Hector: A Cognitive Architecture for Structural Deliberation via Request-Confirmation Networks

Anonymous submission

Abstract

Conventional reinforcement learning often yields high-performing but opaque agents that lack inspectable internal structure for deliberation. We present Hector, a cognitive architecture based on Request-Confirmation Networks (ReCoN), designed to study how hierarchical subgoals and planning horizons can emerge from self-organizing symbolic structures. Using chess endgames as a controlled symbolic microcosm, we show that a unified topology containing both KPK (king+pawn vs. king) and KQK (king+queen vs. king) subgraphs shifts control from promotion to checkmate with one-move latency, with the transition observable in internal activations. On a KPK curriculum, Hector achieves a 97.0% win rate, while PPO baselines reach 26.3% (50k timesteps) and 35.9% (200k timesteps). We also report exploratory structural maturation via stem cells and inertia pruning as an extension beyond fixed topologies.

We argue that autonomous strategic handover is a minimal operational requirement for deliberative agency: the ability to maintain, suspend, and reallocate control across competing internal models based on global context rather than local reward signals. While we make no claims about phenomenal consciousness, Hector provides a concrete, inspectable mechanism for global control, working memory, and top-down/bottom-up integration—properties central to multiple leading theories of consciousness.

Introduction: Beyond Reactive Agency

The Prodigy Problem

Modern deep reinforcement learning has produced agents capable of superhuman performance in games ranging from Go to StarCraft. Yet these prodigies exhibit a fundamental limitation: their competence is often a form of brute-force matching rather than structural decomposition. Because a neural network can achieve high reward by memorizing complex sensory-motor mappings, it is never forced to develop the deeper reasoning chains required for genuine deliberation.

We call this the **Prodigy Problem**: high-performance competence that lacks the internal structure necessary for inspection, modification, or maturation into complex planning. This aligns with recent calls for a "Third Wave" of AI that synthesizes neural learning with symbolic reasoning structures.

Chess as Drosophila Model

We utilize chess as a drosophila-model—a deterministic environment for quantifying long-range temporal credit assignment and subgoal inference. Our contribution is not "yet another chess engine" but a demonstration that:

- Strategic handover can be expressed as changes in internal activations rather than an external finite-state controller.
- Hierarchical subgoals can **emerge** from self-organizing symbolic structures.
- Learned structures remain **interpretable** throughout training.

Thesis

We propose that genuine deliberation requires a distributed orchestrator capable of functional decomposition. Hector, built on the ReCoN formalism, provides such an orchestrator by combining:

- Top-down goal delegation (requests flow from abstract goals to concrete sensors).
- Bottom-up confirmation (evidence flows from sensors to validate hypotheses).
- Temporal sequencing (POR/RET links enforce causal ordering).
- Structural plasticity (stem cells discover and solidify new patterns).

Claim preview. In this work, we demonstrate that a ReCoN-based architecture can function as a distributed deliberative orchestrator: autonomously discovering, coordinating, and handing over between hierarchical subgoals without a separate hand-engineered phase controller. Using chess endgames as a controlled symbolic microcosm, Hector exhibits structural maturation through self-organizing subgraphs while remaining fully interpretable throughout training. This establishes a concrete, inspectable alternative to black-box reinforcement learning for studying deliberation and long-horizon control.

Relevance to Machine Consciousness

This paper contributes to machine consciousness research not by asserting consciousness, but by grounding deliberative control in a testable architecture. In terms of the symposium themes, Hector provides:

- **Theory**: an operational, global-workspace-like control mechanism without a homunculus.
- **Implementation**: an engineered architecture in which global availability is explicit in the graph dynamics.
- **Measurement**: handover events are observable, timestamped internal state changes.
- **Ethics (light touch)**: transparency enables attribution and auditability debates.

The ReCoN Formalism: A Grammar of Deliberation

Request-Confirmation Networks (ReCoN) provide a neuro-symbolic framework for combining neural computation with hierarchical script execution. We summarize the key concepts and our extensions. For background on ReCoN and related cognitive frameworks, see Bach and Herger (ReCoN), Baars (global workspace), Cowan (working memory), and Schulman et al. (PPO) [1, 5, 6, 7].

Node Types

ReCoN networks consist of two fundamental node types:

Type	Definition	Role
SCRIPT	Hypothesis requiring validation	Intermediate goals, composite patterns
TERMINAL	Performs measurement/action	Sensors or actuators

Table 1: Core ReCoN node types.

Edge Types

Four directed edge types connect nodes, forming the grammar of deliberation:

Edge	Direction	Message	Purpose
SUB	Parent → Child	request	Request subgoal validation
SUR	Child → Parent	wait/confirm	Report progress/success
POR	Predecessor → Successor	inhibit_request	Enforce order
RET	Successor → Predecessor	inhibit_confirm	Prevent early confirmation

Table 2: ReCoN edge types and their roles.

State Machine and Message Passing

Each node implements an 8-state finite state machine. Transitions follow message passing rules (see Appendix A for the full transition table). The `inhibit_confirm` signal via RET links enables POR chains to function as sequences rather than parallel alternatives, preventing premature confirmation during multi-step plans.

Top-Down/Bottom-Up Integration

ReCoN integrates top-down prediction and bottom-up verification. Requests propagate from abstract goals to concrete sensors, and confirmations flow back to validate hypotheses. This bidirectional flow enables top-down control while keeping perception grounded in measurable affordances.

Hector's Roadmap: Developmental Scaffolding

Method Overview

Hector operates as a request-confirmation hierarchy updated in discrete ticks:

1. Goals issue top-down requests; terminals return confirmations.
2. Action scripts select moves from the most active subgraph.
3. Within-game fast plasticity updates edge weights using eligibility traces.
4. Cross-game consolidation updates base weights.
5. Optional M5 growth adds stem-cell sensors and pack structures (exploratory).

Hector's development proceeded through milestones M1–M5, each building capabilities for the next. The first stage executed heuristics in a fixed ReCoN network. Later stages introduced learned coordination and structural discovery.

Fixed-Topology Phase (M1–M4): Modular Coordination

M1–M2: continuous activations and trainable edge weights transform the ReCoN graph into a differentiable decision tree while preserving interpretability. **M3:** fast plasticity updates weights within a game using eligibility traces. **M4:** slow consolidation aggregates changes across games, stabilizing useful coordination patterns.

Key Result: Autonomous Deliberative Handover. A single topology containing multiple strategic regimes (KPK, KQK) shifts control from pawn promotion to checkmate with one-move latency, with the transition visible in internal activations. The handover is driven solely by activation dynamics.

```
t0: Pawn reaches 8th rank
t0+eps: KPK affordance -> 0
t0+eps: KQK affordance -> 1
t1: First KQK action selected
```

This makes the handover event inspectable and directly measurable in internal state, satisfying a key requirement for attribution and interpretability.

Exploratory Extension: Structural Growth Beyond Fixed Regimes

We emphasize that the empirical contribution of this paper concerns autonomous deliberative handover within a fixed topology. Structural growth via stem cells is presented as an exploratory extension, illustrating how deliberative depth may scale beyond pre-specified regimes. We provide an overview here and defer detailed lifecycle tables and growth logs to Appendix B.

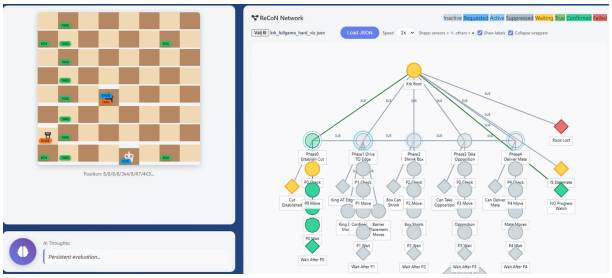

Figure 1: Global topology overview showing activations, edge weights, and bindings.

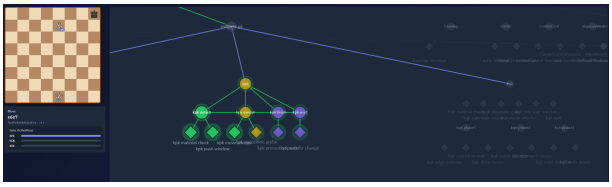

Figure 2: KPK leg dominance during pawn promotion.

Interpretability: Visualizing the Binding Mechanism

A key advantage of ReCoN over black-box approaches is inherent interpretability. Our visualization renders node activations (color intensity), edge weights (line thickness), state machine states (node colors), node types (shape), and sensor bindings on the chessboard.

This real-time visualization represents the agent's instantiated focus of attention within a broader working memory architecture. Structurally, Hector provides a concrete instantiation of Global Workspace-like dynamics, where specialized subgraphs compete for access to the motor system.

Activation Graph and Strategic Handover

During the KPK→KQK transition, the visualization reveals the precise moment the orchestrator reallocates requests based on environmental affordance spikes.

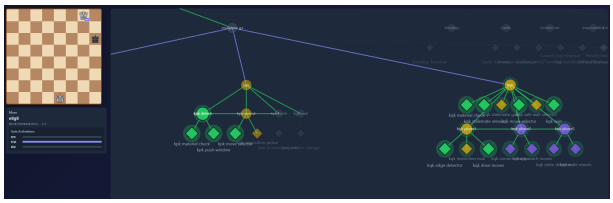

Figure 3: KQK leg activation immediately after promotion.

Experiments and Results

Experimental Setup

Environment: KPK endgame with a 64-dimensional observation vector (piece positions), legal moves as actions, and a random-move opponent. Rewards follow `scripts/kpk_gym_env.py`: +1.0 checkmate, +0.5 promotion (episode terminates on promotion), -1.0 loss, -0.5 draw/stalemate, and -0.2 timeout (max 100 plies).
Curriculum (ReCoN): an 8-stage curriculum from easy to hard positions.
Baseline (PPO): Stable-Baselines3 PPO trained on the Stage 7 distribution (not curriculum-trained), evaluated for 100 episodes per checkpoint.

Reproducibility Details

We report win rate (fraction of wins over games played in Stage 7), sample efficiency (episodes or environment steps required to reach target performance), and wall-clock training time (seconds). Unless stated otherwise, results are single runs and we do not report error bars.

PPO baseline: trained using `scripts/ppo_kpk_baseline.py` (Stable-Baselines3 PPO) with policy/value MLP 64–64. Hyperparameters: learning_rate=3e-4, n_steps=2048, batch_size=64, n_epochs=10, gamma=0.99, gae_lambda=0.95, clip_range=0.2, ent_coef=0.01. We report two training budgets (50k and 200k timesteps).

ReCoN (KPK): trained over an 8-stage curriculum with 10 cycles per stage and 100 games per cycle (8,000 games). Fast plasticity uses eta_tick=0.05 (within-game eligibility traces) and consolidation uses eta_consolidate=0.01 (cross-game updates). Stem cells and TRIAL promotion are enabled for exploratory growth statistics.

KRK curriculum (supporting/verification runs): `scripts/run_krk_curriculum.py` supports a full-engine mode and optional M5. A representative command used in larger runs was:

```
M5_HEURISTIC_PROB=0 LOG_KRK_POLICY=1
uv run python
scripts/run_krk_curriculum.py
--mode recon --enable-m5
  --games-per-cycle 500
--min-games-per-stage 500
--max-cycles-per-stage 10
  --output-dir
/tmp/krk_stage0_delta
```

Smaller runs (e.g., `--games-per-cycle 30--50`) were used during development for debugging and verification. We did not perform systematic hyperparameter sweeps; reported configurations reflect fixed settings selected during development.

Computing infrastructure: all experiments were run on a Lenovo ThinkPad (Intel Core Ultra 7 155U CPU, 32 GB RAM; CPU-only) on a 64-bit Windows host OS, using WSL2 (Linux 6.6.87.2-microsoft-standard-WSL2, glibc 2.39). Experiments were executed via `uv run python` (uv 0.8.13). Software: Python 3.12.3; numpy 2.4.x; python-chess 1.11.2. For PPO: Stable-Baselines3 2.7.1; Gymnasium 1.2.3; PyTorch 2.9.1+cu128 (GPU disabled); cloudpickle 3.1.2.

The full codebase will be released publicly upon publication, with scripts and configuration pinned to a specific commit. (For anonymous submission, we omit repository URLs and commit identifiers.)

Metric	PPO (50k)	PPO (200k)	ReCoN (Hector)
Stage 7 win rate	26.3%	35.9%	**97.0%**
Training budget	50k steps (˜2.9k games)	200k steps (˜11.7k games)	8,000 games
Wall-clock time	42s	194s	180s
Interpretable	No	No	**Yes**

Table 3: KPK baseline comparison (single runs; see Reproducibility Details).

Metric	Value
Total active nodes	152
Pack nodes (AND/OR gates)	45
Maximum depth	4
TRIAL→MATURE promotions	12
Inertia-pruned cells	58

Table 4: Exploratory structural growth statistics (representative run).

KPK Curriculum Results

Structural Growth (Exploratory)

Representative topology statistics from a KPK growth run (illustrative example):

KPK→KQK Handover

Using a pre-trained unified topology with both endgame subgraphs:

Metric	Value
Successful handovers	100%
Handover latency	1 move

Table 5: Autonomous handover results.

Discussion

Scope of Claims

We distinguish between (i) empirical results reported here (handover behavior, win rates, sample efficiency), (ii) architectural hypotheses (structural growth as a mechanism for deeper deliberation), and (iii) broader implications for machine consciousness, which we treat as speculative and non-committal.

What Is Hardcoded vs. Learned

Given only legal moves and win/loss rewards, Hector autonomously discovers hierarchical patterns corresponding to known chess theory (opposition, key squares, timing) without being programmed with those concepts. By mapping these patterns onto a transparent ReCoN topology, we move away from black-box heuristics toward structural deliberation that is inspectable and verifiable.

Structural Deliberation vs. Search-Driven Opportunism

Traditional engines such as Stockfish achieve superhuman performance through high-speed search. Hector instead relies on a distributed orchestrator to activate the most relevant strategic subgraph, offering transparent, hierarchical deliberation without brute-force look-ahead.

Relation to Theories of Conscious Access

We do not claim consciousness, but the architecture provides an inspectable testbed for theory-constrained falsification:

- **Global Workspace Theory:** global broadcast implemented via request propagation and shared activation.
- **Attention Schema Theory:** internal modeling of control state via explicit orchestration and handover events.
- **IIT:** not claimed, but causal structure is explicit and measurable.

Limitations

- Information gain in high-draw environments remains challenging without reward shaping.
- Structural pruning is still maturing; the current metabolic filter can be too aggressive or too permissive.
- Tactical precision in complex middle-game positions may require hybridization with search.

Conclusion: Toward Structured Control

We presented Hector, a cognitive architecture demonstrating that (1) hierarchical subgoals can emerge from self-organizing structures, (2) strategic phase transitions occur autonomously via activation dynamics rather than hard-coded orchestration, and (3) learned structures remain interpretable throughout training. ReCoN provides a general-purpose framework for structured control in domains requiring compositional reasoning, temporal sequencing, and top-down/bottom-up integration. By prioritizing structural decomposition over raw performance, Hector offers a path from game playing to autonomous deliberation in open-world tasks.

References

[1] Bach, J., and Herger, P. 2015. Request Confirmation Networks for Neuro-Symbolic Script Execution. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*.

[2] McCarthy, J. 1990. Chess as the Drosophila of AI. In *Finding a Solution of a Problem: A Festschrift for Robert K. Lindsay*.

[3] Garcez, A. S. d., and Lamb, L. C. 2020. Neurosymbolic AI: The 3rd Wave. *arXiv preprint arXiv:2012.05876*.

[4] Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*.

[5] Baars, B. J. 1988. *A Cognitive Theory of Consciousness*. Cambridge University Press.

[6] Cowan, N. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity and its adaptive value. *Behavioral and Brain Sciences*.

[7] Schulman, J., et al. 2017. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*.

[8] Kahneman, D. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Reproducibility Checklist

Instructions for Authors: This document outlines key aspects for assessing reproducibility. Please provide your input by editing this `.tex` file directly. For each question (that applies), replace the "Type your response here" text with your answer. **Example:** If a question appears as

```
\question{Proofs of all novel claims
are included} {(yes/partial/no)}
Type your response here
```

you would change it to:

```
\question{Proofs of all novel claims
are included} {(yes/partial/no)}
yes
```

Please make sure to:

- Replace ONLY the "Type your response here" text and nothing else.
- Use one of the options listed for that question (e.g., **yes**, **no**, **partial**, or **NA**).
- **Not** modify any other part of the `\question` command or any other lines in this document.

You can `\input` this .tex file right before `\end{document}` of your main file or compile it as a stand-alone document. Check the instructions on your conference's website to see if you will be asked to provide this checklist with your paper or separately.

---

1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) yes

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) yes

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) yes

2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) no
If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) no

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) no

2.4. Proofs of all novel claims are included (yes/partial/no) no

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) no

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) no

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) NA

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) NA

3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) no
If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) NA

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) NA

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) NA

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) NA

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) NA

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) NA

4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) yes
If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) partial

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) yes

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) partial

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) yes

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) partial

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) no

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) yes

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) yes

4.10. This paper states the number of algorithm runs used to compute each reported result (yes/no) yes

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) no

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) no

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) yes

ReCoN State Machine (Complete Table)

The Hector engine implements the standard ReCoN state machine with an 8-state transition logic. Transitions occur at discrete clock ticks based on top-down requests and bottom-up confirmations.

Current State	Condition	Next State
INACTIVE	Receives request from parent	REQUESTED
REQUESTED	Clock tick	WAITING
WAITING	All children confirmed or sensor true	TRUE
TRUE	Clock tick	CONFIRMED
WAITING	Sensor false or child failed	FAILED
ACTIVE	(Continuous settling)	WAITING
SUPPRESSED	POR predecessor not confirmed	INACTIVE

Table 6: Simplified ReCoN transition logic.

Code Appendix (Key Files)

- `scripts/kpk_gym_env.py` (KPK Gym environment)
- `scripts/ppo_kpk_baseline.py` (PPO baseline training)
- `scripts/run_kpk_bridge.sh` (bridge positions wrapper)
- `demos/persistent/full_game_train.py` (unified graph training/demo driver)
- `demos/visualization/export_bridge_demo.py` (export handover visualization traces)
- `scripts/run_krk_curriculum.py` (KRK curriculum driver; optional M5)
- `src/recon_lite_chess/graph/unified_builder.py` (unified KRK/KPK/KQK graph)

Structural Growth Details (Exploratory)

The maturation phase replaces hand-designed sensor nodes with self-discovering stem cells, enabling the ReCoN graph to autonomously grow its own topology. This process externalizes learned correlations into persistent, inspectable symbolic structures. M5 is currently exploratory.

Stem Cell Lifecycle

```
EXPLORING -> CANDIDATE -> TRIAL -> MATURE
                                -> DEMOTED (XP <= 0)
```

FeatureHub

Discovered tactical patterns are registered in a global FeatureHub, enabling transfer across strategic contexts (e.g., from KRK to KPK).

Inertia Pruning

Inertia pruning acts as a Bayesian filter for causal significance, removing nodes whose information gain does not justify their metabolic cost.

Pack Templates

Consistently coactive stem cells are hoisted into AND/OR gate packs that form higher-level compositional features.