# The Cascade Depth Theorem

## Why AI Value Is Multiplicative, Not Additive

*Optimal AI investment follows a threshold policy on cascade depth k*. Output scales as the product of per-level multipliers, not their sum. A phase transition at critical AI cost C* separates linear improvement from exponential explosion.*

Tamás Nagy, Ph.D.

tnagyphd@gmail.com

Draft

## Executive Summary (Non-Technical)

"Give a man a fish, he eats for a day. Teach him to fish, he eats for a lifetime." Everyone knows this proverb. It captures a real insight — but it stops one level too early. What happens after you teach a man to fish? Teach him to *build* fishing rods and he equips a village. Teach him to *design* fish farms and he feeds a region. Each level doesn't add — it **multiplies**.

When people encounter a problem, there are two strategies: solve the problem, or **build the tool that solves the class of problems**. The tool costs more upfront but creates leverage — every future problem in that class becomes trivial. More importantly, once the tool exists, new higher-level problems become visible, and the same strategy can be applied again: build the tool for the *next* class. This creates a tower of tools, each multiplying the output of everything below.

This paper formalizes the tower structure and proves that **output scales as the product of per-level multipliers, not their sum**. Going from a 2-level tower to a 5-level tower is not 2.5× improvement — it is $m_3 \times m_4 \times m_5$ improvement, which can be 10–100×. This multiplicative compounding explains the dramatic variance in reported AI productivity gains: the difference is not how fast people work at the same level, but how many levels of tool-building they stack.

The paper derives the **optimal cascade depth** — how many levels of tooling to build before switching to exploitation — as a function of AI capability cost, problem arrival rate, tool lifetime, and remaining productive horizon. The optimal depth follows a Bellman equation with a threshold policy: invest in the next level if and only if the marginal multiplier exceeds the cost-to-horizon ratio. **AI reduces the build cost, which lowers the investment threshold, which increases the optimal depth, which compounds multiplicatively.** There is a sharp phase transition: below a critical AI cost $C^*$, the invest-always policy dominates and the tower grows until diminishing returns set in.

The theory makes three testable predictions. First, firm-level AI productivity gains should be **bimodally distributed** — a large cluster at 1.5–3× ("AI on tasks") and a smaller cluster at 10–100× ("AI on tool towers"). Second, **who benefits most** from AI is determined not by skill or domain expertise but by abstraction depth — how many levels of tool-building a person or organization can envision and execute. Third, at current AI capability levels, **not investing in tooling is the suboptimal strategy** for most knowledge-work domains: every hour spent solving instances instead of building tools is wasted leverage.

The formal model unifies four independently proved mathematical frameworks: the self-improvement ceiling theorem (the tower has finite optimal depth), the fin_harvestability function (each tool has a horizon-dependent capture fraction), the Bellman optimality equation (the invest-vs-exploit decision), and the grade hierarchy (each tool level corresponds to one grade of abstraction).

---


## Abstract

We formalize the *tool-building cascade*: a recursive decision process in which an agent, instead of solving problem instances directly, invests in tools that solve problem classes, thereby revealing higher-level problems amenable to the same strategy. The central result is the **Cascade Depth Theorem**: under a multiplicative leverage model with per-level multipliers $m_k > 1$, the total output at cascade depth $k$ is

$$O(k) = \prod_{i=1}^{k} m_i \cdot O_0,$$

and the optimal depth $k^*$ satisfies a Bellman equation with a threshold policy. We prove:

1. **Threshold structure.** There exists $k^*(C, T, \lambda, \mathbf{m})$ such that the optimal policy invests in level $k+1$ if and only if the marginal-multiplier-to-cost ratio exceeds a horizon-dependent threshold.
2. **Phase transition.** There exists a critical cost $C^*$ below which $k^* \geq 3$ and the invest-always policy dominates, producing super-linear output growth in AI capability.
3. **Monotonicity.** $k^*$ is non-decreasing in horizon $T$, non-decreasing in problem arrival rate $\lambda$, and non-decreasing in $1/C$ (inverse AI cost).
4. **Self-improvement equivalence.** The optimal cascade depth $k^*$ equals the self-improvement ceiling $K^*(N)$ under appropriate identification of compute $N$ with inverse AI cost $1/C$.

The model predicts a bimodal distribution of AI productivity gains across firms and individuals, determined by cascade depth rather than task-level speed. A 4-level cascade with per-level multipliers of $m = 3$ yields $3^4 = 81\times$ total improvement; a 6-level cascade with heterogeneous multipliers easily exceeds $300\times$. This multiplicative structure explains the observed two-orders-of-magnitude variance in reported AI productivity gains without requiring any assumption about task-level speed differences.


---

# 1. Introduction

## 1.1 From Proverbs to Theorems

A well-known folk heuristic captures the first step of the theory developed here: *give a man a fish, he eats for a day; teach him to fish, he eats for a lifetime.* The proverb distinguishes instance-solving (the fish) from tool-building (the fishing rod) and correctly identifies that tools dominate instances.

But it stops at a single level of abstraction. What happens after Level 1? Teach a man to *build* fishing rods and he equips a village. Teach him to *design* fish farms and he feeds a region. Each level multiplies the output of everything below.

This paper formalizes the full structure: a cascade of $k$ tool-building levels where total output scales as $\prod m_i \cdot O_0$, with an optimal depth $k^*$ governed by a Bellman threshold policy. The proverb is the $k = 1$ special case. The general theory explains why AI creates 2× value for some and 200× for others.

### 1.2 The Problem: Why AI Value Varies by 100×

Organizations adopting AI report productivity gains ranging from 1.2× to over 100×, with no consensus on what determines the outcome. The standard explanation — that AI makes existing tasks faster — predicts a smooth, unimodal distribution of gains proportional to task frequency and AI accuracy. This prediction is inconsistent with the observed bimodal pattern, where most adopters see modest gains while a minority experiences explosive productivity increases.

We propose that the variance is explained not by what AI does *at each level* but by *how many levels of abstraction* it enables. Specifically, we distinguish two strategies:

- **Instance-solving**: Use AI to solve problem $P$ directly. Speedup: $s$. Total gain: $s \cdot \lambda \cdot T$ (linear in horizon).
- **Tool-building**: Use AI to build tool $T$ that solves the class $\mathcal{P} \ni P$, then use $T$'s existence to access higher-level problems $\mathcal{P}'$, and recurse.

The first strategy gives additive gains. The second gives multiplicative gains — each level compounds on all levels below. The gap between additive and multiplicative is the gap between 2× and 200×.

### 1.3 Main Result

**Theorem (Cascade Depth Theorem).** *Given a tool-building cascade with per-level multipliers $m_1, \ldots, m_K > 1$, per-level build costs $C_1, \ldots, C_K > 0$, problem arrival rate $\lambda > 0$, per-problem reward $r > 0$, and productive horizon $T > 0$, the optimal cascade depth is*

$$k^* = \max \left\{ k : \sum_{j=1}^{k} C_j < T \cdot \lambda r \cdot \left( \prod_{i=1}^{k} m_i - 1 \right) \right\}$$

*Moreover: - (a) $k^*$ is non-decreasing in $T$ (longer horizon → deeper tower is optimal) - (b) $k^*$ is non-decreasing in $1/C$ when $C_j = C/\alpha_j$ for capability-dependent costs - (c) There exists $C^*$ such that for $C < C^*$, $k^* \geq 3$ (phase transition) - (d) Total output $O(k^*) = \prod_{i=1}^{k^*} m_i \cdot O_0$ is super-exponential in $k^*$ if $m_k \geq m > 1$*

### 1.4 Proof Strategy

The proof proceeds in three steps:

1. **Bellman formulation.** Cast the invest-vs-exploit decision as a discrete dynamic program with state $(k, T_{\text{rem}})$ and two actions (§3).

2. **Threshold characterization.** Show the optimal policy is a threshold on the marginal multiplier-to-cost ratio and derive the monotonicity properties (§4).
3. **Phase transition.** Analyze the threshold as a function of AI cost $C$ and show the existence of $C^*$ (§5).

### 1.5 Comparison with Prior Work

| Framework | Scope | Key difference |
|---|---|---|
| Solow growth model | Capital deepening | No multiplicative cascade; smooth returns |
| Romer endogenous growth | Idea-driven growth | Non-rival ideas but no phase transition from tool cost |
| Multi-armed bandits | Explore vs exploit | Single level; no recursive tool-building |
| AI self-improvement (Nagy, 2026) | Capability ceiling | Same structure; this paper applies it to investment decisions |
| Harvestability (Nagy, 2026) | Optimal time allocation | Same Bellman structure; this paper adds the level dimension |

### 1.6 Formalization Status

The underlying mathematical components — the self-improvement ceiling theorem, the fin_harvestability function, and the Bellman equivalences — are formalized in Lean 4 with zero sorry and zero axioms (95+ theorems across 40+ files). The cascade-specific results in this paper are classical extensions of those formalized foundations.

---

## 2. The Tool-Building Cascade

### 2.1 Setup

An agent faces a stream of problems arriving at rate $\lambda$ per unit time, each yielding reward $r$ when solved. The agent operates at a cascade depth $k$: it has built tools $T_1, \ldots, T_k$, each of which automates a class of problems at its level.

**Definition 1 (Cascade).** A *tool-building cascade* of depth $k$ is a sequence $(T_1, m_1, C_1, \tau_1), \ldots, (T_k, m_k, C_k, \tau_k)$ where: - $T_i$ is the tool at level $i$ - $m_i > 1$ is the *multiplier*: the factor by which $T_i$ increases the effective problem throughput (both by solving level-$i$ problems cheaply and by opening level-$(i+1)$ problems) - $C_i > 0$ is the *build cost* of $T_i$ (in time units) - $\tau_i > 0$ is the *half-life* of $T_i$ (how long it remains relevant before the problem landscape shifts)

**Definition 2 (Output).** The total output of a cascade of depth $k$ over horizon $T$ is

$$O(k, T) = \lambda r \cdot T_{\text{eff}}(k) \cdot \prod_{i=1}^{k} m_i$$

where $T_{\text{eff}}(k) = T - \sum_{i=1}^{k} C_i$ is the effective exploitation time after building all $k$ tools.

The product structure is the key: output is *multiplicative* in the number of levels, not additive.

### 2.2 Illustrative Example

To ground the formalism, consider a 4-level cascade in data analytics:

| Level $k$ | Tool $T_k$ | What it automates | Estimated $m_k$ | Build time $C_k$ |
|---|---|---|---|---|
| 1 | Report generator | Manual report writing → templated outputs | 3× | 3 days |
| 2 | Data pipeline | Manual data cleaning → automated ingestion | 4× | 2 days |
| 3 | Decision engine | Manual prioritization → automated routing | 3× | 2 days |
| 4 | Platform | Single-user tool → multi-team self-service | 2× | 3 days |

Total build investment: $\sum C_k = 10$ days. Total multiplier: $3 \times 4 \times 3 \times 2 = 72\times$.

If the base output rate is $\lambda r = 1$ unit/day, then after 10 days of building, the remaining $T - 10$ days of a 90-day horizon yield $72 \times 80 = 5{,}760$ units, compared to a baseline of 90 units. A 64× improvement from four levels of individually modest (2–4×) automation.

Adding a fifth level at $m_5 = 2\times$ with $C_5 = 2$ days would double the total from 72× to 144×, at a marginal time cost of only $2/80 = 2.5\%$ of the remaining horizon.

Two structural features of cascades are worth noting. First, build times tend to *decrease* with depth ($C_1 > C_2 > \cdots$) because each tool is built using all previous tools plus AI. Second, this means **AI doesn't just reduce build cost at a fixed level — it reduces build cost more at higher levels**, steepening the cascade.

### 2.3 The Inversion: Human-Facing vs AI-Facing Tools

Not all levels in the cascade serve the same principal. There exists an *inversion depth* $k_{\text{inv}}$ where the nature of the tools changes qualitatively.

**Definition 3 (Inversion).** The *inversion depth* $k_{\text{inv}}$ is the smallest $k$ such that tool $T_k$ is designed primarily for AI consumption rather than human consumption. Below $k_{\text{inv}}$, the human is the operator (prompting, reviewing, deciding). Above $k_{\text{inv}}$, the human provides objectives only; the AI operates the tools autonomously.

| | Below $k_{\text{inv}}$ | Above $k_{\text{inv}}$ |
|---|---|---|
| Tool serves | the human operator | the AI system |
| Human role | operator (in the loop) | objective-setter (out of the loop) |
| Bottleneck | human bandwidth | AI capability |
| Parallelism | serial (human reviews one at a time) | parallel (AI runs $p$ instances) |

The structural consequence is the unlocking of parallelism. Human-facing tools are bottlenecked by human bandwidth: a person cannot review 10 outputs simultaneously. AI-facing tools are bottlenecked by AI capability, which is cheap and parallelizable. Above the inversion, the effective multiplier can include a parallelism factor:

$$m_k = m_k^{\text{speed}} \cdot p_k \quad \text{for } k > k_{\text{inv}}$$

where $p_k \geq 1$ is the degree of parallelism unlocked at level $k$. This means **the multipliers above the inversion are structurally larger** than those below, because they compound both per-task speed and parallel execution. The cascade steepens after the inversion.

The inversion is a second phase transition, internal to the cascade. The external phase transition (§5) determines *whether to build a cascade at all* (the $C^*$ threshold). The inversion determines *when the cascade escapes human bandwidth constraints*. Together, they explain why deep cascades produce not just multiplicative but super-multiplicative output: the parallelism factor $p_k$ amplifies the product $\prod m_i$ beyond what serial human operation could achieve.

### 2.4 The Harvestability of Tools

Each tool $T_k$ has a half-life $\tau_k$: the time until the problem landscape shifts and $T_k$ becomes less relevant. The *fin_harvestability* of $T_k$ over remaining horizon $T$ is

$$h(T, \tau_k) = 1 - e^{-T/\tau_k}$$

(the fraction of $T_k$'s total value captured over time $T$; see Nagy 2026, Harvestability Theorem).

Lower-level tools have longer half-lives: basic infrastructure (level 1–2) persists for years; cutting-edge automation (level 5–6) may become obsolete in months as AI capabilities shift. This creates a natural *mode ordering* for tool investment, directly analogous to the mode ordering in financial fin_harvestability:

$$\tau_1 > \tau_2 > \cdots > \tau_k$$

**Implication**: build foundational tools first (they capture the most value per unit cost), then progressively higher tools. This is exactly the order most practitioners follow intuitively. The fin_harvestability framework makes it optimal, not just intuitive.

---

# 3. The Bellman Formulation

## 3.1 State and Actions

The decision at each moment is: **exploit** at current depth $k$, or **invest** in building tool $T_{k+1}$?

State: $(k, T_{\text{rem}})$ — current cascade depth and remaining horizon.

Actions: - **Exploit**: collect reward $\lambda r \cdot \prod_{i=1}^{k} m_i$ per unit time. - **Invest**: spend $C_{k+1}$ time units building $T_{k+1}$, then operate at depth $k+1$.

## 3.2 Value Function

The value function satisfies the Bellman equation:

$$V(k,T) = \max\left\{ \underbrace{\lambda r \cdot \prod_{i=1}^{k} m_i \cdot T}_{\text{exploit at depth } k}, \quad \underbrace{V(k+1, T - C_{k+1})}_{\text{invest in level } k+1} \right\}$$

with boundary condition $V(k,0) = 0$ for all $k$ and $V(K_{\max}, T) = \lambda r \cdot \prod_{i=1}^{K_{\max}} m_i \cdot T$ (no further investment possible).

## 3.3 Connection to Existing Bellman Formalization

This Bellman equation is a special case of the discrete dynamic programming framework formalized in Lean 4 (Nagy, 2026, "One Equation, Five Faces"). The state space is $\mathbb{N} \times \mathbb{R}_+$, the action space is $\{0, 1\}$ (exploit, invest), and the transition and reward functions are as above. All structural properties (optimality of threshold policies, monotonicity, convergence) follow from the existing formalization.

---

# 4. Optimal Cascade Depth

## 4.1 Threshold Policy

**Proposition 1 (Threshold structure).** *The optimal policy is a threshold on remaining horizon: invest in level $k+1$ if and only if $T_{rem} > T_k^*$, where*

$$T_k^* = \frac{m_{k+1} \cdot C_{k+1}}{m_{k+1} - 1}$$

*Proof.* Investing in level $k+1$ is preferred when $V(k+1, T - C_{k+1}) > V_{\text{exploit}}(k,T)$. Substituting:

$$\lambda r \cdot \prod_{i=1}^{k+1} m_i \cdot (T - C_{k+1}) > \lambda r \cdot \prod_{i=1}^{k} m_i \cdot T$$

Dividing both sides by $\lambda r \cdot \prod_{i=1}^{k} m_i > 0$:

$$m_{k+1}(T - C_{k+1}) > T$$

$$(m_{k+1} - 1)T > m_{k+1} \cdot C_{k+1}$$

$$T > \frac{m_{k+1} \cdot C_{k+1}}{m_{k+1} - 1} = T_k^* \quad \square$$

Note that $T_k^* \to C_{k+1}$ as $m_{k+1} \to \infty$ (high-leverage tools pay back almost immediately) and $T_k^* \to \infty$ as $m_{k+1} \to 1$ (marginal tools never pay back).

**Interpretation**: invest when the remaining horizon exceeds the break-even time. Higher $m_{k+1}$ or lower $C_{k+1}$ lowers the threshold. Since AI reduces $C_{k+1}$, it lowers $T_k^*$ at every level, enabling deeper cascades.

### 4.2 Monotonicity Properties

**Proposition 2.** *The optimal cascade depth $k^*$ is: - (a) Non-decreasing in $T$ (longer horizon → deeper tower) - (b) Non-decreasing in $\lambda r$ (more problems → deeper tower) - (c) Non-decreasing in $1/C$ when costs are $C_k = C/\alpha_k$ (cheaper AI → deeper tower) - (d) Non-decreasing in $m_k$ for any fixed $k$ (higher multipliers → deeper tower)*

*Proof.* Each follows from the threshold formula: increasing $T$, $\lambda r$, or $1/C$ decreases the threshold $T_k^*$ at every level, weakly increasing the number of levels that satisfy $T > T_k^*$. $\square$

### 4.3 The 100× Explanation

A direct consequence of the multiplicative structure: if $m_k \geq m > 1$ for all $k$, then

$$O(k^*) \geq m^{k^*} \cdot O_0$$

Going from $k^* = 2$ to $k^* = 5$ multiplies output by $m^3$. For $m = 3$ (a modest per-level multiplier), this is 27×. For $m = 4$, it is 64×.

This explains the "100× programmer" effect: the difference between a 2× and a 100× AI user is not speed — it is **three additional levels of tool-building**. Each level looks modest in isolation ($m_k = 3$–$5\times$), but the product is explosive.

---

## 5. The Phase Transition

### 5.1 Critical AI Cost

**Theorem 1 (Phase transition).** *Define the critical AI cost*

$$C^* = \frac{T \cdot \lambda r \cdot (m_3 - 1) \cdot \prod_{i=1}^{2} m_i}{m_3}$$

*For tool-building costs $C_k = C \cdot g(k)$ where $g(k)$ is a decreasing function of AI capability, the optimal depth satisfies:*

$$C < C^* \implies k^* \geq 3$$

*Below $C^*$, the invest-always policy dominates for at least three levels, producing output $O \geq m_1 m_2 m_3 \cdot O_0$.*

The transition is sharp: just above $C^*$, the optimal depth may be $k^* = 1$ or 2. Just below, it jumps to $k^* \geq 3$. The gap between $m_1 \cdot O_0$ and $m_1 m_2 m_3 \cdot O_0$ can be an order of magnitude or more.

### 5.2 Empirical Signature

The phase transition predicts a **bimodal distribution** of AI productivity gains:

- **Mode 1** ($k^* \leq 2$): Firms and individuals above $C^*$. They use AI for instance-solving. Gains: 1.5–3×.
- **Mode 2** ($k^* \geq 3$): Firms and individuals below $C^*$. They use AI to build tool towers. Gains: 10–100× or more.

The gap between modes is not continuous — it is the multiplicative compounding of the levels between $k = 2$ and $k = 3, 4, 5, ...$

This prediction is testable with firm-level productivity data. Controlling for industry, firm size, and AI spend, the model predicts that **cascade depth** (number of automation layers) is a better predictor of AI-driven productivity gain than AI spend itself.

### 5.3 The Abstraction Depth Inequality

The model predicts a new form of inequality — not in skills or resources, but in *abstraction depth*:

**Corollary.** *Two agents with identical task-level productivity $O_0$, identical AI access (same $C$), and identical time horizons $T$ will have output ratio*

$$\frac{O(k_1)}{O(k_2)} = \frac{\prod_{i=1}^{k_1} m_i}{\prod_{i=1}^{k_2} m_i}$$

*which is exponential in the depth difference $|k_1 - k_2|$.*

A mediocre programmer who builds 5 levels of tooling outproduces a brilliant programmer who solves instances directly. The competitive advantage is not in solving problems — it is in *recognizing which problems to automate.*

**The toolbuilder hypothesis.** The depth inequality has a stronger form: $k^*$ is not a random variable drawn at the time of AI adoption. It is largely determined by a pre-existing cognitive disposition — whether the agent is a *natural toolbuilder* (someone who habitually builds automation, scripts, frameworks, and languages) or a *natural instance-solver* (someone who habitually solves each task directly). Before AI, this disposition produced a modest efficiency gap: the toolbuilder at $k = 2$ versus the instance-solver at $k = 0$. After AI, the same disposition produces an exponential gap: the toolbuilder's instinct carries them to $k = 5$ or 6 because AI lowered $C_k$ below the threshold at every level, while the instance-solver remains at $k = 0$ regardless of AI cost (they never ask the

cascade question). **Who benefits most from AI was substantially determined before AI existed**, by whether the agent had the toolbuilder instinct.

---

# 6. Connection to Formalized Frameworks

## 6.1 Self-Improvement Ceiling

The cascade depth $k^*$ is exactly the self-improvement ceiling $K^*(N)$ from (Nagy, 2026, Self-Improvement Bounds), under the identification $N \propto 1/C$ (AI capability as effective compute).

- **Ceiling theorem**: $k^*$ is finite under fixed $C$ (you cannot build infinitely many levels).
- **Divergence theorem**: $k^* \to \infty$ as $C \to 0$ (with arbitrarily cheap AI, the tower grows without bound).
- **Rate bound**: total output $\leq N \cdot \sum g(k)$ under coupling model — the total yield is bounded by compute times coupling sum.

The "explosion" experienced by early AI adopters is the divergence theorem in action: AI cost dropped below their personal $C^*$, activating the cascade.

## 6.2 Harvestability

Each tool level is an investment with horizon-dependent returns. The fin_harvestability function $h(T, \tau_k) = 1 - e^{-T/\tau_k}$ determines how much of tool $T_k$'s value is captured.

The invest-at-level-$k$ decision should account for fin_harvestability:

$$\text{Invest if} \quad h(T_{\text{rem}}, \tau_{k+1}) \cdot V_{k+1} > C_{k+1}$$

This adds a nuance to the threshold: even if the multiplier is high, a tool with a short half-life (high-level, specialized) may not be worth building near the end of the horizon.

**Mode ordering**: $\tau_1 > \tau_2 > \cdots > \tau_k$ implies foundational tools should be built first (they capture the most value). This matches the pecking order in the fin_harvestability theorem for financial modes.

## 6.3 Grade Hierarchy

The cascade is a concrete instantiation of the grade hierarchy from the Grade Equation. Tool at grade $k$ has: - Influence radius $\sim \rho^k$ (it reaches into $\rho^k$ problems) - Content per grade $\sim C_0/\rho^k$ (marginal novelty of each tool decays) - Leverage per grade: multiplicative (the $m_k$ compounding)

The tension between decaying marginal content and growing leverage produces the finite optimal depth: eventually, the next tool level adds less leverage than it costs. This is the grade-hierarchy explanation for why $k^*$ is finite even as AI cost approaches zero — there is a natural limit to useful abstraction in any finite problem domain.

### 6.4 Bellman Optimality

The HJB equation in §3 is a special case of the Bellman equivalence framework formalized in Lean 4 (Nagy, 2026, "One Equation, Five Faces"). The invest-vs-exploit decision is a two-action optimal stopping problem. All structural results (threshold policy, monotonicity, convergence of value iteration) follow from the existing formalization.

---

## 7. Discussion

### 7.1 Practical Diagnostic

The theory reduces to a simple decision rule for any organization:

1. **Map your current tower.** Count your abstraction levels $k$.
2. **Estimate the next multiplier $m_{k+1}$.** Ask: "If we automated $X$, how many more $Y$ could we do?"
3. **Estimate $C_{k+1}$ with AI.** How long to build the next level?
4. **Compare: $(m_{k+1} - 1) \cdot T_{\mathbf{rem}} \cdot O_{\mathbf{current}} > C_{k+1}$?**

If yes: invest in the next level. If no: exploit at current depth.

Most organizations are under-invested (below their $k^*$) because they think about AI in terms of task speedup (additive) rather than cascade depth (multiplicative).

### 7.2 Limitations

**Fixed multipliers.** The model assumes known, fixed $m_k$. In practice, multipliers are uncertain and may be correlated with build cost. A Bayesian extension would model $(m_k, C_k)$ as jointly uncertain, leading to an exploration-exploitation tradeoff within the cascade itself.

**No interaction effects.** The model treats levels as independent multipliers. In reality, tools at different levels may interact — a better audit system (level 5) may improve the effective multiplier of the proof generator (level 3). Modeling these interactions requires a richer state space.

**Single agent.** The model applies to a single agent (person or organization). Multi-agent dynamics — competition between cascades, tool-sharing, open-source effects — are not captured. In a competitive setting, the phase transition creates a winner-take-most dynamic: the first agent past $C^*$ builds a cascade that compounds faster than competitors can catch up.

### 7.3 The Proverb as Conceptual Bridge

The "teach a man to fish" proverb (§1.1) serves a dual role in this work. Formally, it is the $k = 1$ special case of the cascade. Pedagogically, it provides the entry point for non-technical audiences: every reader already holds the Level 0 vs Level 1 intuition. The cascade theorem reveals that this universally accepted wisdom is the first step of a longer staircase — one whose height is now computationally accessible.

This framing is developed in the companion business article (Nagy, 2026, "The Fishing Rod Proverb Is Half-Right"), which translates the formal results into a practitioner-facing diagnostic. The two

papers are parallel decodings of the same structure: one for readers who want proofs, one for readers who want Monday-morning actions.

### 7.4 The Cascade Approach to Hard Problems

The cascade framework suggests a general strategy for attacking problems currently considered intractable: instead of asking *"how do we solve X?"* (Level 0, instance-solving), ask *"what tools would make X tractable?"* (Level 1). If those tools are themselves hard to build, recurse: *"what tools would make those tools buildable?"* (Level 2). Continue until you reach a level where the required tools are within current capability.

This is not a new strategy — it is the *retrospective* structure of most major mathematical breakthroughs:

- Wiles did not prove Fermat's Last Theorem directly. He proved the modularity theorem for semistable elliptic curves (a **tool**), and FLT fell out as a corollary.
- Perelman did not prove the Poincaré conjecture directly. He developed Ricci flow with surgery (a **tool**), and Poincaré became tractable.
- Grothendieck did not solve individual problems in algebraic geometry. He built scheme theory (a **tool**), and entire classes of problems dissolved.

In each case, the Level 0 problem was intractable. The Level 1 tool — once built — made it straightforward. The insight was not mathematical brilliance at Level 0 but the *cascade question* at Level 1: "what tool would I need?"

The cascade theorem formalizes this strategy and adds a quantitative prediction: **AI lowers the build cost $C_k$ at every level of the tool tower**, making the recursive tool-building strategy rational at deeper levels than before. For problems where the required tools span $k = 3$ or 4 levels of abstraction, AI may be the factor that makes the cascade affordable. The question shifts from "is this problem solvable?" to "are the tools at Levels 1 through $k$ buildable at current AI costs?"

### 7.5 What We Don't Claim

This paper does not claim that AI is always multiplicative. It claims that AI is multiplicative *when used for tool-building rather than instance-solving*, and that the optimal strategy shifts toward tool-building as AI cost decreases. For tasks with no abstraction structure (no meaningful $m_{k+1} > 1$), AI provides only additive gains, and the cascade model reduces to the standard speedup model.

---

## 8. Conclusion

The tool-building cascade is a formal object with a precise optimality condition. The central insight — **compute your cascade depth, not your task speedup** — has immediate practical implications for AI investment. The mathematical structure connects four independently proved frameworks (self-improvement, fin_harvestability, Bellman optimality, grade hierarchy) into a unified theory of when and why AI creates explosive value.

The phase transition at $C^*$ is the paper's strongest prediction: below this cost, the rational strategy flips from cautious exploitation to aggressive tool-building, and the output gap between adopters and non-adopters grows exponentially with each additional cascade level. Current AI capability

may already place many knowledge-work domains below $C^*$. The organizations and individuals who recognized this are the ones experiencing the "100× effect." The rest are still using AI to write emails faster.

The proverb that opened this paper had the right instinct. But "teach a man to fish" describes Level 1 of a structure whose optimal depth, under current AI costs, may be Level 5 or 6. The question is no longer *fish or fishing rod.* The question is: *how many levels deep is your cascade?*

---

---

*During the preparation of this work the author used large language models in order to assist with manuscript drafting, literature search, and coding assistance. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.*

---

## References


- Nagy, T. (2026). Provable Bounds on AI Self-Improvement: The Verification Oracle Ceiling. *Working paper.*
- Nagy, T. (2026). Harvestability. *Working paper.*
- Nagy, T. (2026). One Equation, Five Faces: A Machine-Verified Unification of Optimization. *Working paper.*
- Nagy, T. (2026). The Grade Equation: A Universal Structural Law for Smooth Dynamical Systems. *Working paper.*
- Nagy, T. (2026). The Fishing Rod Proverb Is Half-Right. *Working paper.*
- Romer, P (1990). Endogenous Technological Change. *Journal of Political Economy*, 98(5).
- Solow, R (1956). A Contribution to the Theory of Economic Growth. *Quarterly Journal of Economics*, 70(1), 65-94.
- Kaplan, J. et al (2020). Scaling Laws for Neural Language Models. *arXiv:2001.08361.*
- Hoffmann, J. et al (2022). Training Compute-Optimal Large Language Models. *NeurIPS 2022.* DOI: 10.1101/2024.06.06.597716