AIRD: Admissibility-Aware Intervention Residual Distillation on a Sparse Tactical Frontier

Derun Ai[1], Laurent Perrussel[1]

[1]IRIT - Université Toulouse Capitole, France

Abstract

Deep self-play reinforcement learning can produce adversarial game agents with superhuman performance, but the resulting neural policies remain difficult to audit. Interpretable policy distillation replaces such policies with decision-tree students, yet conventional distillation often optimizes teacher-replay agreement rather than robustness on the learner-induced tactical frontier, precisely where deployment errors matter most.

We introduce *Admissibility-aware Intervention Residual Distillation* (AIRD), a composite-policy framework that separates routine nominal play from sparse tactical exceptions. AIRD uses a human-interpretable nominal decision tree for routine states and routes selected frontier states to a residual corrective tree through an admissibility-aware intervention interface, without retraining the nominal tree. In an audited deterministic Connect4 5×6 instantiation, exact induced-runtime-graph analysis shows a two-ply improvement in the first unsafe horizon and forced-loss depth over direct deployment of the same VIPER-distilled tree.

Keywords

interpretable reinforcement learning, policy distillation, decision trees, adversarial games, formal analysis

1. Introduction

Deep self-play reinforcement learning has produced high-performing agents in adversarial games [8, 9], but the resulting neural policies are difficult to inspect. In deterministic two-player perfect-information settings, this opacity limits auditability because the deployed policy must be understood under all legal opponent replies. *Interpretable policy distillation* offers one route: a decision-tree student emulates a high-performing neural teacher, yielding a policy whose decisions can be traced and analyzed [1, 3, 2]. A common pitfall is that distilled trees struggle on the sparse, high-stakes tactical states encountered during learner deployment.

Existing distillation methods focus on refining single-policy representations. VIPER, for example, reweights training states by Q-value to prioritize critical scenarios [3]. Later work broadens this monolithic route through programmatic policies, differentiable decision-tree optimization, and formal verification of tree-based policies [4, 5, 6]. Runtime shielding has also been studied as a minimal-intervention mechanism for correcting unsafe reinforcement-learning actions at execution time [10]. None of these extensions challenges a fundamental assumption: should a single readable policy encode both *routine play* and *rare, outcome-critical exceptions*? The two regimes require different approximations. Routine states often admit coarse, high-margin partitions, whereas critical states may require remembering specific danger patterns, forced defensive moves, or value discontinuities. Forcing one tree to handle both creates a size–fidelity tradeoff: increasing the tree may improve coverage, but it also makes the policy harder to audit while still leaving rare frontier failures.

We propose *Admissibility-aware Intervention Residual Distillation* (AIRD), a composite distillation framework for deterministic adversarial games. A nominal decision tree is distilled from a strong teacher and used as the primary auditable policy for routine states. Learner-induced rollouts expose a sparse tactical frontier, where an admissibility-aware selector decides whether the nominal action should be preserved or delegated to a residual corrective branch. This separation improves auditability

8th International Workshop on Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis. OVERLAY 2026 @ FLoC, July 18–19, Lisbon, Portugal

derun.ai@irit.fr (D. Ai); laurent.perrussel@irit.fr (L. Perrussel)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4324/paper10.pdf

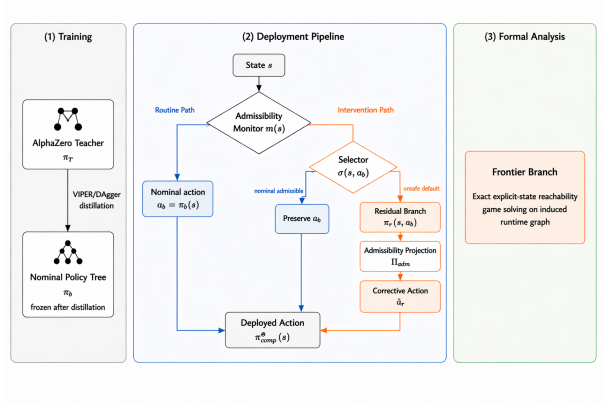


Figure 1: Overview of AIRD as a minimal-intervention composite policy.

relative to the original neural policy and enables post-hoc runtime model checking of the exported composite policy. This paper reports preliminary evidence:

- **Auditable decision-tree policy packaging.** AIRD packages the nominal tree, selector, residual branch, and deployment-time decision semantics as an inspectable policy object. Although the resulting object is larger than a hand-written rule list, its decision logic is encoded by decision-tree components and explicit routing semantics, making it substantially more interpretable than the original neural policy. We illustrate this in deterministic Connect4 5×6.
- **Exact induced-runtime-graph model checking.** We fix the exported policy and expand all legal opponent replies. This exact analysis reports safe horizons, first unsafe horizons, forced-loss depth, and opponent-attractor regions. In Connect4 5×6, AIRD improves the first unsafe horizon and the minimal forced-loss depth by two ply relative to direct deployment of the same VIPER-distilled tree.

2. Method overview

AIRD is a minimal-intervention composite-policy pipeline where key steps are described in Figure 1. In a first stage, a readable nominal tree π_b is distilled from a teacher, through VIPER-style distillation (Fig 1-Step 1): we assume that the neural player is trained and is able to play and win in adversarial games. After training, π_b is frozen and serves as the primary auditable policy for routine states. Later correction is expressed not by retraining π_b, but by adding an intervention mode (Fig 1-Step 2). Given a state s, we first assess if we are in a "routine" context or whether s may denote a "sparse tactical frontier". Typically such states are not well identified by the distilled tree (because of VIPER). Intervention is decided with respect to a set of admissible policies: $\{\psi_j\}$. Each ψ_j maps a game state s, legal-action relation $\mathcal{L}$, and some action a into a binary condition leading to the following definition of the admissibility selector:

$$m(s) = \begin{cases} 1 & \text{(Intervention Path) if } \bigwedge_j \psi_j(s, \mathcal{L}(s), \pi_b(s)) \text{ is } \textit{true} \\ 0 & \text{(Routine Path) otherwise} \end{cases}$$

Example Consider the following defensive admissibility policy ψ in a sequential game such as Connect4. Let $T(s, a)$ be the successor state relation for a state s and an action a, and let $W_{-}(s)$ be the set of opponent actions executable in s and leading to a winning state for player $-$. For a state s, the "blocking actions" admissibility policy ψ is defined as follows (there is a legal action preventing opponent $-$ from winning but action a is not one of them):

$$\psi(s, \mathcal{L}, a) =_{def} \{a' \in \mathcal{L}(s) : W_{-}(T(s, a')) = \varnothing\} \neq \varnothing \text{ and } a \notin \{a' \in \mathcal{L}(s) : W_{-}(T(s, a')) = \varnothing\}$$

Such conditions only require a polynomial complexity and can be used in the supervision pipeline with a low extra-cost; they may use local action masks and one-step successor predicates, rather than invoking MCTS, minimax search, or full game-graph analysis at every candidate state. Selector and residual data are obtained offline from learner-induced frontier records, using bounded or exact labels to identify when the nominal action would be tactically unsafe. The monitor is shield-inspired but not a synthesized temporal-logic shield: it compensates for abstraction errors of a single tree on specific tactical positions, such as missed blocks or forced defensive choices.

- If intervention is not triggered, the initial tree proposes an individual base action $a_b = \pi_b(s)$.
- If intervention is triggered, a residual corrective tree predicts a frontier-local action a_r, so the deployed policy outputs either a_b or a_r.

2.1. Intervention: switching to a new action

The selector σ represented on Fig. 1-Step 2 is actually implemented as a classification tree. Given a state s and an action a, the selector predicts whether the action should be preserved or not, i.e. $\sigma(s, a) \in \{0, 1\}$. In other words $\sigma(s, a_b) = 0$ stands for "preserve the nominal action". If $\sigma(s, a_b) = 1$, then a new action should be selected and executed:

1. we first consider a residual policy π_r encoded as a multiclass CART tree and trained only on routed/intervention states. π_r takes as input a state s and an action a. At runtime, $\pi_r(s, a_b)$ outputs a distribution of probabilities over corrective actions a_r.
2. The list of potential actions is filtered out: only legal actions are preserved (as the residual tree does not consider the legality dimension). This is denoted on Figure 1 by Π_{adm} (Admissibility Projection).
3. Based on the list of actions which are admissible, the most probable one is selected as the *Corrective Action* $\hat{a}_r$.

The resulting composite policy combining Routine and Sparse cases is π_{comp}. This policy can then be audited for explaining the behavior of a neural player.

2.2. Game solving

AIRD exports the resulting policies as an aggregated inspectable policy for local audit and later runtime model checking (Fig. 1-Step 3).

2.2.1. Auditing the composite policy

Auditability here is component-level rather than state-by-state manual inspection. The exported AIRD object records the nominal decision tree, the selector, the residual corrective tree, and the admissibility projection used at deployment time. This does not make every reachable state simple to inspect individually, but it replaces the original neural policy with explicit decision-tree components and routing semantics that can be reconstructed, queried, and checked separately.

2.2.2. Runtime model checking

In the deterministic two-player perfect-information setting studied here, the exported player policy is fixed and the analysis expands all legal opponent replies. The resulting finite induced game graph is then solved adversarially, allowing us to report safe horizons, forced-loss witnesses, and opponent-attractor regions. Stochastic transitions are outside this experimental setting.

Quantity	VIPER tree	AIRD policy	Note
Loss rate vs random	8.9%	1.3%	1000-game evaluation
Reachable graph states	1,501,998	2,536,945	fixed player, legal opponent moves
Largest safe horizon	8	10	induced-runtime-graph model checking
First unsafe horizon	10	12	induced-runtime-graph model checking
Full min forced-loss depth	10	12	full exact adversarial solving
Opponent-attractor fraction	48.05%	23.46%	force-loss region in reachable graph

Table 1
Empirical and formal comparison in the audited Connect4 5×6 setting. 'VIPER-tree' denotes direct deployment of the AlphaZero-to-VIPER distilled CART player; 'AIRD policy' denotes the residual composite player.

3. Experimenting with AIRD in the Connect4 game

We evaluate AIRD on the audited Connect4 5×6 instantiation. The teacher is an AlphaZero-style policy/value network in the `alpha-zero-general` lineage [8, 9], using a 30-cell board input, four 512-unit hidden layers, and separate policy and value heads. The nominal player is a VIPER-style CART policy tree [2, 3], distilled with 10 DAgger-style iterations.

In 1000 games against a random opponent, the AIRD player lost 1.3% of games, while the corresponding VIPER-style CART baseline lost 8.9%. Under exact runtime analysis, AIRD induces a larger reachable graph: $2{,}536{,}945$ states, compared with $1{,}501{,}998$ for VIPER-tree deployment. The reachable graph is the space obtained by fixing the student player policy and expanding all legal opponent replies.

Table 1 summarizes the exact induced-runtime-graph model checking results. Both policies share the same AlphaZero-to-VIPER distilled tree origin and the same explicit-state adversarial solving route. The opponent-attractor fraction denotes the fraction of reachable states from which an adversarial opponent can force a loss. AIRD delays the first unsafe horizon from 10 to 12 ply; AIRD also reduces the opponent-attractor fraction from 48.05% to 23.46% despite the larger induced set of states. If the selector misroutes an unseen state, AIRD does not by itself provide a universal correctness guarantee; its value in this setting is that the composite policy improves over traditional single-tree deployment by correcting specific tactical failure modes exposed on the learner-induced frontier.

Table 2
Policy-only runtime cost over 1,000 Connect4 5x6 games against a random opponent. Timing excludes AlphaZero, MCTS, oracle-agreement computation, model loading, queue time, and file transfer.

Policy	Losses	Loss rate	Wall time (s)	s/game	Policy steps	ms/decision
VIPER tree	89	8.9%	3.715	0.00371	6,774	0.548
AIRD policy	13	1.3%	65.191	0.06519	8,935	7.296

4. Conclusion and limitations

AIRD demonstrates that composite distillation with an admissibility-aware intervention interface yields measurable formal verification improvement over direct tree deployment, without retraining the nominal policy. The exact induced-runtime-graph model checking used here is not merely a verification endpoint: it exposes safe horizons and attractor regions as audit signals for the policy object.

Unlike VIPER's CartPole and fixed-opponent Pong settings, Connect4 5×6 is adversarial and fully discrete, with over 1.5 million reachable states under direct tree deployment; its learner-induced frontier is sparse enough to make composite-policy separation computationally feasible and formally auditable. However, as stressed by Table 2, there is an extra cost: computational time. Preliminary results show a total wall-time ratio of approximately 17.5 between AIRD and VIPER. We believe that this ratio can be improved by using specific techniques (Knowledge compilation, SAT, BDD...) for the implementation of the admissibility monitor. Such improvement will then allow further progress in model checking and

policy certification.

Declaration on Generative AI

ChatGPT assisted with plotting code for Figure 1; Codex assisted with LaTeX formatting.

References

[1] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815*.

[2] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 627–635.

[3] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, 2499–2509.

[4] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically Interpretable Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 5045–5054.

[5] Andrew Silva, Taylor Killian, Ivan Jimenez, Sung-Hyun Son, and Matthew Gombolay. 2020. Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1855–1865.

[6] Christian Schilling, Anna Lukina, Emir Demirovic, and Kim Guldstrand Larsen. 2023. Safety Verification of Decision-Tree Policies in Continuous Time. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

[7] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. 1999. Model Checking. MIT Press.

[8] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676, 354–359. Nature Publishing Group UK London.

[9] Shantanu Thakoor, Surag Nair, and Megha Jhunjhunwala. 2016. Learning to play Othello without human knowledge. Stanford University, Final Project Report.

[10] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, number 1.

[11] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002. https://doi.org/10.1145/585265.585270, doi: 10.1145/585265.585270.