero Breakout Preference ↑
FutureCert	64.7	64.7	84.5	84.5
KernelSHAP	56.3	56.3	80.2	80.2
LIME	53.9	53.9	78.2	78.2
RISE	**71.5**	**71.5**	**87.9**	**87.9**

SEZ single-depth and MuZero interval faults

Method	FL-1 SEZ Pong Action ↑	FL-1 SEZ Pong Preference ↑	FL-2 MuZero Pong Action ↑	FL-2 MuZero Pong Preference ↑
FutureCert	76.4	76.4	70.4	64.7
KernelSHAP	72.8	72.8	**71.7**	**67.7**
LIME	68.2	68.2	71.1	63.1
RISE	**84.8**	**84.8**	57.7	37.9

SEZ interval faults

Method	FL-2 SEZ Pong Action ↑	FL-2 SEZ Pong Preference ↑
FutureCert	91.0	67.1
KernelSHAP	94.2	69.9
LIME	**96.5**	**82.4**
RISE	87.5	27.7

Table 83: Selected repair depths and action recovery in three controlled single-depth fault cases.

Case	Fault depth	FutureCert	KernelSHAP	LIME	RISE
MuZero Pong	2	2 Yes	3 No	3 No	4 No
MuZero Breakout	2	2 Yes	3 No	3 No	2 Yes
SEZ Pong	3	3 Yes	4 No	3 Yes	3 Yes

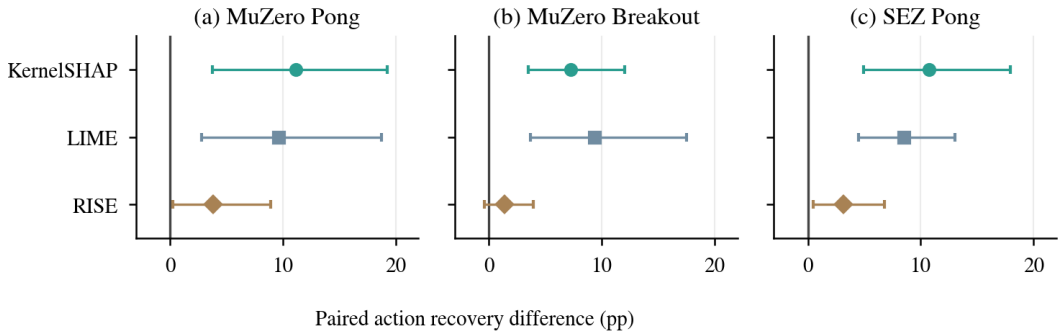


Figure 13: Paired action-recovery differences on controlled single-depth faults. The horizontal axis is FutureCert minus comparator in percentage points; bars are paired 95% intervals.

L.4 Online Compute Savings from Certificates

Skipping. Given a verified retained set, a planner can supply the declared references elsewhere and skip network forwards whose outputs cannot affect the masked execution. A reward head runs only for retained reads, dynamics steps run only up to the deepest retained read in each phase, and the policy and Q ensemble run only when the terminal value is retained; when they are skipped, the planner still draws the same random numbers. On 60 archived Cheetah Run and Cup Catch inputs and 2,123 (input, set) pairs, the skipping planner reproduces the masked intervention bitwise in action, mean, standard deviation, all elite indices, per-iteration values, and CUDA and CPU random states, and forward counts match an analytic dependency model. With the exact minimum certificate it would avoid 61%, 61%, and 93% of batch-512 forwards on Cheetah Run, Cup Catch, and Finger Spin (by the analytic dependency model), but an exact certificate requires the audit itself.

Closed loop. A deployable controller predicts a retained mask before planning. We retrieve the nearest development states using only pre-planning features and choose the cheapest mask verified on all of them; a fallback reverts to full planning far from development data; an aggressive arm ignores development validity. These are predicted masks on new inputs: development validity does not establish fidelity to the new input's full planner. Skipping preserves the corresponding masked execution. All rules were fixed on development data before twenty new episodes per task and controller were run. Table 84 shows the outcome. On Finger Spin, the predicted mask avoids 71% of forwards and halves latency; with the fallback it avoids 50%, cuts latency by 35%, and keeps 96.9% of the full-planning return. On Cheetah Run and Cup Catch the development-validity constraint keeps full planning on most states, saving under 5%; the aggressive arm saves 11-35% at a measurable loss of return. The measured benefit is a task-dependent computation-return trade-off.

Table 84: **Closed-loop control with certificate-guided skipping.** Twenty new episodes per task and controller (seeds disjoint from all archived ones), 500 decisions each. The skipping planner executes only forwards that can affect a retained read and reproduces the masked planner bitwise; every 25th deployed decision is replayed against it (3,600/3,600 equal). The certificate is predicted from development states only, before planning; the fallback reverts to full planning far from development data; the aggressive arm ignores development validity. Return: mean over episodes; paired difference vs. full planning with 95% interval. Agreement: share of decisions whose action is within 0.20 of a shadow full planner (not charged). Saved: share of batch-512 network forwards avoided. Latency is relative to full planning under shared GPU load.

Task	Controller	Return↑	Δ vs. full	Agreement↑	Saved↑	Latency↓
Cheetah Run	Full planning	862.9	-	1.000	0.000	1.00
	Predicted mask	824.0	-38.9 [-91.4, 13.6]	0.861	0.045	0.980
	Predicted + fallback	855.4	-7.5 [-18.5, 3.5]	0.876	0.040	0.984
	Aggressive prediction	827.6	-35.3 [-57.2, -13.4]	0.233	0.346	0.735
Cup Catch	Full planning	984.5	-	1.000	0.000	1.00
	Predicted mask	984.8	0.2 [-0.3, 0.7]	0.950	0.006	1.014
	Predicted + fallback	984.6	0.1 [-0.5, 0.7]	0.953	0.005	1.015
	Aggressive prediction	981.2	-3.4 [-4.4, -2.3]	0.496	0.111	0.927
Finger Spin	Full planning	988.6	-	1.000	0.000	1.00
	Predicted mask	933.7	-55.0 [-64.1, -45.8]	0.735	0.708	0.480
	Predicted + fallback	957.8	-30.9 [-37.1, -24.6]	0.839	0.501	0.649
	Aggressive prediction	843.0	-145.7 [-162.8, -128.5]	0.629	0.859	0.356

M Recovery and Tolerance Profiles

This appendix collects recovery-query, tolerance, and simulation-budget plots supporting Sections 5.3 and 5.6, together with deployment and arithmetic summaries for Sections 5.7 and 5.6.

M.1 Tolerance and Simulation Budget

The predicted-mask deployment uses full planning as fallback on 20 new episodes per task. Finger Spin skips 50.1% of network forwards at 0.65× latency and 96.9% of full-planning return; its paired

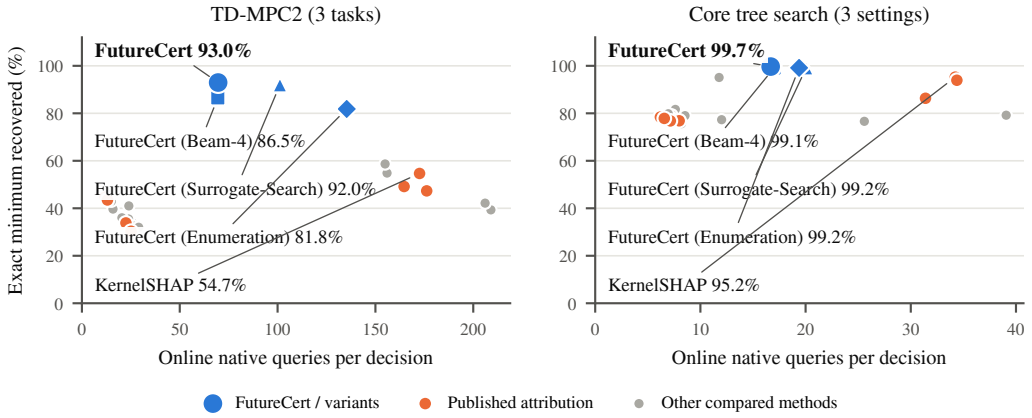


Figure 14: **Exact-minimum recovery against online queries.** Each point is one compared method at the 384-query cap, averaged over the three TD-MPC2 tasks (left) or the three core tree-search settings (right). FutureCert reaches the highest recovery with fewer queries than the native-query attribution pipelines.

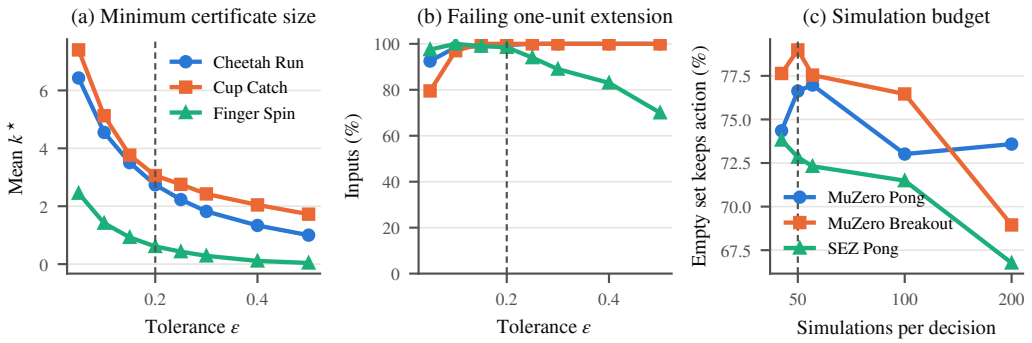


Figure 15: **Tolerance and simulation budget.** (a) Mean minimum size and (b) inputs with a failing one-unit certificate extension versus TD-MPC2 tolerance. (c) Action retention after replacing all reward/value reads versus tree-search simulations. Dashed lines mark defaults: $\epsilon = 0.20$ and 50 simulations.

return difference is -30.9 $[-37.1, -24.6]$. Cheetah Run and Cup Catch mostly retain full planning. Appendix L.4 gives the complete deployment protocol and paired results.

Arithmetic audit populations. The fixed final-set audit rejects 30/30 submitted sets under both FP32 and constant-free arithmetic on each of Cheetah Run and Cup Catch (ten inputs and three explainers per task). A separate Finger Spin audit rejects 267/600 and 271/600 submissions, respectively (200 inputs and three explainers). These counts use different input populations and are not pooled. The re-solved tree-search families remain identical under reference changes on 99.1%, 66.6%, and 42.2% of MuZero Pong, MuZero Breakout, and SEZ Pong decisions (Appendix E.1).