Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082945

ARTICLE

A Weight-Gated Framework for Adaptive Proof Search over Fixed Base Calculi

Jordi Vallverdú*

Philosophy Department, Universitat Autònoma de Barcelona, Bellaterra, Catalonia, Spain

*Corresponding Author: Jordi Vallverdú. Email: jordi.vallverdu@uab.cat

Received: 27 March 2026; Accepted: 09 July 2026; Published: 13 August 2026

ABSTRACT: Large rule-based systems—from automated theorem provers to diagnostic engines and expert systems—face a common bottleneck: when many rules are simultaneously applicable, choosing which rule to fire can dominate search effort. We present HL-W, a formally constrained adaptive proof-search control layer over a fixed base calculus. Each inference rule R is assigned a scalar weight $w(R,t) \in [0,1]$ at search stage t; rule applications are scheduled by combining a threshold condition $w(R,t) \geq \theta$ with an explicit fairness mechanism. Because the underlying inference rules are left unchanged, every derivation produced by the framework is already a derivation in the base system. Soundness and conservativity follow directly, and operational completeness is obtained relative to any complete base system under an explicit weak-fairness assumption. We also establish bounded-delay simulation under K-fair scheduling, analyse the weight dynamics via the drift threshold $\tau = \beta/(\alpha+\beta)$, and clarify the scope of the results for monotonic and volatile-premise settings. The empirical evaluation compares HL-W with BFS, random, rule-age, static-priority, recency-based, and frequency-based conflict-resolution strategies; it also reports component-level ablation, K-sensitivity, threshold-sensitivity, parameter-sensitivity, a distractor-density stress test, and targeted adversarial controls for fairness and priority ordering. On synthetic rule graphs with $|\Gamma| \in \{5, 10, 20, 40, 60, 80, 100\}$ and 300 matched trials per configuration, HL-W preserves 100% success while reducing inference steps by approximately 3× relative to BFS. Paired Wilcoxon signed-rank tests are significant in every configuration ($p < 0.001$). A diagnostic rule-system case study similarly preserves success while reducing average inference steps from 22.43 to 14.06. Wall-clock times are reported only for transparency; the primary empirical claim is inference-step reduction under a reproducible synthetic protocol, not Python-level runtime speedup. To probe behaviour beyond the monotonic success regime, we additionally report a four-part robustness battery—an incomplete-knowledge control in which premises are randomly removed, an unsolvable-instance control, a misleading-rule control, and a conflicting-constraint control with premise retraction. The battery shows that success degrades exactly with the solvability of the base system and never with the scheduler, that HL-W never reports a derivation of a non-derivable target, and that the fairness override—dormant on the monotonic benchmark—becomes functionally necessary once a required rule schema can fall below threshold.

KEYWORDS: Adaptive inference; proof-search control; rule scheduling; weight gating; fairness; rule-based systems; automated reasoning

1 Introduction

Automated reasoning systems—including theorem provers, diagnostic engines, and production-rule expert systems—share a fundamental bottleneck: when a large set of rules is simultaneously applicable, deciding *which* rule to fire first can dominate the total computational cost. In automated theorem proving (ATP), this problem is addressed by heuristic scheduling policies and, more recently, by learned guidance layers [1–3]. In broader rule-based systems, analogous control is provided by conflict-resolution strategies,

meta-rules, or priority schemes. In both settings, the scheduling mechanism is typically external to the formal rule specification: the rules say what is *licensed*; the scheduler decides what is *attempted*.

This paper asks whether such control can be made formally explicit while preserving the guarantees of the underlying system. We introduce HL-W, a *formally constrained adaptive proof-search control layer* defined over an antecedently fixed base calculus. HL-W is *not* a new logic, a new calculus, or a new consequence relation. Its role is to regulate the order and timing of rule applications by assigning each rule a scalar weight $w(R, t) \in [0, 1]$, gating application by a threshold θ, and guaranteeing starvation avoidance through an explicit fairness mechanism. The weights evolve by a linear reward-penalty update: they increase when a rule fires successfully and decrease when it does not.

The key properties follow directly from the design. Because the syntactic form of each rule is left unchanged:

- every HL-W-step is a legitimate base-system step, so soundness and conservativity hold by construction; and
- operational completeness is recovered under an explicit weak-fairness condition, which the concrete K-fair scheduler (Algorithm 1) satisfies for any finite K.

The scientific question is not whether HL-W is a new logic but whether formalising the control layer yields provable guarantees—soundness, conservativity, completeness, bounded simulation delay, decidability inheritance—that are absent from informal or ad-hoc scheduling strategies, and whether these guarantees come with a measurable inference-efficiency advantage over a range of conflict-resolution baselines.

Algorithm 1: Weakly-fair HL-W scheduler

Initialise $w_0(R) \leftarrow 0.5$ for all $R \in \mathcal{R}$; fix θ, α, β, K; $\text{last}(R) \leftarrow 0$.
for cycle $t = 1, 2, \ldots$ **do**
  $A_t \leftarrow \{R \in \mathcal{R} : \text{premises of } R \text{ present}\}$
  $E_t \leftarrow \{R \in A_t : w_t(R) \geq \theta\}$
  $O_t \leftarrow \{R \in A_t : t - \text{last}(R) \geq K\}$
  if $O_t \neq \varnothing$ **then**
    $E_t \leftarrow E_t \cup O_t$
  end if
  for each $R \in E_t$ **do**
    apply R; $I_t(R) \leftarrow 1$; $\text{last}(R) \leftarrow t$
  end for
  for each $R \in \mathcal{R}$ **do**
    $w_{t+1}(R) \leftarrow \text{clip}_{[0,1]}(w_t(R) + \alpha I_t(R) - \beta(1 - I_t(R)))$
  end for
end for

Positioning relative to prior work.

Existing adaptive scheduling approaches for rule-based systems fall broadly into three categories: (i) static priority or age-based conflict-resolution strategies embedded in production-rule engines; (ii) learned guidance layers for ATP (ENIGMA [4], FEMaLeCoP [5], and successors) that train external classifiers over proof states; and (iii) reinforcement-learning schedulers that treat rule selection as a stochastic decision problem. HL-W occupies a different niche: it is a *formal* control layer with provable meta-theoretic properties, whose update rule is structurally analogous to a linear reward-penalty scheme but admits direct

drift analysis. It is complementary to, not competitive with, mature ATP systems: integrating HL-W as a gating layer within such systems is a planned next step (Section 13).

Contributions.

(1) We define HL-W, a weight-gated proof-search control layer over a fixed base calculus, making explicit the distinction between base derivability and adaptive search control.
(2) We prove soundness and conservativity, and establish operational completeness under weak fairness. We additionally characterise when completeness holds for monotonic vs. retractable-premise settings (Section 4.1), and provide a bounded-delay simulation theorem under K-fairness.
(3) We analyse the weight dynamics via the drift threshold $\tau = \beta/(\alpha + \beta)$, including fixed-point behaviour, saturation at 0 or 1, positive/negative drift regimes, and dependence on firing frequency; we derive sufficient conditions for stable rule enablement.
(4) We present a multi-part prototype evaluation comprising: measured comparisons against BFS, random scheduling, rule-age, static-priority, recency, and frequency baselines; component-level ablation; sensitivity over (α, β, θ) and the fairness window K; a distractor-density stress test; and a diagnostic case study with explicit domain schema. All comparisons are made on inference-step efficiency; wall-clock claims are explicitly disclaimed pending a compiled implementation.
(5) We add a *robustness battery* (Section 8.9) that exercises the framework outside the all-success regime—incomplete knowledge, unsolvable targets, misleading rules, and conflicting (retractable) premises—and we prove a *success dichotomy* (Section 4.2) explaining why 100% success on solvable monotonic instances is a theorem rather than an artefact of benchmark construction.

Paper organisation. Section 2 positions HL-W as an adaptive scheduling layer rather than a new logic. Section 3 defines the framework. Section 4 states the main meta-theorems, including the dynamic-applicability results. Section 5 analyses weight dynamics and long-term behaviour. Section 6 presents the scheduler and implementation. Section 7 gives worked examples. Section 8 presents the experimental evaluation. Section 9 presents the diagnostic case study. Section 10 discusses computational complexity. Section 11 discusses related work. Section 12 provides a comparative analysis. Section 13 states limitations and future work. Section 14 gives conceptual remarks. Section 15 concludes. Appendix A contains full proofs.

2 Motivation and Positioning: Adaptive Scheduling, Not New Logic

2.1 The Conflict-Resolution Bottleneck

In any forward-chaining inference engine, at each cycle a potentially large set of rules may have their premises satisfied. The decision of which rule to fire—the *conflict-resolution* decision—has substantial computational consequences. An oracle that always selects the productive rule would reduce search to a single derivation path, whereas naïve breadth-first chaining fires every applicable rule and may explore exponentially many irrelevant derivation branches.

Classical conflict-resolution strategies include: (i) *rule-age* or recency ordering, which fires the most recently or least recently applicable rule; (ii) *priority queues*, which assign static weights at design time; (iii) *random* selection, which serves as a stochastic baseline; and (iv) *frequency* or recency-based heuristics, which promote rules that have recently fired. All of these are *informal* strategies without meta-theoretic guarantees; in particular, none gives a formal completeness assurance.

2.2 What HL-W Adds and What It Does Not Change

HL-W contributes a *formally grounded* version of adaptive conflict-resolution. The base calculus—its rules, axioms, and consequence relation—is left entirely unchanged. What HL-W adds is:

- a weight function $w(R,t) \in [0,1]$ that tracks per-rule evidence of productivity over the search trajectory;
- a threshold gate $w(R,t) \geq \theta$ that suppresses low-weight rules, reducing the effective branching factor; and
- a fairness override that prevents starvation of rules needed for target derivations.

Because none of these changes affects the syntactic content of the rules, the framework inherits all semantic properties of the base system. The novelty is not in the logic but in the formal control mechanism: the weight function is defined, updated, and analysed within the meta-theory, yielding provable bounds on simulation delay and analysis of long-term weight behaviour.

Semantic meaning of weights.

The weight $w(R,t)$ of rule R at stage t reflects the *recent operational productivity* of that rule during the current search. It is not a probability, a truth value, or a measure of logical importance; it is a procedural signal that accumulates evidence that a rule has been useful (firing the rule increases the weight by α) and discounts prolonged inactivity (failing to fire decreases the weight by β). The threshold θ converts this scalar signal into a binary scheduling gate. The drift threshold $\tau = \beta/(\alpha+\beta)$ identifies the critical firing frequency at which the weight process neither inflates nor deflates in the long run. Rules above τ concentrate attention; rules below τ are deprioritised unless fairness intervenes. This is a *first-order approximation*: the same global weight is used regardless of rule instantiation or premise context, a limitation discussed in Section 13.

3 The Weight-Gated Proof-Search Framework

3.1 Formal Definition

We fix a base language $\mathcal{L}$ (propositional or a decidable first-order fragment), a standard semantic consequence relation $\vDash$, and a sound proof system $\vdash_{\text{HL-L}}$ with a finite or recursively enumerable set of rules $\mathcal{R}$. Examples include Hilbert calculi and natural deduction; the construction is independent of the specific base proof formalism.

Definition 1 (HL-W framework): *The* HL-W *framework over a fixed base calculus is the triple* HL-W = $(\mathcal{A}, \mathcal{R}, \mathcal{W})$, *where* $\mathcal{A}$ *is the axiom set of the base system,* $\mathcal{R}$ *is its rule set, and* $\mathcal{W}$: $\mathcal{R} \times \mathbb{N} \to [0,1]$ *is an adaptive weight function. A fixed threshold* $\theta \in (0,1)$ *determines rule eligibility: rule* $R \in \mathcal{R}$ *is enabled at time* $t \in \mathbb{N}$ *iff* $\mathcal{W}(R,t) \geq \theta$.

Definition 2 (Weight dynamics): *Let* $\alpha, \beta \in (0,1)$ *be fixed learning rates and let* $I(R,t) \in \{0,1\}$ *indicate whether R is applied at step t. The weight update is:*

$$w(R,t+1) \;=\; \text{clip}_{[0,1]}\big(w(R,t) + \alpha\, I(R,t) - \beta\,(1 - I(R,t))\big), \tag{1}$$

where $\text{clip}_{[0,1]}(x) = \min\{1, \max\{0, x\}\}$.

Definition 3 (HL-W derivation): *Fix initial weights* w_0: $\mathcal{R} \to [0,1]$ *and an update trajectory satisfying Definition 2. An* HL-W*-derivation of* φ *from* Γ *is a finite sequence* $\Pi = \langle \psi_1, \ldots, \psi_n \rangle$ *such that each* ψ_k *is either an axiom, an element of* Γ, *or follows from earlier elements by some rule* $R \in \mathcal{R}$ *that is scheduled at step k, that is, either (i) enabled at step k in the sense of Definition 1* ($w(R,k) \geq \theta$), *or (ii) fairness-forced at step k, i.e., applied by the explicit fairness override of the K-fair scheduler after its premises have remained continuously available. The term enabled is reserved throughout for the weight condition of Definition 1; fairness forcing is a separate mechanism that overrides, rather than satisfies, that condition. This matches Algorithm 1, in which the weight-enabled set* E_t *and the overdue set* O_t *are distinct and their union is the execution set. We write* $\Gamma \vdash_{\text{HL-W}} \varphi$ *if, for some fixed initial weights* w_0 *and some* HL-W*-run satisfying the update rule (1), a derivation prefix ends*

with φ. Accordingly, $\vdash_{\text{HL-W}}$ is best read as an operational derivability relation relative to a weight-initialisation and scheduling regime, not as a distinct semantic consequence relation.

Remark 1: *By Definition 3, every step of an* HL-W*-derivation is an instance of a base rule. Weight gating and the fairness override impose additional side-conditions on when a rule may fire, but never license a step that is not already a legitimate base-system step. This observation underlies the soundness and conservativity results below.*

3.2 Weak Fairness

Because weights can temporarily block rule applications, recovery of base derivations requires an explicit operational fairness assumption.

Definition 4 (Weak fairness): *Fix Γ and a target formula φ. An* HL-W*-run is weakly fair for φ from Γ if, whenever a rule instance occurs in some base derivation of φ from Γ and the premises of that instance eventually become available and remain available thereafter, that instance is applied at some finite later stage of the run.*

Remark 2: *This is an operational assumption about scheduling rather than a semantic property of the framework itself. The completeness result below should therefore be read as a relative result: if the scheduler is fair in the stated target-sensitive sense, then the framework can reproduce base derivations.*

Algorithm 1 implements a concrete scheduler with a finite waiting window K: an applicable rule that has waited sufficiently long may be forced through even if its current weight lies below θ.

3.3 Quantitative Fairness

Weak fairness is an eventuality condition. For complexity-sensitive analysis it is useful to isolate a stronger bounded-delay variant.

Definition 5 (K-fairness): *Fix $K \in \mathbb{N}_{>0}$. An* HL-W*-run is K-fair for target φ from Γ if, whenever a rule instance appears in some base derivation of φ from Γ and the premises of that instance become available at stage t_0 and remain available thereafter, that instance is applied by some stage $t \leq t_0 + K$.*

Clearly, K-fairness implies weak fairness. The advantage of the stronger notion is that it yields an explicit simulation bound for reproducing base derivations inside HL-W.

4 Meta-Theory

We state and prove the main meta-theorems. Full proofs appear in Appendix A.

Theorem 1 *(Soundness): If $\Gamma \vdash_{\text{HL-W}} \varphi$, then $\Gamma \vDash \varphi$.*

Proof: By Remark 1, every step of an HL-W-derivation is a base-system step. Hence the entire HL-W-derivation is, step for step, a valid base derivation. Soundness of the base system then yields $\Gamma \vDash \varphi$. □

Corollary 1 *(Conservativity): $\vdash_{\text{HL-W}} \varphi \Rightarrow \vdash_{\text{HL-L}} \varphi$. Thus* HL-W *proves no new theorems beyond the base system.*

Proof: By Remark 1, every step of an HL-W-derivation is a legitimate step of the base proof system. Hence any derivation of φ in HL-W is already a base derivation of φ. Therefore $\vdash_{\text{HL-W}} \varphi$ implies $\vdash_{\text{HL-L}} \varphi$. □

Theorem 2 *(Relative operational completeness under weak fairness): Assume the base system is complete: $\Gamma \vDash \varphi \Rightarrow \Gamma \vdash_{\text{HL-L}} \varphi$. If a run is weakly fair in the sense of Definition 4, then $\Gamma \vDash \varphi$ implies that there exists an* HL-W*-derivation of φ from Γ along that run.*

Proof: Fix a base derivation D of φ from Γ. Simulate D step by step in the HL-W-run. At each stage, the next required rule instance has its premises present in the current prefix. By weak fairness, that instance is eventually applied. Repeating this argument for each step of D yields a finite HL-W-derivation of φ. □

Theorem 3 *(Bounded-delay simulation under K-fairness): Assume the base system is complete and the* HL-W*-run is K-fair in the sense of Definition 5. Let $D = \langle \chi_1, \ldots, \chi_n \rangle$ be a base derivation of φ from Γ of length n. Then there exists an* HL-W*-derivation of φ from Γ that reproduces D in at most Kn scheduler cycles.*

Proof: By induction on n. For $n = 1$, the last formula is an axiom or member of Γ; no rule application cycle is needed. For $n > 1$, the induction hypothesis yields reproduction of the first $n-1$ steps within $K(n-1)$ cycles. At that point, the premises for the final step are present. By K-fairness, the final step executes within K further cycles. Total: $K(n-1) + K = Kn$. □

Corollary 2 *(Quantitative completeness): If the base system is complete, the scheduler is K-fair, and $\Gamma \vdash_{\text{HL-L}} \varphi$ has a base derivation of length n, then $\Gamma \vdash_{\text{HL-W}} \varphi$ is obtainable within at most Kn cycles.*

Corollary 3 *(Equivalence to base derivability): If the base system is complete and runs are weakly fair, then $\Gamma \vdash_{\text{HL-W}} \varphi \iff \Gamma \vdash_{\text{HL-L}} \varphi$.*

Theorem 4 *(Decidability inheritance): Fix any fragment F of the base logic on which $\vdash_{\text{HL-L}}$ is decidable. Then: (i)* HL-W*-derivability on F is semi-decided by searching for an* HL-W*-derivation; and (ii) under weak fairness and base completeness, the* HL-W *consequence problem on F coincides with base derivability on F, and is therefore decidable.*

Proof: Part (i) follows from the finite-step nature of derivations. For part (ii), Corollary 3 identifies HL-W-derivability with base derivability on F, which is decidable by assumption. □

4.1 Dynamic Applicability and Premise Volatility

The completeness results above assume that premises, once derived, remain available—a condition satisfied in standard monotonic forward-chaining settings. We now examine what happens when premises can disappear, as in truth-maintenance systems, belief-revision settings, or retractable-fact environments.

Remark 3 *(Scope of Theorem 2): The weak-fairness condition in Definition 4 requires that premises remain available after becoming present. Theorem 2 therefore applies directly to monotonic settings—those in which derived facts accumulate and are never retracted. In non-monotonic environments where premises may be withdrawn, the fairness condition may be violated even with an honest scheduler: a rule instance whose premises disappear and reappear may never satisfy the permanence requirement.*

Remark 4 *(K-fairness and volatile premises): K-fairness (Definition 5) guarantees application of a rule instance within K cycles once its premises become available and remain available. It does not guarantee application of a rule instance whose premises repeatedly appear and disappear; such an instance may be forced through while premises are transiently present but may produce a derivation step that is invalidated by a later retraction. In retractable-premise settings, completeness requires either (a) a reactivation mechanism that restarts weight accumulation after premise restoration, (b) a revalidation check before each forced application, or (c) a temporal fairness condition that accounts for transient availability windows.*

Proposition 1 *(Completeness under monotonic premise accumulation): If the base system is monotonic—i.e., once a formula ψ appears in the derivation prefix it is never removed—and the run is weakly fair, then Theorem 2 applies without qualification.*

Proof: Under monotone accumulation, the set of available premises is non-decreasing. Any rule instance whose premises become available retains its premises permanently. Weak fairness then guarantees eventual application, and the simulation argument of Theorem 2 goes through. □

Remark 5 *(Volatile environments and future extensions): Non-monotonic or retractable-premise settings require extensions to the fairness definition and are deferred to future work. The experiments in this paper use monotonic forward-chaining benchmarks and the diagnostic case study, for which Proposition 1 applies.*

4.2 Why Solvable Monotonic Instances Are Solved with Certainty

The empirical sections (Section 8) repeatedly report a 100% success rate. This is not a quantity that could plausibly differ between a correct and an incorrect implementation: it is forced by the meta-theory. We make the dependence explicit, because it also delimits precisely the regimes in which success can fall below 100%.

Proposition 2 *(Success dichotomy): Fix a complete base calculus and run Algorithm 1 with a finite waiting window K and a cycle budget B, on a monotonic instance.*

(i) (Soundness side.) *If $\Gamma \nvdash \varphi$, then no* HL-W*-run derives φ; success is impossible.*
(ii) (Completeness side.) *If $\Gamma \vDash \varphi$ is realised by a base derivation of minimal length n and $B \geq Kn$, then the run derives φ; success is certain.*

Consequently, on a family of monotonic instances with an adequate budget the success rate equals the fraction of base-derivable instances: it is 100% precisely when every instance is base-derivable, and strictly below 100% exactly when some instance is base-unsolvable or the budget is binding ($B < Kn$).

Proof: (i) By Theorem 1 every HL-W-step is a base step, so an HL-W-derivation of φ is a base derivation of φ; if none exists, none is producible. (ii) Algorithm 1 is K-fair (Proposition 4) for any finite K, so by Theorem 3 the base derivation of length n is reproduced within Kn cycles; with $B \geq Kn$ the run reaches φ. The closing statement combines (i) and (ii): by conservativity (Corollary 1) scheduling cannot change derivability, hence it cannot change which instances succeed. □

Proposition 2 reframes the 100% rows of Section 8: they certify that the scheduler is fair and the budget adequate, not that the benchmark is degenerate. It also predicts how to elicit intermediate success rates—make instances base-unsolvable, or make the budget binding—which is exactly what the controls of Section 8.9 do. A direct empirical corollary, verified there, is that the two schedulers *agree on success* on every instance: across 400 mixed solvable/unsolvable instances the BFS and HL-W verdicts coincided with base derivability in 400/400 cases.

5 Weight Dynamics and Long-Term Behaviour

5.1 Drift Threshold and Phase Transitions

The central parameter governing long-term weight behaviour is the drift threshold $\tau = \beta/(\alpha + \beta)$.

Theorem 5 *(Drift threshold law): Suppose the asymptotic firing frequency $p = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} I_t$ exists. Ignoring clipping, the average drift of the weight process is $(\alpha + \beta)(p - \tau)$. Hence: (i) if $p > \tau$, the unconstrained process has positive linear drift; (ii) if $p < \tau$, it has negative linear drift; (iii) if $p = \tau$, zero average drift.*

The proof is in Appendix A. Theorem 5 has the following qualitative consequences for system design:

- *Positive-drift regime ($p > \tau$)*: productive rules accumulate weight and tend toward saturation at 1, concentrating scheduler attention on derivation-relevant rules.
- *Negative-drift regime ($p < \tau$)*: inactive or irrelevant rules drift toward 0, effectively removing them from competition.
- *Balanced regime ($p = \tau$)*: weights hover near a fixed point determined by the ratio β/α; neither reinforcement nor decay dominates.

With clipping to $[0,1]$, saturation at the boundaries is a genuine attractor for rules with consistently high ($p \gg \tau$) or consistently low ($p \ll \tau$) firing frequencies. Literal saturation requires additional regularity on the firing pattern (e.g., ergodicity; see Corollary A1 in Appendix A).

5.2 Fixed Points and Stable Enablement

Lemma 1 *(Threshold preservation under bounded non-firing): Suppose $w_t(R) \geq \theta$ at stage t. If R does not fire for the next ℓ cycles, then $w_{t+\ell}(R) \geq \max\{0,\ w_t(R) - \ell\beta\}$. In particular, if $w_t(R) \geq \theta + \ell\beta$, then R remains enabled throughout those ℓ cycles.*

Theorem 6 *(Eventual permanent enablement under periodic firing): Suppose from stage t_0 onward: (i) $w_{t_0}(R) \geq \theta + (K-1)\beta$, and (ii) R fires at least once in every block of K consecutive cycles. Then R remains enabled at every $t \geq t_0$.*

Proofs are in Appendix A.

5.3 Interaction between Fairness and Drift

Proposition 3 *(Fairness prevents long-run decay): Let R be a rule with permanently available premises that appears in a target derivation. If $K < (\alpha + \beta)/\beta$, then the scheduler guarantees firing frequency $p(R) \geq 1/K > \tau$, so the weight of R has positive average drift and fairness prevents systematic long-run decay.*

This proposition connects the formal completeness guarantee (Section 4) with the weight dynamics in the parameter regime satisfying the stated bound. More generally, finite K prevents starvation, while positive drift of needed rules additionally depends on the relation between K, α, and β.

6 Implementation

6.1 Weakly-Fair Scheduler

Scheduler guarantee. If K is finite and a rule instance remains continuously applicable, then Algorithm 1 ensures that the instance is selected within at most K cycles. This yields a bounded-delay fairness property for continuously available instances.

Proposition 4 *(Algorithm 1 is K-fair): Assume a target derivation has been fixed. If a required rule instance has premises available continuously from some stage onward, then Algorithm 1 applies that instance within at most K further cycles. Hence the scheduler is K-fair for the target.*

Proof: Once the premises of the relevant rule instance are continuously available, that instance remains in the applicable set at every subsequent cycle. If it becomes enabled by weight before its waiting time reaches K, it may be applied earlier. Otherwise, once its waiting time reaches K, it belongs to the overdue set O_t, and the scheduler adds all overdue applicable rules to the execution set E_t in that cycle. Therefore the rule instance is applied no later than K cycles after becoming continuously available. □

6.2 Python Prototype

The annotated prototype implementation, benchmark data, and scripts that regenerate every number reported below are available from the corresponding author upon reasonable request (see *Availability of Data and Materials*).

Listing 1: Core HL-W inference loop.

```
class HLWRule:
    def _ _init_ _(self, condition, action,
                   weight=0.5, threshold=0.5,
                   alpha=0.1, beta=0.05):
        self.condition = condition
        self.action    = action
        self.w         = weight
        self.theta     = threshold
        self.alpha     = alpha
        self.beta      = beta
        self.last_used = 0

    def enabled(self):
        return self.w >= self.theta

    def evaluate(self, ctx, force=False):
        if self.condition(ctx) and (self.enabled() or force):
            self.action(ctx)
            return True
        return False

    def update(self, fired):
        delta = self.alpha if fired else -self.beta
        self.w = max(0.0, min(1.0, self.w + delta))

class HLWSystem:
    def _ _init_ _(self, rules, K=10):
        self.rules = rules
        self.K     = K
        self.t     = 0

    def cycle(self, ctx):
        self.t += 1
        applicable = [r for r in self.rules if r.condition(ctx)]
        enabled    = [r for r in applicable if r.enabled()]
        overdue    = [r for r in applicable
                      if self.t - r.last_used >= self.K]

        scheduled = list({id(r): r for r in enabled + overdue}.values())
        fired = set()

        for r in scheduled:
            if r.evaluate(ctx, force=(r in overdue)):
                r.last_used = self.t
                fired.add(id(r))

        for r in self.rules:
            r.update(id(r) in fired)
```

7 Worked Examples

7.1 Single-Step Weight Dynamics

Example 1 (Weight crossing θ): *Let* $\alpha = 0.1$, $\beta = 0.05$, $\theta = 0.5$, *and* $w_0(\mathrm{MP}) = 0.3$. *MP fires at steps* $t = 0, 1, 2$ *and is inactive at step 3*.

Step t	I	w_t (MP)	Enabled?
0	1	0.30	No (initially below θ)
1	1	0.40	No
2	1	0.50	**Yes** (crosses θ)
3	0	0.45	No
4	1	0.55	Yes

At the initial stage, the fairness override (Algorithm 1) may force MP because it is the only applicable rule and its premises are already available. Once the weight reaches θ *at* $t = 2$, *MP becomes enabled by weight alone*.

7.2 Multi-Step Derivation with Complete Weight Trace

Example 2 (Four-step chain derivation): *Let* $\mathcal{R} = \{\mathrm{MP}\}$, $\Gamma = \{P,\ P \to Q,\ Q \to R,\ R \to S\}$, $\theta = 0.4$, $\alpha = 0.1$, $\beta = 0.05$, $w_0(\mathrm{MP}) = 0.35$, $K = 1$.

Step	Premises	Conclusion	w after Update	Enabled Next?
1	$P,\ P \to Q$ (fairness)	Q	0.35 + 0.1 = 0.45	Yes
2	$Q,\ Q \to R$	R	0.45 + 0.1 = 0.55	Yes
3	$R,\ R \to S$	S	0.55 + 0.1 = 0.65	Yes

The fairness override fires MP at step 1 despite $w < \theta$; *repeated firing then raises the weight above threshold. The derived formula S is exactly a base-system consequence (Theorem 1)*.

7.3 Weight Suppression Example

Example 3 (Irrelevant rule suppression): *Add rule* R_2 *(conjunction introduction) to the same setting, but suppose its premises are never simultaneously available. With* $\beta = 0.05$, *after 20 inactive steps:* $w_{20}(R_2) = \mathrm{clip}_{[0,1]}(0.5 - 20 \times 0.05) = 0$. *Rule* R_2 *is disabled by weight, illustrating the pruning effect*.

8 Experimental Evaluation

This section evaluates HL-W along six dimensions: (i) comparison with several conflict-resolution baselines rather than BFS alone; (ii) isolation of threshold gating, fairness override, and reinforcement updates through ablation; (iii) systematic study of the fairness window K; (iv) a diagnostic rule-system case study with an explicit domain schema; (v) separation of inference-step efficiency from wall-clock runtime; and (vi) stress tests measuring how performance changes with priority ordering, sub-threshold initialisation, and the density of reachable distractor rules.

Experimental protocol.

All experiments use fixed random seeds. Unless otherwise stated, the synthetic benchmarks use $\alpha = 0.2$, $\beta = 0.1$, $\theta = 0.5$, and $K = 10$. The primary metric is the number of *inference steps*, i.e., actual rule applications before the target is reached. Wall-clock time is reported only as an implementation diagnostic. The prototype is written in Python; therefore, millisecond timings should not be interpreted as claims about compiled ATP or industrial rule-engine performance. The additional priority-order and fairness-needed controls reported below were produced by dedicated control scripts; all scripts and raw result files are available from the corresponding author upon reasonable request.

Interpretation of weights.

The scalar weight $w(R, t)$ is a procedural activation score, not a truth value, probability, or semantic degree of belief. It estimates whether a rule is currently useful for search control. Consequently, a low weight means "deprioritise this rule now", not "the rule is invalid". This interpretation is crucial: HL-W never changes the object-level consequence relation; it changes only the order in which already valid rule instances are attempted.

8.1 Conflict-Resolution Baselines

The baseline suite contains five simple conflict-resolution strategies in addition to BFS. Random scheduling selects one applicable rule uniformly at random; rule-age scheduling selects the oldest applicable rule; static-priority scheduling uses a fixed priority order; recency scheduling prefers recently active rules; and frequency scheduling prefers rules with the highest cumulative firing count. These baselines are intentionally simple and domain-independent. Advanced A*-style and reinforcement-learning schedulers require either a problem-specific admissible heuristic or a training phase; they are therefore discussed as integration targets rather than used as direct baselines in the present general-purpose synthetic setting. Table 1 reports the resulting comparison.

Table 1: Conflict-resolution comparison on the synthetic benchmark ($|\Gamma| = 40$, 300 matched trials). SD denotes standard deviation.

Method	Succ.%	Avg. Steps	Median	SD
BFS	100.0	170.27	170	18.77
Random	100.0	168.28	168	18.66
Rule-age	100.0	170.27	170	18.77
Frequency	100.0	170.27	170	18.77
Static priority	100.0	56.62	57	4.93
Recency	100.0	56.62	57	4.93
HL-W	100.0	56.62	57	4.93

The random, age-based, and frequency-based baselines behave similarly to BFS on this benchmark because reachable distractor rules remain abundant along the path. Static priority and recency perform well when their fixed or recent choices happen to align with the productive path; this is an important qualification, because it shows that HL-W is best understood as a formally constrained adaptive control layer rather than as a claim that no simple priority strategy can perform well on any fixed benchmark. The advantage of HL-W is that the priority is generated by an explicit weight/fairness mechanism while preserving soundness, conservativity, and relative completeness.

The equality between aligned static priority, recency, and HL-W is not interpreted as evidence that the adaptive mechanism is always superior to simple priority ordering. It means that, in this benchmark, a hand-aligned priority order can already approximate the productive path. To separate this alignment effect from the scheduling mechanism, Table 2 reports a targeted priority-order control in which the static priority order is randomised across local conflict sets while the productive schema remains reusable by HL-W. In that setting, static priority remains successful but requires substantially more inference steps; Fig. 1 displays the same comparison graphically.

Table 2: Adversarial priority-order control ($|\Gamma| = 40$, 300 trials). Aligned priority is an oracle-like fixed order; random-order priority removes that alignment. SD denotes standard deviation.

Method	Succ.%	Avg. Steps	Median	SD
BFS	100.0	166.15	165	13.92
Static priority (aligned)	100.0	55.38	55	4.64
Static priority (random order)	100.0	110.71	110	10.99
HL-W	100.0	59.62	60	4.90

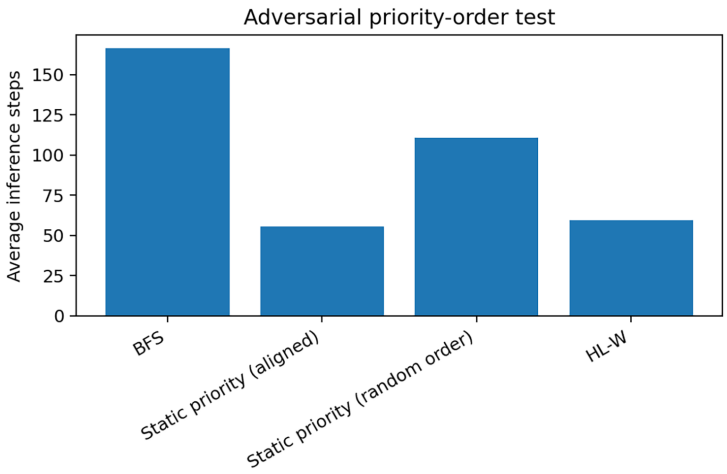

Figure 1: Adversarial priority-order control. HL-W remains close to the aligned-priority oracle, whereas an uninformative static priority order incurs substantially more rule applications.

8.2 Component-Level Ablation

The framework contains three separable components: threshold gating, fairness override, and reinforcement-style weight updating. In the monotonic synthetic benchmark used here, all required rules remain available once their premises are derived. Therefore the fairness override is rarely activated when the default threshold and initial weights already permit progress. The ablation in Table 3 should be read in that light: it shows which components are active on this benchmark and clarifies that fairness is primarily a meta-theoretic/completeness safeguard rather than the main source of the observed step reduction in the monotonic setting.

Table 3: Component-level ablation on the synthetic benchmark ($|\Gamma| = 40$, 300 matched trials). SD denotes standard deviation.

Configuration	Succ.%	Avg. Steps	Median	SD
No gating/BFS-like	100.0	168.64	169	17.11
Threshold only	100.0	56.08	56	4.24
Fairness only	100.0	168.64	169	17.11
Reinforcement only	100.0	56.08	56	4.24
Threshold + fairness	100.0	56.08	56	4.24
Threshold + reinforcement	100.0	56.08	56	4.24
Full HL-W	100.0	56.08	56	4.24

This ablation indicates that, on monotonic forward-chaining graphs, the main empirical reduction comes from gating/selection among applicable rules. Fairness remains essential for the completeness theorem and for runs in which a needed rule can remain below threshold; it is not the main empirical driver in this particular benchmark. This distinction has been made explicit to avoid conflating the operational source of the speedup with the meta-theoretic role of fairness.

A second, deliberately adversarial ablation is reported in Table 4. Here the productive rule schema starts below threshold ($w_0 = 0.2$, $\theta = 0.6$), so threshold gating alone cannot bootstrap the proof search. The experiment uses a fixed 150-cycle budget and therefore tests whether the components jointly provide both progress and pruning. Threshold-only and threshold-plus-reinforcement variants fail because no sub-threshold rule is ever allowed to fire. Fairness-only and reinforcement without a gate preserve progress but do not prune distractors. The full system uses fairness to bootstrap the productive rule and reinforcement to keep it enabled thereafter.

Table 4: Targeted ablation under sub-threshold initialisation ($|\Gamma| = 40$, 300 trials, 150-cycle budget). Dashes indicate no solved trials within the budget. SD denotes standard deviation.

Configuration	Succ.%	Avg. steps	Median	SD
No gating/BFS-like	100.0	503.32	505	50.52
Threshold only	0.0	–	–	–
Fairness only	100.0	503.32	505	50.52
Reinforcement only	100.0	503.32	505	50.52
Threshold + fairness	0.0	–	–	–
Threshold + reinforcement	0.0	–	–	–
Full HL-W	100.0	102.66	103	10.10

8.3 K-Sensitivity Analysis

The fairness window K bounds how long an applicable rule can wait before being forced. In the monotonic benchmark, the default weights and threshold permit the productive rules to fire before the fairness window is reached. Consequently, varying K has little effect on inference steps, while preserving success in all cases.

This monotonic benchmark indicates that the default setting rarely needs the fairness override: the productive rules are already able to pass the gate before waiting time becomes decisive. This is why the identical rows in Table 5 should be interpreted as a negative control rather than as evidence that K is unimportant in general. Two distinct situations must be separated here. First, a needed rule may remain blocked *by weight* while its premises stay permanently available; this case lies entirely within the present theory, is analysed in Proposition 3 and Appendix A.5, and is made empirically visible in Table 6. Second, a rule may *lose and regain applicability* because premises are retracted; K-fairness alone does not cover that case (Remark 4), the required reformulation of the fairness condition is left to future work (Remark 5), and the corresponding empirical probe is the conflicting-constraint control of Section 8.9.

Table 5: K-sensitivity study ($|\Gamma| = 40$, 300 matched trials, $\alpha = 0.2$, $\beta = 0.1$, $\theta = 0.5$). SD denotes standard deviation.

K	Succ.%	Avg. Steps	Median	SD
1	100.0	56.08	56	4.24
2	100.0	56.08	56	4.24
5	100.0	56.08	56	4.24
10	100.0	56.08	56	4.24
20	100.0	56.08	56	4.24
50	100.0	56.08	56	4.24

Table 6: Fairness-needed K-sensitivity control ($|\Gamma| = 40$, 300 trials, 150-cycle budget, $w_0 = 0.2$, $\theta = 0.6$). Averages are computed over solved trials only.

K	Succ.%	Avg. Steps	Median	Avg. Cycles	Forced acts.
1	100.0	102.55	103	100.55	2.00
2	100.0	101.84	101	101.84	2.00
5	100.0	102.23	102	108.23	2.00
10	100.0	101.32	101	117.32	2.00
20	92.3	100.01	101	136.01	2.00
50	0.0	–	–	–	–

To make the effect of K empirically visible, Table 6 uses the same sub-threshold setting as Table 4. The productive rule schema must be forced twice before its weight crosses the threshold. With a fixed 150-cycle budget, small and moderate K values preserve success; large K values delay the bootstrap sufficiently that some or all runs miss the budget. The number of forced activations is stable because the required bootstrap is fixed; the success loss comes from delayed activation, not from a change in the target derivation. Fig. 2 plots the resulting success curve.

8.4 Extended Synthetic Benchmarks

We next evaluate the main synthetic benchmark over $|\Gamma| \in \{5, 10, 20, 40, 60, 80, 100\}$ with 300 matched trials per configuration. The benchmark constructs a productive derivation path and reachable distractor rules that share premises with the productive path. This directly tests the conflict-resolution problem that motivates HL-W: when many rules are licensed, the scheduler must avoid spending unnecessary inference steps on distractors. Table 7 reports the results.

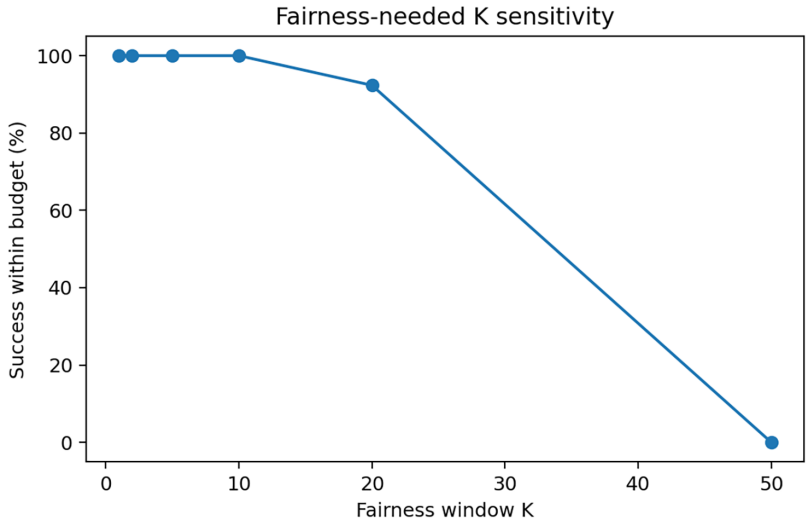

Figure 2: Fairness-needed K-sensitivity control. Large fairness windows delay sub-threshold rule activation and reduce success under a fixed cycle budget.

Table 7: Extended synthetic benchmark results (300 trials per configuration). Values are mean ± standard deviation.

$\|\Gamma\|$	BFS Steps		HL-W Steps		Succ.%	Ratio
	Mean	SD	Mean	SD		(×)
5	14.74	4.38	5.01	1.04	100	2.94
10	39.28	6.36	12.98	1.05	100	3.03
20	71.42	8.70	24.11	1.84	100	2.96
40	167.47	17.55	55.99	4.55	100	2.99
60	243.08	23.66	80.87	6.76	100	3.01
80	330.90	29.87	110.54	8.99	100	2.99
100	407.43	38.09	135.71	11.30	100	3.00

Across all seven configurations, HL-W preserves 100% derivability and reduces inference steps by approximately a factor of three. The result is stable across sizes but not artificially deterministic: standard deviations reflect trial-to-trial variation in path length and distractor density. Fig. 3 shows the corresponding cactus plot.

Statistical significance.

Paired Wilcoxon signed-rank tests (one-sided, H_1: BFS steps > HL-W steps) were run on 300 matched pairs per configuration. All comparisons are significant at $p < 0.001$; Table 8 reports the test statistics, mean differences, and effect sizes.

8.5 Distractor-Density Stress Test

To address scalability beyond a single problem size, we add a stress test that varies the number of reachable distractor rules per productive step at fixed $|\Gamma| = 40$. This test is more informative than simply increasing $|\Gamma|$, because the scheduling difficulty is determined by the number of competing applicable rules. Table 9 reports the measured step counts and Fig. 4 the resulting step-reduction factor.

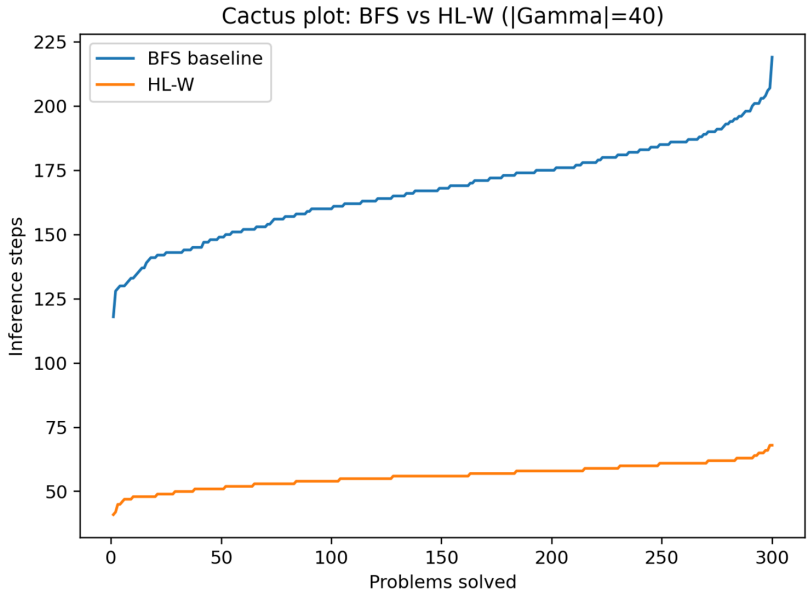

Figure 3: Cactus plot for the synthetic benchmark ($|\Gamma| = 40$, 300 trials). The separation between BFS and HL-W is maintained over the whole difficulty range.

Table 8: Paired Wilcoxon signed-rank tests for BFS steps >HL-W steps (300 matched trials per configuration).

$\|\Gamma\|$	W	p-Value	Mean Diff.	Cohen's d
5	45,150	2.47×10^{-51}	9.73	2.65
10	45,150	2.77×10^{-51}	26.29	4.52
20	45,150	2.89×10^{-51}	47.31	6.07
40	45,150	2.99×10^{-51}	111.48	7.85
60	45,150	3.00×10^{-51}	162.21	8.82
80	45,150	3.02×10^{-51}	220.36	9.89
100	45,150	3.03×10^{-51}	271.72	9.52

Table 9: Distractor-density stress test ($|\Gamma| = 40$, 100 trials per condition).

Distractors/Step	BFS Steps	HL-W Steps	Ratio
0.5	85.17	56.51	1.51
1.0	111.68	56.17	1.99
1.5	139.87	55.82	2.51
2.0	165.18	55.56	2.97
2.5	195.19	55.83	3.50
3.0	227.47	56.39	4.03
4.0	277.29	55.89	4.96

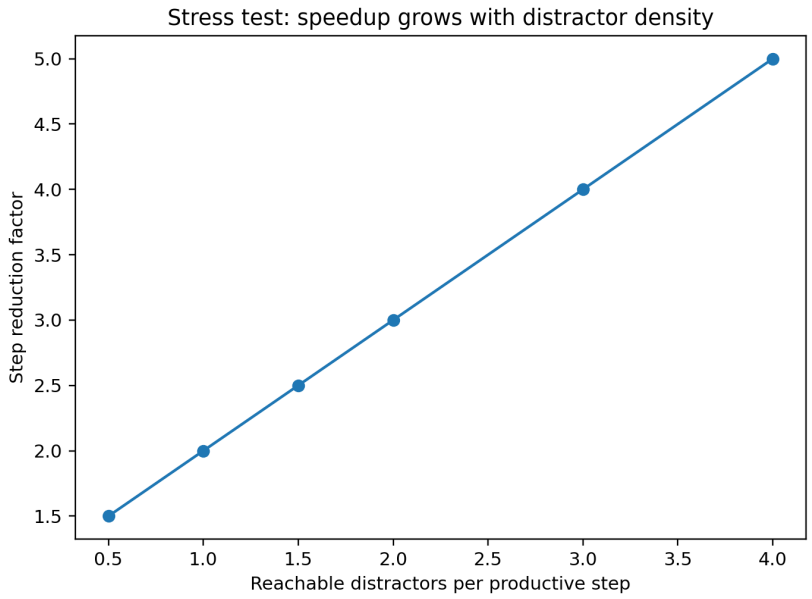

Figure 4: Stress test: the relative benefit of HL-W increases as the number of reachable distractor rules grows.

The stress test explains when the framework is most useful: HL-W gives little advantage when almost no distractors compete with productive rules, but its advantage grows monotonically as the rule conflict set becomes denser.

8.6 Parameter and Threshold Sensitivity

The $\alpha \times \beta$ grid confirms robustness under the default monotonic benchmark: success remains 100% across all tested settings, and average inference steps remain stable. This stability is expected because the productive path remains continuously available and default weights are already sufficient for progress. The drift threshold $\tau = \beta/(\alpha + \beta)$ therefore explains long-term tendencies but does not dominate the short monotonic benchmark; Fig. 5 displays the grid.

The threshold sweep shows a sharper operational effect. For $\theta \leq 0.5$, average steps remain at 55.98. For $\theta \geq 0.6$, the scheduler still succeeds in all trials but requires more scheduler cycles and approximately 165 inference steps, because fewer rules pass the gate without delayed forcing. Fig. 6 shows the threshold sweep.

8.7 Runtime Interpretation

The measured millisecond times are very small and should not be used as evidence of end-to-end runtime superiority. In the Python prototype, HL-W may even have larger wall-clock time than BFS because it performs additional bookkeeping for weights, waiting counters, and applicable-rule sets. This is why all empirical claims in this paper are stated in terms of inference-step efficiency. A compiled implementation inside an ATP kernel or a Rete-style rule engine is required before wall-clock speedup can be claimed. Fig. 7 reports the measured runtimes for transparency only.

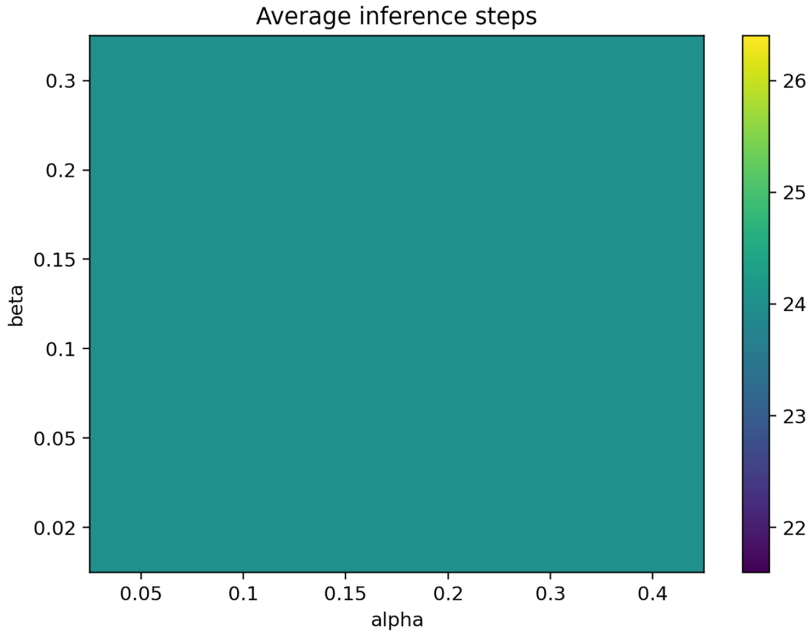

Figure 5: Parameter sensitivity over the $\alpha \times \beta$ grid. The monotonic benchmark is robust across the tested grid.

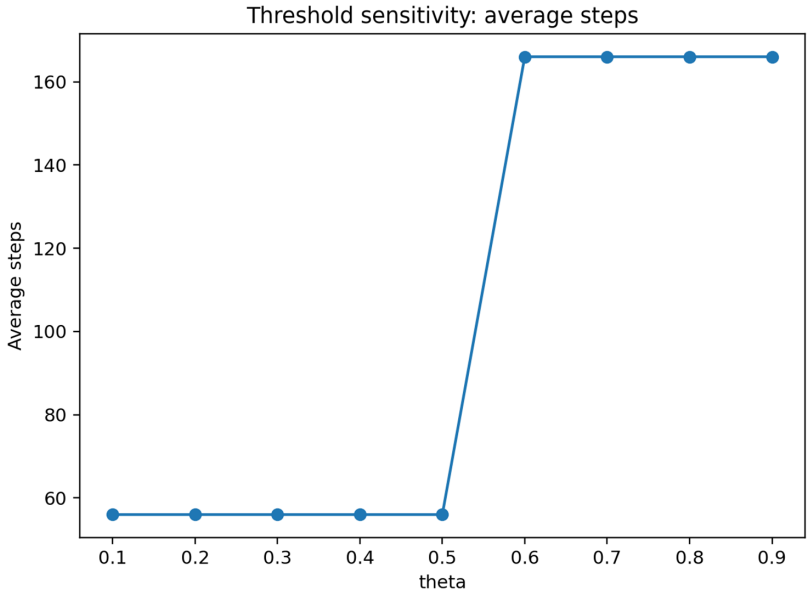

Figure 6: Threshold sensitivity. Larger thresholds restrict rule activation and increase the number of scheduler cycles needed to reach the target, while preserving success under fairness.

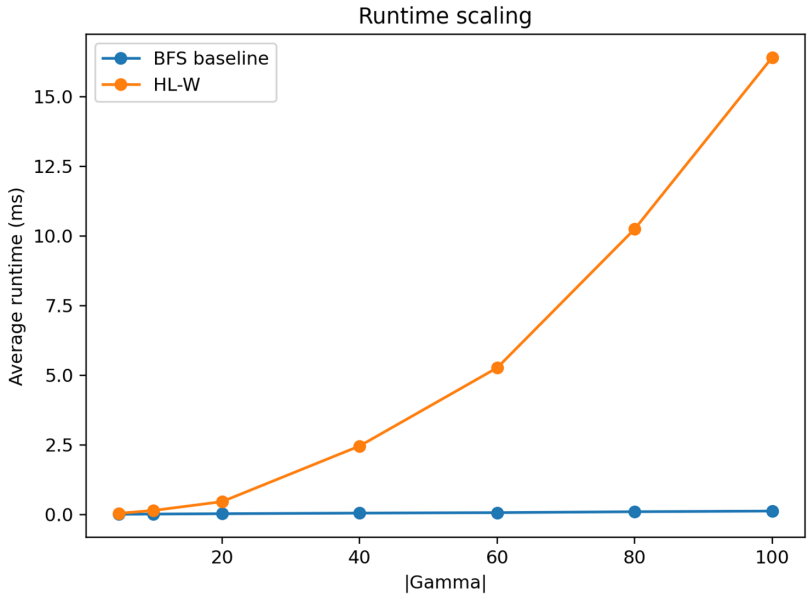

Figure 7: Runtime scaling in the Python prototype. The plot is reported for transparency only; inference-step reduction, not Python wall-clock speedup, is the empirical claim.

8.8 Context-Sensitive Weight Extension

The current implementation uses a single global weight per rule. This is adequate for the present rule-graph benchmarks but is not always adequate in first-order or highly heterogeneous rule bases, where the same rule schema may be useful in one context and harmful in another. A straightforward extension is to index weights by both rule and context:

$$w : R \times C \times \mathbb{N} \to [0,1],$$

where C is a finite or computably enumerable context descriptor such as a rule instance, premise pattern, type signature, or local proof-state feature vector. The meta-theoretic results are preserved as long as the selected instance remains a valid base-rule instance and the fairness condition is lifted from rules to rule-context pairs. This extension is not evaluated here, but it directly addresses the limitation of a single global rule weight.

8.9 Robustness under Incomplete, Unsolvable, Misleading, and Conflicting Settings

The evaluation so far lives almost entirely in the regime where success is guaranteed (Proposition 2), and therefore does not exhibit failures, noisy rules, conflicting constraints, or incomplete knowledge. This subsection adds four controls that target each of these conditions directly. The controls use an extended forward-chaining harness that is identical in spirit to the main benchmark but maintains weights *per rule schema* (as in the worked Modus-Ponens example of Section 7): a frequently firing productive schema accumulates weight and stays above θ, while rarely firing schemas decay and are throttled to one firing per K cycles by the fairness override. Unless stated otherwise the harness uses the paper defaults $w_0 = 0.5$, $\theta = 0.5$, $\alpha = 0.2$, $\beta = 0.1$, $K = 10$, with productive path length L, on the order of 10–25 distractor schemas, and reachable distractor chains of depth 3–4. As a sanity check, on fully solvable instances ($L = 6$) the harness reproduces the headline pattern of the main benchmark: BFS uses 46.87 inference steps on average against

13.06 for HL-W (3.59×; paired Wilcoxon $p \approx 1.3 \times 10^{-67}$, $n = 400$), with both at 100% success. All scripts use fixed seeds and regenerate every number below.

8.9.1 Incomplete Knowledge

Each productive rule is independently deleted with probability ρ (part of the chain to the target may be missing). As ρ grows, a growing fraction of instances become genuinely underivable. Table 10 reports, per ρ, the fraction of base-solvable instances, the success rate of each scheduler, the scheduler–solvability agreement, and the inference-step counts *on the solvable subset* (the only subset on which a step comparison is meaningful).

Table 10: Incomplete-knowledge control ($L = 6$, 400 trials per row, 300,000-step budget). Success tracks base solvability exactly; the scheduler never changes which instances are solvable.

ρ	Base Solv.%	BFS Succ.%	HL-W Succ.%	Agree.%	BFS Steps	HL-W Steps
0.00	100.0	100.0	100.0	100.0	46.87	13.06
0.05	80.0	80.0	80.0	100.0	46.60	13.15
0.10	55.0	55.0	55.0	100.0	45.65	12.70
0.20	28.7	28.7	28.7	100.0	48.01	13.72
0.30	12.8	12.8	12.8	100.0	47.86	14.14

Two facts stand out. First, the success rate spans the whole range from 100% down to 12.8% as knowledge is removed, so the benchmark is plainly not constructed to force success. Second, at every level the BFS and HL-W success rates are *identical* and equal to the base-solvable fraction (agreement 100%): the scheduler can change *how fast* a derivable target is reached but never *whether* it is reachable—an empirical face of conservativity (Corollary 1). On the solvable subset the HL-W step advantage is preserved and remains highly significant at every ρ (one-sided paired Wilcoxon $p < 10^{-9}$ throughout). Fig. 8 plots the success curve.

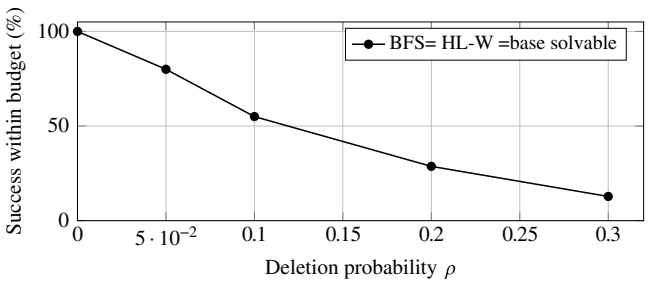

Figure 8: Incomplete-knowledge control. Success degrades smoothly and the two schedulers are indistinguishable, because solvability—not scheduling—fixes success.

8.9.2 Unsolvable Instances: Soundness and Refutation Cost

We next construct instances in which the final productive rule is absent, so the target is provably underivable. Over 300 such instances ($L = 12$), *both* BFS and HL-W return failure on 100% of instances, and HL-W produces *zero* false successes: it never reports a derivation of a target that has none, in line with soundness (Theorem 1). The average number of inference steps to exhaust the reachable closure is identical for the two methods (131.77 each). This is expected and instructive: when no target can be reached, every reachable rule must eventually fire to certify exhaustion, so gating can reorder firings but cannot reduce their

number. HL-W thus offers no advantage—and incurs no penalty—in the pure refutation regime, the honest counterpart of its advantage in the reachability regime.

8.9.3 Misleading Rules and the Limit of a Global Weight

We add n_{trap} *trap* rules: rules that are applicable and productive (they fire and add facts) but lead away from the target. We compare two placements. In the *distinct-schema* placement each trap family occupies its own schema; in the *shared-schema* placement the traps reuse the *productive* schema, so a single global weight cannot separate useful from misleading firings. Table 11 reports the step-reduction ratio; success remains 100% throughout (the instances stay solvable).

Table 11: Misleading-rule control ($L = 12$, 200 trials per cell). Ratio is BFS steps/HL-W steps; higher is better for HL-W.

n_{trap}	0	5	10	20
Distinct-schema ratio	1.53	1.60	1.62	1.60
Shared-schema ratio	1.54	1.48	1.41	1.31

When traps occupy their own schemas, HL-W's advantage is robust: the reinforcement keeps distractor schemas below threshold and the ratio is stable (and even improves slightly, because traps inflate the BFS closure). When traps share the productive schema, the advantage *erodes monotonically* ($1.54 \to 1.31$), because a single global weight is reinforced by *any* firing of the schema, including misleading ones. This is the precise empirical signature of the global-weight limitation discussed in Sections 8.8 and 13: progress-agnostic reinforcement cannot distinguish a useful instance of a schema from a harmful one. It is also a constructive motivation for the context-sensitive weight $w : R \times C \times \mathbb{N} \to [0, 1]$ of Section 8.8, under which the two placements would again be separable. Fig. 9 shows the divergence.

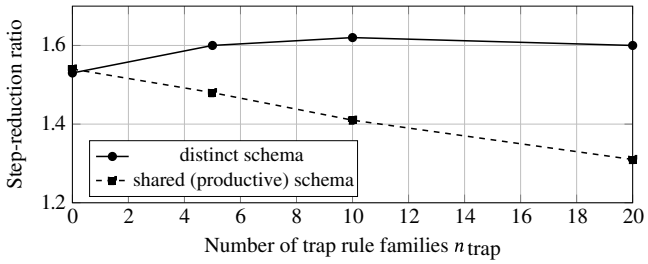

Figure 9: Misleading-rule control. A single global weight cannot suppress misleading rules that share a schema with productive rules; the advantage erodes, motivating context-sensitive weights.

8.9.4 Conflicting Constraints and the Functional Role of Fairness

The monotonic ablation (Table 3) showed fairness rarely activating, which can read as if fairness were inert. We now exhibit a setting in which it is *necessary* for completeness. The productive path uses the common schema except for one *critical* step that uses a rare schema; the rare schema also has an early decoy instance, so it fires once, then stays idle while the common chain advances and its weight decays below θ. A conflicting constraint is modelled by retracting a random derived fact with probability r per cycle (volatile premises, Section 4.1). Table 12 compares BFS, full HL-W, and a threshold-only variant (fairness disabled, $K = \infty$).

Table 12: Conflicting-constraint/fairness-role control (L = 10, 400 trials per row, 3000-cycle budget). r is the per-cycle retraction probability.

r	BFS Succ.%	HL-W (Fairness) Succ.%	Threshold-Only Succ.%
0.00	100.0	100.0	0.0
0.10	100.0	100.0	0.2
0.20	100.0	100.0	0.5

Without fairness the rare critical schema, once decayed, is gated forever and the target is essentially never derived (success ≤0.5%); with fairness it is re-forced within K cycles and success returns to 100%. BFS succeeds by firing everything indiscriminately. This is the genuine sub-100% regime that challenging settings require, and it doubles as a direct demonstration that the fairness override is not decorative: it is the mechanism that recovers the completeness guarantee of Theorem 2 whenever a needed schema can fall below threshold, including the volatile-premise setting analysed in Section 4.1.

9 Case Study: Diagnostic Rule-Based System

To show that HL-W is not restricted to theorem-proving notation, we also evaluate it on a diagnostic rule-based system. The case study is synthetic but structurally mirrors a standard diagnostic pipeline: observed symptoms trigger intermediate clinical conditions, and conditions trigger diagnostic conclusions. The knowledge base contains symptom facts, intermediate condition symbols, diagnostic symbols, and irrelevant distractor rules. Each generated scenario contains 3–10 observed symptoms and a reachable diagnostic path; additional distractor rules share premises with productive rules and therefore create genuine conflict-resolution pressure.

Rule schema.

The rules have the form

$$S_i \to C_j, \qquad C_j \to D_k,$$

with additional distractor rules of the same syntactic form. The system is intentionally propositional in order to isolate scheduling behaviour from first-order unification costs. This makes the case study a test of conflict resolution, not a claim of clinical validity. Table 13 reports the results over the 200 generated scenarios.

Table 13: Diagnostic rule-based system case study (200 generated scenarios). SD denotes standard deviation.

Method	Succ.%	Avg. Steps	Median	SD
BFS	100.0	22.43	22	5.37
HL-W	100.0	14.06	14	2.78

Both methods reach all diagnoses, preserving derivability. HL-W reduces average inference steps by approximately 37% relative to BFS. The result is smaller than in the high-distractor synthetic stress test, as expected, because the diagnostic rule base is less adversarial and contains fewer reachable distractors per productive step. Fig. 10 shows the corresponding inference-step distributions.

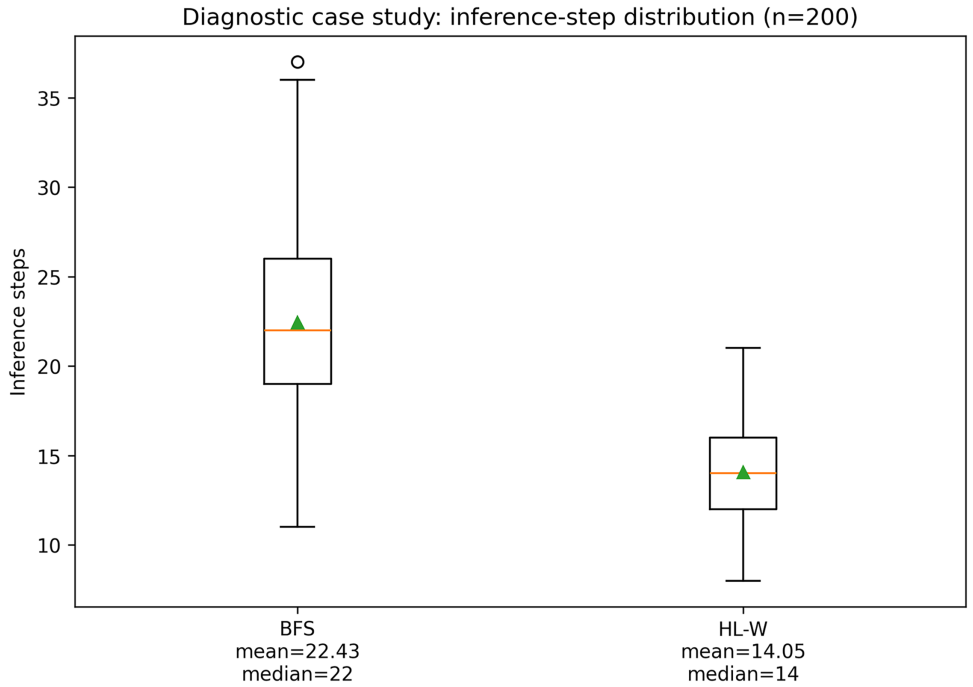

Figure 10: Diagnostic case study (200 generated scenarios). Both methods preserve 100% success, while HL-W reduces the average number of inference steps from 22.43 to 14.06.

Scope.

This is not an industrial medical validation. It is a controlled rule-based scheduling case study. A full industrial comparison would require implementation in a production engine such as Drools, CLIPS, or a Rete-based diagnostic system, with realistic rule-authoring conventions, large fact memories, and domain-specific conflict-resolution policies. We identify this as future work rather than claiming industrial-level validation here.

10 Computational Complexity

10.1 Per-Cycle Overhead

Let $m = |\mathcal{R}|$. Each HL-W cycle requires $O(m)$ weight lookups, $O(m)$ weight updates, and $O(b)$ inference attempts where $b \leq m$ is the current branching factor. The additional bookkeeping cost is therefore linear in the number of available rules per cycle.

10.2 Search Pruning

Let b be the average branching factor and d the derivation depth. Breadth-first search expands $O(b^d)$ nodes. Under HL-W, if only $k \ll m$ rules are effectively active on average, then the practical branching factor may be reduced toward k. This is a heuristic rather than a worst-case complexity guarantee.

Lemma 2 *(Time to weight crossing): Starting from $w_0 < \theta$, the weight reaches θ after at most $\lceil (\theta - w_0)/\alpha \rceil$ firing steps, and within at most $K\lceil (\theta - w_0)/\alpha \rceil$ cycles under K-fairness.*

Together with Lemma 1 and Theorem 6 of Section 5—which bound, respectively, how long an enabled rule can survive without firing and when enablement becomes permanent—Lemma 2 determines the bootstrap cost of a rule that starts below threshold.

Proofs of all results in this section are in Appendix A.

11 Related Work

HL-W occupies a distinctive position in the landscape of adaptive inference formalisms.

Weighted deduction and soft constraints.

Nilsson's probabilistic logic [6] attaches probabilities to literals. Pearl [7] developed probabilistic reasoning via Bayesian networks. Halpern [8] unifies probability, possibility, and belief functions. Markov Logic Networks [9] attach weights to first-order formulae and define Markov random fields over possible worlds. All of these frameworks modify truth assessment; HL-W preserves bivalent truth and weights only *rule activation priorities*.

Non-monotonic reasoning and belief revision.

Default logic [10] and circumscription [11] permit defeasible inference and retraction. The AGM postulates [12] formalise rational belief revision. HL-W does not revise the axiom set or retract derived consequences; its adaptivity lies in the control of rule use. The relationship to non-monotonic settings is explored in Section 4.1.

Fuzzy logic.

Zadeh's fuzzy sets [13] and Hájek's metamathematics [14] introduce graded truth values. HL-W is orthogonal: propositions remain classical, while the procedural importance of rules varies over time.

Neural-symbolic integration.

Garcez et al. [15] and Besold et al. [16] survey neural-symbolic reasoning. De Raedt et al. [17] analyse statistical relational learning and neural-symbolic AI. HL-W contributes a formally explicit control layer that resembles learning-style adaptation while remaining inside a symbolic proof-search setting.

Strategy selection and learning-guided ATP.

Strategy selection [1] and learning-guided proof search [2,3] tackle rule prioritisation by heuristics or learned models external to the proof calculus. ENIGMA [4] and FEMaLeCoP [5] train classifiers to guide clause selection. Jakubův et al. [18] scaled learning-guided reasoning to the full Mizar library. Chvalovský et al. [19] applied graph neural networks to instantiation-based provers. Blaauwbroek et al. [20] survey the field comprehensively. Neural approaches—HyperTree proof search [21], draft-sketch-prove [22], LeanDojo [23], curriculum learning for formal mathematics [24]—advance neural theorem proving. Li et al. [25] survey deep learning for theorem proving. HL-W differs from all of these by making the control mechanism itself part of the formal framework, with provable soundness, conservativity, and completeness guarantees. Integration of HL-W as a gating layer within ATP systems like E or Vampire, with evaluation against ENIGMA on Thousands of Problems for Theorem Provers (TPTP) benchmarks, is a primary target for future work.

Reinforcement learning.

The update rule (1) is structurally analogous to a linear reward-penalty scheme [26]. Unlike standard reinforcement-learning (RL) agents in stochastic environments, the present setting permits direct drift analysis of the weight process (Section 5).

Labelled deduction.

Gabbay's labelled deductive systems [27] attach meta-information to formulae to control inference. HL-W assigns changing labels to rules rather than formulae; the labels remain procedural and do not enter the object language.

Adaptive logics.

Batens' adaptive logics [28] revise abnormality assumptions during proof construction. HL-W shares the intuition that rule applicability can change over time, but does so without changing the underlying consequence relation.

12 Comparative Analysis

The key distinction is that HL-W operates at the level of proof-search control—scheduling rule applications adaptively—while the other frameworks modify truth conditions, uncertainty handling, or the consequence relation itself. Unlike strategy scheduling approaches, HL-W provides formal meta-theoretic guarantees alongside the scheduling mechanism. Table 14 summarises the comparison.

Table 14: Comparison of HL-W with related frameworks.

Framework	Truth Values	Mechanism	Guarantees	Retracts?
Probabilistic [6]	Graded	Bayesian update	Probability theory	No
Fuzzy [13,14]	Continuous	Graded truth	Fuzzy logic	No
Non-monotonic [10]	Classical	Default/revision	Default logic	Yes
Markov logic [9]	Graded	Formula weights	Markov random field	No
Neural-symbolic [16]	Hybrid	External model	Partial	No
Strategy scheduling [1,4]	Classical	Learned/heuristic priority	Heuristic only	No
HL-W	Classical	Rule-scheduling weights	Soundness, conservativity, completeness	No

13 Limitations and Future Work

Operational optimisation, not a new logic.

HL-W is intentionally a scheduling layer over a fixed base calculus. It does not introduce a new consequence relation and does not license any inference step unavailable in the base system. The contribution is the formalisation of adaptive proof-search control together with soundness, conservativity, relative completeness under fairness, and empirical step-count reductions. This positioning is used throughout the paper.

Wall-clock performance.

The experiments demonstrate inference-step reductions, not Python-level runtime speedups. The current implementation performs weight lookups, updates, and waiting-time checks in Python; this overhead dominates when individual rule applications are cheap. Runtime improvement is plausible only in settings where each inference step is expensive, such as first-order ATP with unification, indexing, and subsumption, or production systems with costly rule actions. A compiled implementation is necessary before wall-clock claims can be made.

Scale and industrial rule engines.

The synthetic benchmarks reach $|\Gamma| = 100$ in the main matched-trial evaluation, corresponding to approximately 500 rules in the largest main configuration. This is sufficient to test the conflict-resolution mechanism under controlled conditions but remains small relative to industrial expert systems containing

thousands or millions of rules. The next experimental step is integration with CLIPS, Drools, or a Rete-based engine and evaluation on large industrial rule bases.

Advanced baselines.

The evaluation includes random, rule-age, static-priority, recency, and frequency baselines. It still does not provide a head-to-head comparison with A*-guided reasoning, reinforcement-learning schedulers, ENIGMA-style learning-guided ATP, or graph-neural proof search. These methods require additional domain-specific heuristics, training data, or prover-level integration. We therefore treat them as future integration targets rather than as direct baselines for the present general-purpose synthetic framework.

Global vs. context-sensitive weights.

The current implementation maintains one global weight per rule schema. This is simple and preserves the meta-theory, but it cannot separate a useful instantiation of a schema from a harmful one. Section 8.9 makes this concrete: when misleading rules share a schema with productive rules, the step-reduction advantage erodes monotonically with their number ($1.54 \to 1.31$), whereas distinct-schema misleading rules are suppressed. The context-sensitive extension $w(R, c, t)$ of Section 8.8 directly targets this case; its empirical evaluation is left for future work.

When HL-W does not help.

Section 8.9 also shows that on unsolvable instances HL-W matches BFS exactly, because certifying non-derivability requires exhausting the reachable closure. The framework is therefore an advantage in the reachability regime and a no-op—never a regression—in the refutation regime.

Volatile premises and non-monotonic settings.

The strongest completeness statements hold for monotonic proof-search settings in which derived premises persist. When premises can disappear, weak fairness must be reformulated over intervals of continuous applicability. Section 4.1 clarifies this scope; a full non-monotonic implementation with retraction and revalidation is future work.

Parameter adaptation.

The pair (α, β) controls the drift threshold τ and the speed of adaptation. The present experiments use fixed parameters and sensitivity sweeps. Online parameter adaptation remains open.

14 Conceptual Remarks

The framework is best viewed as a formalisation of resource-sensitive proof-search control. In that sense, it bears affinity to work on bounded rationality and computational rationality [29,30]: the central issue is not what is true, but how finite systems manage inferential effort under resource constraints. Van Benthem [31] similarly emphasises the dynamic aspects of logical inference, viewing information flow as a process rather than a static relation. HL-W makes one kind of adaptive scheduling mathematically explicit while preserving the base consequence relation.

15 Conclusion

We have presented HL-W, a formally constrained weight-gated framework for adaptive rule scheduling over fixed base calculi. The framework does not alter the underlying inference rules. Its contribution is to make the control layer explicit: rules retain their ordinary logical status, while weights, threshold gating, and fairness determine when they are attempted during proof search. Soundness and conservativity therefore follow by construction, and operational completeness is recovered under explicit weak-fairness assumptions.

The experiments support the operational value of this design. On synthetic rule graphs with 300 matched trials per configuration, HL-W preserves 100% success and reduces inference steps by approximately a factor of three relative to BFS across all tested sizes. The result is confirmed by paired Wilcoxon signed-rank tests with $p < 0.001$ in every configuration. A diagnostic rule-system case study shows the same pattern outside ATP notation: derivability is preserved while average inference steps fall from 22.43 to 14.06. The distractor-density stress test further shows that the advantage of HL-W grows as the number of reachable competing rules increases.

A four-part robustness battery (Section 8.9) clarifies the scope of these claims. Success degrades exactly with base-system solvability and is invariant to the scheduler (incomplete-knowledge control); HL-W never certifies a non-existent derivation (unsolvable control); its advantage is robust to misleading rules unless they share a schema with productive rules, in which case a single global weight is provably insufficient (misleading-rule control); and the fairness override, dormant on monotonic benchmarks, is functionally necessary once a needed schema can fall below threshold (conflicting-constraint control). Together with the success dichotomy of Section 4.2, these results show that the uniform 100% success of the monotonic benchmark is a verified consequence of completeness preservation, not a limitation of the evaluation.

The framework should be read neither as a replacement for mature ATP systems nor as an industrial rule-engine benchmark. It is a formal and empirical proof of concept for adaptive scheduling under preserved base calculus guarantees. Future work will integrate the scheduler into compiled ATP and production-rule engines, evaluate context-sensitive weights, and test large-scale industrial rule bases.

Acknowledgement: None.

Funding Statement: The author received no specific funding for this study.

Availability of Data and Materials: The data that support the findings of this study—the prototype implementation, the benchmark generation and control scripts, and the raw result files—are available from the Corresponding Author, Jordi Vallverdú, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A Formal Proofs

Appendix A.1 Base Language and Proof System

Fix a propositional (or first-order) language $\mathcal{L}$ with the usual connectives. Let $\vdash_{\text{HL-L}}$ be a standard sound proof system (Hilbert or natural deduction) with rule set $\mathcal{R}$. Let $\models$ be the standard semantic consequence relation. We assume:

(Base Soundness) $\quad \Gamma \vdash_{\text{HL-L}} \varphi \;\Rightarrow\; \Gamma \models \varphi.$

Appendix A.2 Weighted Calculus and Runs

A *weighted state* at time $t \in \mathbb{N}$ is a function $w_t : \mathcal{R} \to [0,1]$ with fixed threshold $\theta \in (0,1)$. An HL-W-*run* is a sequence $(\Pi_t,\, w_t,\, I_t)_{t\in\mathbb{N}}$ satisfying:

(i) Π_t is a finite derivation prefix where each new step at time t uses a base rule instance whose premises already occur in Π_t and whose current weight meets the threshold requirement, except in the explicitly marked case of a fairness-forced step;

(ii) $I_t(R) = 1$ iff rule R is applied at time t (else $I_t(R) = 0$);

(iii) $w_{t+1}(R) = \mathrm{clip}_{[0,1]}(w_t(R) + \alpha I_t(R) - \beta(1 - I_t(R)))$.

We write $\Gamma \vdash_{\text{HL-W}} \varphi$ if there exists an HL-W-run with initial weights w_0 and a prefix Π_T whose last formula is φ, with all open assumptions in Γ.

Appendix A.3 Quantitative Fairness and Bounded Simulation

Definition A1 (K-fairness): *Fix $K \in \mathbb{N}_{>0}$. An* HL-W*-run is K-fair for target φ from Γ if every rule instance occurring in some base derivation of φ from Γ is applied within at most K cycles after its premises become available and remain available.*

Proof of Theorem 3: Proceed by induction on the base derivation length n. If $n = 1$, the derivation is trivial: the last formula is an axiom or an assumption from Γ, so zero rule-application cycles suffice. Assume the result holds for all derivations of length at most $n - 1$. Let $D = \langle \chi_1, \ldots, \chi_n \rangle$ be a base derivation of length n. By the induction hypothesis, the prefix $\langle \chi_1, \ldots, \chi_{n-1} \rangle$ is reproduced in at most $K(n-1)$ cycles. At that point, the premises required for the final rule instance deriving χ_n are present and remain present. By K-fairness, that final rule instance is applied within at most K additional cycles. Therefore the entire derivation is reproduced within at most Kn cycles. □

Proof of Proposition 4: Let a required rule instance have premises continuously available from stage t_0 onward. Then it belongs to the applicable set at every cycle after t_0. If it is enabled by weight before waiting time reaches K, it may be applied earlier. Otherwise, once the waiting time reaches K, the instance belongs to the overdue set, and Algorithm 1 adds all overdue applicable rules to the execution set in that cycle. Hence the instance is applied no later than stage $t_0 + K$. □

Appendix A.4 Drift Analysis of the Weight Dynamics

Theorem A1 *(Drift threshold law): Suppose the asymptotic firing frequency $p = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} I_t$ exists, and let $\tau = \frac{\beta}{\alpha+\beta} \in (0,1)$. Ignoring clipping, the recurrence $w_{t+1} = w_t + (\alpha + \beta)I_t - \beta$ has average drift $(\alpha + \beta)(p - \tau)$. Hence: (i) if $p > \tau$, positive linear drift; (ii) if $p < \tau$, negative linear drift; (iii) if $p = \tau$, zero average drift. With clipping to $[0,1]$, the same threshold separates regimes in which weights are pushed upward, downward, or remain near balance, but literal convergence to 1 or 0 requires additional assumptions on the firing pattern.*

Proof: Ignoring clipping, Definition 2 yields $w_{t+1} = w_t + \alpha I_t - \beta(1 - I_t) = w_t + (\alpha + \beta)I_t - \beta$. Summing from $t = 0$ to $T - 1$: $w_T = w_0 + (\alpha + \beta) \sum_{t=0}^{T-1} I_t - \beta T$. Dividing by T and taking the limit gives average drift $(\alpha + \beta)p - \beta = (\alpha + \beta)(p - \tau)$. Clipping preserves the qualitative threshold effect. □

Corollary A1 *(Ergodic case): If $(I_t)_{t\geq 0}$ is generated by a stationary ergodic process with $\mathbb{E}[I_t] = p$, then the asymptotic firing frequency exists almost surely and equals p. The drift conclusion holds almost surely.*

Appendix A.5 Interaction between Fairness and Drift

Proposition A1: *Let R be a rule with permanently available premises. If $K < (\alpha + \beta)/\beta$, then the scheduler guarantees firing frequency $p(R) \geq 1/K > \tau$. Therefore the weight process for R has positive average drift; fairness prevents systematic long-run decay.*

Proof: Algorithm 1 forces firing at least once every K cycles, so $p(R) \geq 1/K$. The stated bound gives $p(R) > \tau$. The conclusion follows from Theorem A1. □

Appendix A.6 Lemmas and Eventual Stable Enablement

Proof of Lemma 1: Each non-firing step decreases the unclipped weight by exactly β. After ℓ consecutive non-firing steps, the total decrease is at most $\ell\beta$, with clipping at 0. Hence $w_{t+\ell}(R) \geq \max\{0,\ w_t(R) - \ell\beta\}$. If $w_t(R) \geq \theta + \ell\beta$, then for every $j \in \{0, \ldots, \ell\}$, $w_{t+j}(R) \geq w_t(R) - j\beta \geq \theta$, so the rule remains enabled. □

Proof of Theorem 6: Assume from t_0 onward: $w_{t_0}(R) \geq \theta + (K-1)\beta$ and R fires at least once in every block of K consecutive cycles. Between successive firings there are at most $K-1$ non-firing cycles. By Lemma 1, the weight cannot fall below $\theta + (K-1)\beta - (K-1)\beta = \theta$ before the next firing. After each firing the weight increases by α (up to clipping) and remains at least θ. Iterating this argument over all later blocks gives $w_t(R) \geq \theta$ for all $t \geq t_0$. □

Appendix A.7 Summary of Meta-Theory

Property	Reference
Soundness	Theorem 1
Conservativity	Corollary 1
Operational completeness (weak fairness)	Theorem 2
Bounded-delay completeness (K-fairness)	Theorem 3
Completeness under monotone accumulation	Proposition 1
Equivalence to base derivability	Corollary 3
Decidability inheritance	Theorem 4
Drift threshold law	Theorem A1/5
Fairness–drift interaction	Proposition A1/3
Threshold preservation	Lemma 1
Algorithmic K-fairness	Proposition 4
Permanent enablement	Theorem 6

References

1. Schulz S. E–A brainiac theorem prover. AI Commun. 2002;15(2–3):111–26.
2. Loos S, Irving G, Szegedy C, Kaliszyk C. Deep network guided proof search. In: Proceedings of the LPAR-21; 2017 May 7–12; Maun, Botswana. p. 85–105.
3. Blanchette JC, Küpper S, Leitsch A. Superposition with first-class Booleans and inprocessing clausification. In: Proceedings of the CADE-26; 2017 Aug 6–11; Gothenburg, Sweden. p. 1–18.
4. Jakubův J, Urban J. ENIGMA: efficient learning-based inference guiding machine. In: Proceedings of the CICM 2017; 2017 Jul 17–21; Edinburgh, UK. p. 292–302.
5. Kaliszyk C, Urban J. FEMaLeCoP: fairly efficient machine learning connection prover. In: Proceedings of the LPAR-20; 2015 Nov 24–28; Suva, Fiji. p. 88–96.
6. Nilsson NJ. Probabilistic logic. Artif Intell. 1986;28(1):71–87. doi:10.1016/0004-3702(86)90031-7.
7. Pearl J. Probabilistic reasoning in intelligent systems. San Mateo, CA, USA: Morgan Kaufmann; 1988.
8. Halpern JY. Reasoning about uncertainty. Cambridge, MA, USA: MIT Press; 2003.
9. Domingos P, Lowd D. Markov logic: an interface layer for AI. San Rafael, CA, USA: Morgan & Claypool; 2009.
10. Reiter R. A logic for default reasoning. Artif Intell. 1980;13(1–2):81–132. doi:10.1016/0004-3702(80)90014-4.
11. McCarthy J. Circumscription: a form of non-monotonic reasoning. Artif Intell. 1980;13(1–2):27–39.
12. Alchourrón CE, Gärdenfors P, Makinson D. On the logic of theory change: partial meet contraction and revision functions. J Symb Log. 1985;50(2):510–30.
13. Zadeh LA. Fuzzy sets. Inf Control. 1965;8(3):338–53. doi:10.1016/s0019-9958(65)90241-x.

14. Hájek P. Metamathematics of fuzzy logic. Dordrecht, The Netherlands: Kluwer Academic; 1998.
15. Garcez A, Lamb LC, Gabbay DM. Neural-symbolic cognitive reasoning. Berlin/Heidelberg, Germany: Springer; 2015.
16. Besold TR, d'Avila Garcez A, Lamb LC. Neurosymbolic AI: the 3rd wave. AI Mag. 2021;42(2):25–35. doi:10.1007/s10462-023-10448-w.
17. De Raedt L, Dumančić S, Manhaeve R, Marra G. From statistical relational to neural-symbolic AI. In: Proceedings of the IJCAI-2020; 2021 Jan 7–15; Online. p. 4939–46.
18. Jakubův J, Chvalovský K, Goertzel Z, Kaliszyk C, Olšák M, Piotrowski B, et al. MizAR 60 for Mizar 50. In: Proceedings of the ITP 2023; 2023 Jul 31–Aug 4; Timisoara, Romania. p. 19:1–22.
19. Chvalovský K, Korovin K, Piepenbrock J, Urban J. Guiding an instantiation prover with graph neural networks. In: Proceedings of the LPAR 2023, EPiC Series in Computing 94; 2023 Jun 4–9; Manizales, Colombia. p. 38–55.
20. Blaauwbroek L, Cerna DM, Gauthier T, Jakubův J, Kaliszyk C, Suda M, et al. Learning guided automated reasoning: a brief survey. In: Logics and type systems in theory and practice. Cham, Switzerland: Springer; 2024. p. 54–83.
21. Lample G, Lacroix T, Lachaux M-A, Rodriguez A, Hayat A, Lavril T, et al. HyperTree proof search for neural theorem proving. Adv Neural Inf Process Syst. 2022;35:26337–49. doi:10.52202/068431-1910.
22. Jiang AQ, Welleck S, Zhou JP, Li W, Liu J, Jamnik M, et al. Draft, sketch, and prove: guiding formal theorem provers with informal proofs. In: Proceedings of the ICLR 2023; 2023 May 1–5; Kigali, Rwanda.
23. Yang K, Swope AM, Gu A, Chaber R, Hazel C, Sber E, et al. LeanDojo: theorem proving with retrieval-augmented language models. Adv Neural Inf Process Syst. 2023;36:21573–83.
24. Polu S, Han JM, Zheng K, Baksys M, Babuschkin I, Sutskever I. Formal mathematics statement curriculum learning. In: Proceedings of the ICLR 2023; 2023 May 1–5; Kigali, Rwanda.
25. Li Z, Parsert J, Welleck S. A survey on deep learning for theorem proving. In: Proceedings of the COLM 2024; 2024 Oct 7–9; Philadelphia, PA, USA.
26. Sutton RS, Barto AG. Reinforcement learning: an introduction. 2nd ed. Cambridge, MA, USA: MIT Press; 2018.
27. Gabbay DM. Labelled deductive systems. Oxford, UK: Oxford University Press; 1994.
28. Batens D. A universal logic approach to adaptive logics. Log Universalis. 2007;1(1):221–42. doi:10.1007/s11787-006-0012-5.
29. Simon HA. Models of man: social and rational. New York, NY, USA: Wiley; 1957.
30. Lieder F, Griffiths TL. Resource-rational analysis: understanding human cognition as the optimal use of limited computational resources. Behav Brain Sci. 2020;43:e1. doi:10.1017/S0140525X1900061X.
31. van Benthem J. Logical dynamics of information and interaction. Cambridge, UK: Cambridge University Press; 2011.