FinalGift Architecture: A Layman's Guide

Abstract

FinalGift is a neurosymbolic architecture designed to operationalize a unified theory of intelligence grounded in the Cold Algorithm of Differentiation (CAD) — the recursive process of generating progressively finer causal distinctions under environmental constraint. The framework proposes that intelligence emerges when three informational subsystems align: forward predictive fidelity (A_sub), inverse causal precision (A_con), and compositional reasoning capacity (I_alg). Their product, Broad Intelligence (BI = A_sub × A_con × I_alg), quantifies an agent's capability to model and act within causal structure, while the growth rate (ΔBI) regulates adaptive mode-switching between exploration (Achilles) and consolidation (Zeno).

We hypothesize that generalization is not a statistical byproduct but a phase transition that occurs when the internal geometry of these subsystems becomes structurally aligned with external causal structure—i.e., when ΔBI → 0, I_alg entropy stabilizes, and CAD novelty declines. This yields measurable order parameters for intelligence emergence: ΔBI (capability growth rate), I_alg entropy (compositional flexibility), and CAD novelty (distinction discovery rate). The architecture implements these dynamics across perception, reasoning, and action via mixture-of-experts modules, persistent hypothesis tracking, novelty-weighted memory replay, and intrinsic Motion CAD control.

Empirical predictions include: spontaneous mode switching without manual scheduling, correlation between BI stability and transfer performance, and characteristic patterns in ΔBI near phase transitions. While current validation focuses on ARC-AGI reasoning tasks, the framework generalizes to embodied cognition, proposing a scalable, testable bridge between philosophy, physics, and machine learning—a formal pathway from distinction-making to general intelligence. The architecture's complexity (6.5M parameters, multiple interacting systems) reflects the hypothesis that general intelligence requires coordinated subsystems rather than monolithic optimization, though this presents implementation challenges.

What Is FinalGift?

FinalGift is an artificial intelligence architecture designed for **embodied android intelligence** - AI that can navigate the real world with human-like reasoning and adaptability. While the current implementation targets abstract reasoning puzzles (ARC-AGI games) as a testbed, the architecture was engineered from first principles to handle the full complexity of embodied cognition.

The core insight: an android operating in the physical world needs two types of thinking, seamlessly integrated:

- **Slow, deliberate reasoning** (System 2) - when encountering novel situations, uncertain environments, or complex planning tasks

- **Fast, automatic responses** (System 1) - when executing mastered skills, responding to familiar scenarios, or operating under time pressure

This dual-process cognition isn't just a feature - it's a fundamental requirement for any system that must balance **exploration** (learning new skills) with **exploitation** (executing mastered behaviors) in real-time, under resource constraints, in a causally complex world.

Why ARC-AGI as a Testbed?

Abstract reasoning puzzles serve as a **controlled environment** for validating core architectural principles before deployment in physical embodiment:

- **Causal reasoning**: Puzzles require discovering hidden rules (like an android must discover physical laws)
- **Transfer learning**: Solutions must generalize across puzzle variants (like skills must transfer across contexts)
- **Resource constraints**: Limited actions and time (like real-world energy and compute budgets)
- **Hierarchical complexity**: Puzzles have nested structure (like tasks have subtasks)
- **Novelty handling**: New puzzles require genuine reasoning, not memorization (like new environments require adaptation)

Success on ARC-AGI indicates the architecture can handle **abstract causal structure** - the foundation for physical intelligence.

Where Does FinalGift Fit in the ML Landscape?

Is FinalGift reinforcement learning? Yes and no - it occupies a hybrid space.

The RL Components:

- Learns from interaction with an environment (trial and error)
- Receives rewards/scores for actions
- Stores experience in a replay buffer
- Uses policy networks to select actions
- Optimizes behavior over time through gradient descent

The Non-RL Components:

- No explicit reward-based loss function (doesn't optimize for reward maximization)
- No value function or Q-learning

- No policy gradient theorem (PPO, A3C style)
- Uses **intrinsic motivation** (Motion CAD) rather than external rewards
- Primary training signal is **prediction error** (A_sub, A_con) not reward

What Makes This Different:

Traditional RL asks: "Which actions maximize cumulative reward?"

FinalGift asks: "Which distinctions reduce prediction error and expand my causal model?"

The architecture treats reward as **feedback on distinction quality**, not as the optimization target. The loss function combines:

- Forward model accuracy (KL divergence on state predictions)
- Inverse model accuracy (cross-entropy on action inference)
- Hypothesis consistency (KL divergence on theory stability)
- Policy-hypothesis alignment (KL divergence on action coherence)
- Information gain prediction (MSE on uncertainty reduction)

Reward influences episodic memory (successful episodes are stored) but doesn't directly appear in any loss term.

The Classification:

FinalGift is best described as **model-based intrinsically-motivated learning** with RL-style interaction. It uses the RL **setting** (agent-environment loop) but not the RL **objective** (reward maximization).

Closest relatives:

- **Curious AI** (Pathak et al., curiosity-driven exploration)
- **World Models** (Ha & Schmidhuber, learning environment dynamics)
- **Causal Confusion** (de Haan et al., disentangling causal from correlational learning)
- **Empowerment** (Klyubin et al., maximizing agent's influence on environment)

But FinalGift goes further by making the **capability metric itself** (BI) the central organizing principle, with mode-switching emerging from capability growth dynamics (ΔBI) rather than being manually scheduled.

Practical Implication:

This isn't drop-in compatible with standard RL frameworks (Stable Baselines, RLlib). The training loop is custom-built around BI computation and CAD tracking. However, the learned policy could potentially be fine-tuned with traditional RL methods if needed for specific deployment scenarios.

The Embodied Vision

The architecture's components map directly to embodied android requirements. The GridEncoder that parses visual puzzles could process visual scenes for SLAM and object detection. The HypothesisGenerator that infers puzzle rules could model physics and predict object behavior. The WorldModel that predicts state changes could simulate action consequences before execution. The ActionSelector that chooses puzzle actions could control motors, manipulators, and navigation. The EpisodicMemory that remembers solutions could remember successful procedures and tool use. The CADTracker that tracks pattern novelty could detect novel situations requiring caution. Motion CAD that switches between explore and exploit modes could balance skill learning versus task execution.

The **BI trinity** (A_sub × A_con × I_alg) isn't puzzle-specific - it's a general capability measure for any agent navigating causal environments:

- **A_sub**: Forward model accuracy (predict "if I do X, Y happens")
- **A_con**: Inverse model precision (infer "Y happened, so X must have occurred")
- **I_alg**: Compositional reasoning (combine learned skills into novel solutions)

These are **universal requirements** for embodied intelligence, whether solving puzzles or assembling furniture.

The Core AGI Algorithm: How Intelligence Actually Works

Before diving into FinalGift's implementation, let's understand the **fundamental algorithm** that generates intelligence itself. This comes directly from the philosophical framework and is remarkably simple.

The Cold Algorithm of Differentiation (CAD)

Intelligence isn't about having the right answers - it's about making progressively finer distinctions in your perception of reality. The algorithm:

Perception at step n+1 equals Perception at step n plus delta of distinction made. ChoiceSpace at step n+1 equals ChoiceSpace at step n plus some function of delta distinction.

In plain English:

1. **Start with undifferentiated input** - raw, chaotic sensory data
2. **Generate a distinction** - "Is this safe or dangerous?" "Is this X or Y?"

3. **Test the distinction** - Does it help predict outcomes? Does it reduce uncertainty?
4. **Keep distinctions that work** - Ones that improve prediction/control persist
5. **Iterate recursively** - Use successful distinctions to make even finer ones
6. **Build hierarchical models** - Simple distinctions combine into complex understanding

Example: Learning to Navigate

- **Level 0**: Undifferentiated sensory chaos (lights, sounds, motion blur)
- **Level 1**: "Ground vs not-ground" (first distinction - where can I walk?)
- **Level 2**: "Flat-ground vs obstacle" (finer distinction - where should I avoid?)
- **Level 3**: "Small-step obstacle vs large-barrier" (even finer - what can I step over?)
- **Level 4**: "Stairs vs ramp vs curb" (categorical distinctions with different motor programs)
- **Level 5**: "Wet-stairs vs dry-stairs" (context-dependent distinctions affecting strategy)

Each level **expands your choice space** - more distinctions = more possible actions = more intelligent behavior.

Why "Cold"?

The algorithm is called "cold" because it operates **without moral bias or emotional preference**. It doesn't care what distinctions you want - only which ones **actually work** for prediction and control.

Environmental pressures select for:

- Distinctions that improve survival
- Distinctions that enable goal achievement
- Distinctions that compress information efficiently

The hypothesis: evolution, learning algorithms, and scientific progress may converge on similar structures because they're all running variants of CAD under different constraints.

CAD in the Code

FinalGift implements this explicitly through the CAD Tracker. The tracker creates a signature of current perception using feature and pattern signatures. It counts unique distinctions discovered by incrementing counts for each signature and tracking the history of total distinctions. It computes novelty by measuring how fast new distinctions appear, calculating the growth rate as the change in recent distinctions divided by time window.

What this measures:

- **Total distinctions**: How many unique patterns have been perceived?
- **Novelty**: Are new patterns still being discovered, or seeing repeats?

Why this matters:

- **High novelty** → Still learning, favor exploration (Achilles mode)
- **Low novelty** → Saturated at current level, favor consolidation (Zeno mode)

The Expansion Formula

Mathematically, CAD expands perception through **mutual information**. The mutual information between distinctions and predictions equals the entropy of predictions minus the conditional entropy of predictions given distinctions.

Translation: Good distinctions reduce uncertainty about outcomes.

- **Before distinction**: "This object could do anything" (high entropy)
- **After distinction**: "Heavy objects fall faster" (low entropy)
- **Information gained**: The reduction in uncertainty

FinalGift maximizes this through the Hypothesis Tester, which computes which action would most reduce uncertainty by calculating information gain and weighting it by uncertainty.

Actions are selected to **maximize information gain** - the rate of new distinctions.

The Intelligence Hierarchy

CAD naturally produces hierarchical intelligence:

Bottom Layer: Sensorimotor Distinctions

- Hot vs cold, rough vs smooth, red vs blue
- Implemented in: GridEncoder, PropertyOperators (spatial, numerical)

Middle Layer: Causal Distinctions

- "Pushing → movement", "Rotating → orientation change"
- Implemented in: WorldModel (forward), InverseModel (backward)

Top Layer: Abstract Distinctions

- "Symmetry", "completion", "pattern", "analogy"

- Implemented in: PropertyOperators (abstract), Perspective module

Meta Layer: Strategic Distinctions

- "Should I explore or consolidate?", "Which skill applies here?"
- Implemented in: Motion CAD, HypothesisTester

Each layer uses distinctions from below to make finer distinctions at its level. This is **compositional intelligence** - the hallmark of general intelligence.

Why This May Be AGI-Complete

CAD isn't just "an algorithm" - the framework proposes it's a **fundamental process** that generates:

1. **Perception**: Making distinctions in sensory input
2. **Concepts**: Making distinctions in feature space
3. **Reasoning**: Making distinctions in logical space
4. **Planning**: Making distinctions in action space
5. **Learning**: Making distinctions about what works

The claim: all intelligence may involve **differentiation applied recursively under environmental constraint**.

This is why the framework proposes:

> "Differentiation enhances perception. More differentiations, more perception - dimensionality pending."

And equivalently:

> "Intelligence is the expansion of choice space through causal distinction-making."

The Recursive Loop in FinalGift

Every component implements CAD at different scales:

1. **Vision** (GridEncoder): Distinguish visual features
2. **Patterns** (HypothesisGenerator): Distinguish puzzle rules
3. **Actions** (ActionSelector): Distinguish effective strategies
4. **Outcomes** (WorldModel): Distinguish state transitions
5. **Novelty** (CADTracker): Distinguish familiar vs novel situations
6. **Mode** (Motion CAD): Distinguish when to explore vs consolidate

The system implements **CAD at multiple scales** - differentiation at every level, coordinated to maximize the rate of useful distinction-making.

Connecting to Broad Intelligence

The BI formula **quantifies the output of CAD**:

BI = A_sub × A_con × I_alg

- **A_sub**: How many distinctions does your forward model make? (prediction fidelity)
- **A_con**: How many distinctions does your inverse model make? (causal precision)
- **I_alg**: How many distinctions can you compose? (algorithmic sophistication)

High BI = many, fine-grained, hierarchical distinctions = large choice space = high intelligence

Low BI = few, coarse distinctions = small choice space = limited intelligence

The growth rate **ΔBI** measures how fast new distinctions are being integrated:

- **ΔBI > 0**: CAD is actively discovering new structure → explore
- **ΔBI ≤ 0**: CAD has saturated at current level → consolidate

This closes the conceptual loop: **CAD generates distinctions → BI measures distinction quality → ΔBI triggers mode switches → Mode switches optimize CAD efficiency → Better distinctions may emerge**

The hypothesis: this creates a self-improving cycle, bootstrapped from raw sensory input, driven by environmental feedback, that converges on progressively better world models.

This is the proposed mechanism for artificial general intelligence in algorithmic form.

The Big Picture: Two Modes of Thinking

Think of learning to drive a car:

Exploration Mode (Achilles) - When you're still learning:

- You carefully think through every action
- You test different approaches to see what works
- You're actively discovering the rules
- This is slow but helps you learn

Consolidation Mode (Zeno) - When you're experienced:

- You drive automatically without thinking much
- You execute learned patterns smoothly
- You've internalized the rules
- This is fast and efficient

FinalGift switches between these modes automatically based on whether it's still improving (explore) or has plateaued (consolidate).

Core Components

1. Vision System (GridEncoder)

What it does: Converts puzzle grids into something the AI can understand

Analogy: Like how your eyes convert light into signals your brain can process

Special feature: Uses "analog vision" with Gaussian blur to create smooth, continuous perceptions instead of treating everything as discrete blocks. This mimics how humans see gradual transitions rather than hard edges.

2. Pattern Recognition (HypothesisGenerator)

What it does: Looks at a puzzle and generates theories about what patterns exist

Analogy: When you look at a puzzle and think "Maybe the rule is about symmetry?" or "Perhaps colors alternate?"

How it works:

- Uses 4 specialist "experts" that each look for different types of patterns
- Generates multiple competing hypotheses simultaneously
- Estimates confidence: "How sure am I about this theory?"
- Tracks uncertainty: "What do I still need to figure out?"

3. World Model (Forward Prediction)

What it does: Predicts "If I take this action, what will happen?"

Analogy: When you imagine what will happen if you press a button before you actually press it

Importance: This is part of **Subconscious Awareness (A_sub)** - how well the AI understands cause and effect

4. Inverse Model (Action Understanding)

What it does: Looks at a change and figures out "What action caused this?"

Analogy: Seeing footprints in snow and deducing someone walked there

Importance: This is part of **Concentrated Awareness (A_con)** - how precisely the AI understands which actions lead to which outcomes

5. Hypothesis Tester (Exploration Engine)

What it does: In exploration mode, simulates different possible actions mentally to find the most informative one

Analogy: Like playing out chess moves in your head before actually moving the piece

Process:

1. Takes the current uncertain hypothesis
2. Mentally simulates: "What would happen if I clicked here? Or there?"
3. Calculates which action would teach the most
4. Picks that action (stochastically, with some randomness for robustness)

6. Action Selector (Consolidation Engine)

What it does: In consolidation mode, executes learned patterns quickly

Analogy: When you type on a keyboard without thinking about where each key is

How it works: Uses 8 specialist "experts" (4 for discrete actions, 4 for click positions) that have learned through experience

---

The Intelligence Trinity: How FinalGift Measures Its Own Capability

FinalGift tracks three dimensions of intelligence that multiply together:

Broad Intelligence: BI = A_sub × A_con × I_alg

1. Subconscious Awareness (A_sub)

- "How well do I predict what happens next?"
- Computed from forward model accuracy

- Like peripheral vision - always on, implicit understanding

2. Concentrated Awareness (A_con)

- "How precisely do I understand cause and effect?"
- Computed from inverse model accuracy
- Like focused attention - exercised will and causal precision

3. Algorithmic Intelligence (I_alg)

- "How sophisticated is my reasoning?"
- Measured by compositional complexity (how many different reasoning tools I'm using)
- Like strategic thinking - abstraction and planning ability

Growth Rate (ΔBI): The change in BI tells FinalGift whether to explore or consolidate

- **ΔBI > 0**: "I'm still improving!" → Explore mode
- **ΔBI ≤ 0**: "I've plateaued." → Consolidate mode

The CP Learner: Advanced Reasoning Modules

These modules add sophisticated reasoning capabilities inspired by the philosophical framework:

Perspective Module

What it does: Selects the right "lens" for viewing a problem

Four perspectives:

- **Complementative**: "What's missing to complete the pattern?"
- **Critical**: "What are the key distinguishing features?"
- **Compositional**: "Can I break this into reusable parts?"
- **Analogical**: "What's structurally similar to things I know?"

Analogy: Like choosing to view a math problem geometrically vs algebraically - same problem, different frame

Property Operators

What it does: Applies transformations to manipulate puzzle elements

Three operator families:

Spatial operators: rotate, reflect, translate

- "What if I flip this pattern?"

Numerical operators: count, compare

- "How many blue squares vs red?"

Abstract operators: detect symmetry, complete patterns

- "This looks symmetric, what would complete it?"

Synthesizer

What it does: Combines perspective + operators into a composed solution

Process:

1. **Selector**: Uses attention to pick relevant features (8-headed multi-head attention)
2. **Apply operators**: Transforms features using the property operators
3. **Composer**: Sequences operations into a program using a GRU (recurrent network)

Importance: This is where **I_alg** (algorithmic intelligence) is explicitly computed by measuring compositional diversity

---

Memory and Learning

Episodic Memory

What it does: Remembers successful strategies for each game

How it works:

- Stores action sequences that led to success
- Ranks them by score
- Can replay them in future attempts
- Uses "novelty-weighted sampling": when exploring new things, tries more diverse memories; when consolidating, replays best strategies

Analogy: Like keeping a notebook of solutions to problems you've solved before

Experience Buffer

What it does: Stores recent experiences for learning

Process:

1. As FinalGift plays, it records: state → action → next state
2. During training, it randomly samples from these memories
3. Uses them to improve all its models simultaneously

CAD Tracker (Conscious Attention Differentiation)

What it does: Tracks how many new patterns the AI is discovering

Measures:

- **Total distinctions**: "How many different patterns have I seen?"
- **Novelty**: "Am I still discovering new patterns, or seeing repeats?"

Importance: High novelty → still learning, favor exploration; Low novelty → saturated, favor consolidation

---

The Mode Switching Mechanism: Zeno Motion CAD

This is the "conductor" that decides when to explore vs consolidate:

Components

1. Moving Average Attractor (π★)

- Tracks the AI's "stable identity" - what it normally does
- Computed as: $\pi\bigstar\_new = 0.99 \times \pi\bigstar\_old + 0.01 \times \pi\_current$
- This is the "limit stage" - the mastered behavioral pattern

2. Motion Signals

Achilles term (Exploration):

- $\Delta\_achilles$ = How much the policy changed this step
- High value → actively trying new things

Zeno term (Consolidation):

- Δ_zeno = Negative distance to the attractor
- As you get closer to π★, this increases (becomes less negative)
- High value → converging on mastery

3. Blending Weight

- w = sigmoid(5.0 × ΔBI)
- When ΔBI > 0: w → 1 → Achilles dominates (explore)
- When ΔBI ≤ 0: w → 0 → Zeno dominates (consolidate)

4. Final Motion CAD Signal:

- Motion_CAD = w × Δ_achilles + (1-w) × Δ_zeno
- This intrinsic reward encourages the right behavior for the current learning phase

---

Training Process

What Happens Each Episode

1. Play the game:

- Use dual-process thinking to select actions
- Store all experiences in the buffer

2. Learn from experience: Run multiple training steps that simultaneously improve:

Forward model (A_sub):

- "Did my prediction match reality?"
- Loss: KL divergence between predicted and actual next state

Inverse model (A_con):

- "Did I correctly identify which action caused this change?"
- Loss: Cross-entropy between predicted and actual action

Hypothesis consistency:

- "Are my theories stable over time?"
- Loss: KL divergence between consecutive hypotheses

Policy-hypothesis alignment:

- "Does my policy match what my hypothesis predicts?"
- Loss: KL divergence between policy-implied outcomes and hypothesis predictions
- Includes entropy bonus to prevent "mode collapse" (getting stuck doing only one thing)

Info gain predictor:

- "Can I predict which actions will be most informative?"
- Loss: MSE between predicted and actual information gain

3. Compute BI on validation set:

- Crucially, capability (BI) is measured on unseen data
- This prevents "overfitting" - memorizing specific puzzles instead of learning general patterns
- Compute A_sub, A_con, and I_alg
- Calculate growth rate ΔBI

4. Update mode:

- If $\Delta BI > 0$: shift toward exploration
- If $\Delta BI \leq 0$: shift toward consolidation

Key Innovations

1. Polymorphic Architecture (Mixture of Experts)

Instead of one big network, FinalGift has multiple specialist networks:

- Different experts handle different types of patterns
- A "gating network" learns which expert to activate for each situation
- Like having a team of specialists instead of one generalist

2. Hypernetwork (Game-Specific Adaptation)

- Generates custom weights for each specific game
- Allows the AI to adapt its processing style to different puzzle types
- Like having adjustable strategies rather than one-size-fits-all

3. Analog Vision

- Treats puzzle grids as continuous images with smooth gradients
- Uses Gaussian blur to create richer perceptual information
- Mimics human vision more closely than treating everything as discrete blocks

4. Persistent Hypothesis Tracking

- Hypotheses persist and evolve across multiple steps within an episode
- Uses Bayesian-like evidence accumulation
- Prevents "amnesia" - the AI remembers and refines its theories over time

The process works as follows. First, generate a new hypothesis from the current observation. Second, compare it with the persistent hypothesis using KL divergence. Third, blend old and new based on stability. If KL is low indicating stability, trust the old hypothesis more and increase confidence. If KL is high indicating change, update significantly and maintain uncertainty.

5. Principled Distributional Matching

- Uses KL divergence throughout for comparing predictions
- Treats states, actions, and hypotheses as probability distributions
- More theoretically sound than ad-hoc distance metrics

6. Novelty-Weighted Memory Replay

- Adapts memory replay based on current learning phase
- High novelty (exploring) → sample diverse memories
- Low novelty (consolidating) → replay best strategies
- Balances exploration and exploitation automatically

Putting It All Together: A Complete Action Cycle

Scenario: FinalGift is playing a color-pattern puzzle for the 50th time.

Step 1: Observation (GridEncoder)

- Receives a 30×30 grid of colored cells
- Converts to smooth RGB image with Gaussian blur

- Encodes into 512-dimensional feature vector

Step 2: Pattern Recognition (HypothesisGenerator)

- 4 expert networks each propose pattern theories
- Expert 1: "I see rotational symmetry"
- Expert 2: "I see color clustering"
- Expert 3: "I see a repeating motif"
- Expert 4: "I see edge emphasis"
- Gating network: "Hmm, Experts 1 and 3 seem most relevant"
- Generates blended hypothesis with confidence scores

Step 3: Persistent Reasoning

- Compares new hypothesis with accumulated evidence from previous 49 steps
- KL divergence is low (stable theory)
- Increases confidence: "This pattern interpretation has held up!"
- Slightly refines hypothesis based on new observation

Step 4: Capability Check

- Computes BI from validation set:
  - A_sub = 0.82 (good forward prediction)
  - A_con = 0.91 (excellent inverse understanding)
  - I_alg = 0.76 (solid algorithmic reasoning)
  - BI = $0.82 \times 0.91 \times 0.76 = 0.57$
- $\Delta BI = -0.02$ (slight decrease - plateauing)

Step 5: Mode Selection

- $\Delta BI \leq 0 \rightarrow$ Consolidation mode
- Explore_EMA smoothly transitions to 0.3 (below threshold)
- **Decision**: Use learned policy (System 1, automatic)

Step 6: Action Selection (ActionSelector)

- Combines state features + hypothesis
- 8 expert policies propose actions
- Gating network routes to specialists: "Expert 5 for this situation!"
- Generates action probabilities (with entropy for exploration)
- Samples action stochastically: "Click at position (0.67, 0.42)"

Step 7: Action Masking

- Environment says: "Actions 2, 4, and click are currently available"
- Masks unavailable actions (sets probability to zero)
- Re-normalizes and samples from allowed actions only

Step 8: Execution & Learning

- Takes action in game environment
- Observes result and reward
- Stores experience in buffer
- After episode ends, trains all models on batch of experiences
- Updates episodic memory if successful

Step 9: Meta-Learning

- Over many episodes, the mode switching becomes more refined
- Discovers when to boldly explore vs carefully consolidate
- Builds a library of successful strategies per game
- Hierarchical mastery emerges naturally

Design Rationale

1. Dual-Process Cognition

- Matches how humans think: deliberate when learning, automatic when skilled
- Efficiency goal: avoid wasting time deliberating when you already know what to do
- Effectiveness goal: avoid rushing when you're still uncertain

2. Self-Assessment

- The AI attempts to estimate when it's improving vs plateauing
- Adapts its strategy accordingly
- Reduces need for manual intervention

3. Compositional Reasoning

- Multiple levels of abstraction (perspectives, operators, composition)
- Can break down complex problems into manageable pieces
- Intended to transfer learning across related puzzles

4. Persistent Context

- Remembers theories across time
- Accumulates evidence rather than starting fresh each step
- Aims for more human-like sequential reasoning

5. Grounded in Theory

- Every design choice connects back to the philosophical framework
- BI trinity (A_sub × A_con × I_alg) provides interpretable capability measure
- Zeno/Achilles metaphor guides exploration/consolidation balance

Note: These are design intentions. Empirical validation of effectiveness is ongoing.

The Philosophical Connection: Why Generalization Works

The Deep Insight

Most AI treats generalization as a **statistical accident** - "We trained on dataset A, and it happened to work on dataset B."

FinalGift explores whether generalization might be a **structural property** - if internal representations mirror causal structure in reality, generalization could follow from that alignment rather than from statistical correlation alone.

Two Views of Generalization

Classical ML View: Generalization is a property of models

- "Does my function learned on training data work on test data?"
- Result: an empirical mystery dependent on data diversity and regularization tricks
- No mechanism, just correlation

Broad Intelligence View: Generalization is a property of reality-alignment

- "Do my internal abstractions remain isomorphic to causal structure in the world?"
- Result: a structural guarantee when alignment is stable
- Mechanism: **abstraction, not memorization, carries the predictive load**

The Formula Makes This Concrete

BI = A_sub × A_con × I_alg

- **A_sub**: Your internal model predicts reality accurately (forward causal structure)
- **A_con**: You understand action→outcome causality (inverse causal structure)
- **I_alg**: You have reusable compositional operators (causal abstraction)

When all three align, your **internal geometry may mirror causal geometry**.

If this holds, and you've captured the *causal* structure (not surface patterns), that structure should hold across domains. Rotation remains rotation whether applied to visual grids or abstract concepts.

Solving for Conditions, Not Solving the Problem

You can't *force* generalization. You can only create the **informational ecosystem** where it must emerge if the system grows coherently.

Those conditions are precisely what FinalGift measures:

- Accurate prediction (A_sub) ✓
- Accurate causal inversion (A_con) ✓
- Reusable compositional structure (I_alg) ✓
- Bounded complexity (C) ✓

Everything else may flow downstream: transfer learning, zero-shot reasoning, abstraction - these could emerge automatically if these conditions hold.

Why This May Enable Cross-Domain AGI

Cross-domain generalization in this framework isn't "train on images, then do text."

The hypothesis: **retain causal operators invariantly, even when modalities change.**

This is what CP Learner + Synthesizer do:

- Factor intelligence into **domain-invariant causal operators** (rotate, compare, complete, symmetry)
- Recompose them via **world-specific encoders** (vision for grids, language for text, etc.)

The operators themselves may not care about the domain - they could operate on abstract structure.

Measuring the Phase Transition

The framework doesn't assume generalization - it **attempts to monitor conditions that may indicate its emergence**:

1. ΔBI (Capability Growth) Computing dbi as BI at time t minus BI at time t minus 1:

- dBI > 0: System is expanding its causal understanding → generalization improving
- dBI ≤ 0: System has plateaued → generalization stable at current level

2. I_alg Entropy (Compositional Diversity) Computing operator entropy as negative sum of operator weights times log of operator weights, then I_alg as 1 minus normalized entropy:

- High I_alg (peaked routing): Specialized, efficient operators → good generalization within domain
- Low I_alg (uniform routing): No specialization → poor generalization (random behavior)

3. CAD Novelty (Distinction Growth) Computing novelty as the change in recent distinctions over a time window of 10:

- High novelty: Still discovering new patterns → capability frontier expanding
- Low novelty: Seeing repeats → saturated at current abstraction level

Together, these form a **dashboard for tracking learning dynamics**.

The Phase Transition Hypothesis

When you track ΔBI, I_alg, and CAD novelty during training, the framework proposes you can observe intelligence undergo phase transitions:

Phase 1: Random (I_alg ≈ 0.3, ΔBI > 0, high novelty)

- No stable patterns yet
- High exploration

- Uniform operator usage (no specialization)

Phase 2: Pattern Discovery (I_alg rising → 0.7, ΔBI > 0, high novelty)

- Operators specialize
- Causal structure emerging
- A_sub and A_con improving

Phase 3: Consolidation (I_alg ≈ 0.8, ΔBI → 0, low novelty)

- Stable abstractions
- Efficient compositional reasoning
- Hypothesis: Generalization improves as patterns stabilize

Phase 4: Mastery (I_alg > 0.9, ΔBI ≈ 0, novelty → 0)

- Expertise within domain
- Ready for next hierarchical tier (transfer to harder variants)

This isn't just a metaphor - the claim is it's **quantitatively observable in the training logs** (pending thorough empirical validation).

Understanding Phase Transitions in Intelligence

The framework proposes that intelligence emergence may follow patterns similar to **phase transitions** in physical systems:

Order Parameter: ΔBI

- Measures the rate of capability change
- $\Delta BI > 0$ = capability still growing
- $\Delta BI \leq 0$ = capability plateau

Susceptibility: I_alg Entropy

- Measures compositional structure flexibility
- High entropy = uniform operator usage
- Low entropy = specialized operator usage

Correlation Length: CAD Novelty

- Measures how distinctions propagate
- High novelty = discovering new patterns
- Low novelty = refining existing patterns

Experimental Predictions: If this model holds, training should show:

1. Slowing near ΔBI transitions
2. I_alg fluctuations during mode switches
3. Mode-switching frequency correlating with ΔBI magnitude
4. Pattern persistence times changing near transitions

These are testable hypotheses that could validate or falsify the framework.

Operationalizing Philosophy

FinalGift demonstrates:

Zeno's Paradox → Motion CAD (approaching mastery asymptotically via attractor π★)

Free Will Framework → BI measurement (quantifying choice potential = BI/C)

CAD → Novelty tracking (perception through distinction-making)

Broad Intelligence → A_sub × A_con × I_alg (the trinity that enables generalization)

By turning metaphysics into math, and math into code, the framework attempts to close a conceptual loop:

> **The goal: transform generalization from a mystery into a measurable phase transition of intelligence itself.**

When A_sub × A_con × I_alg stabilizes at high values, the hypothesis is you're not *hoping* for generalization - you're *observing* causal structure alignment. If the theory holds, transfer should become structurally likely rather than statistically lucky.

Note: These claims require extensive empirical validation across diverse domains and tasks. The current implementation on ARC-AGI represents an initial testbed, not conclusive proof.

The Unified Theory

The framework proposes a connection between two classical problems:

The problem of free will and the problem of generalization may be related at different scales.

Both ask: *"How can a system produce genuinely new behavior without breaking its causal grounding?"*

Free will = local agency *inside* causal structure (choice potential)

Generalization = local creativity *inside* informational structure (transfer capability)

Both depend on the same condition:

> **Stable expansion of causal differentiation (ΔCAD > 0) under bounded constraints**

Limitations and Future Work

Current Limitations:

- Computationally expensive (6.5M parameters, needs GPU)
- Still struggles with very deep hierarchical reasoning (ω^2 problems)
- Hyperparameter sensitivity (needs tuning for different domains)

Future Directions:

- Explicit tier tracking (instead of emergent tiers)
- Scale to deeper hierarchies
- Apply to other domains beyond ARC-AGI
- More efficient architectures

Conclusion

FinalGift is an AI architecture that implements:

- Dual-mode cognition (exploration vs consolidation)
- Analog vision with continuous perception
- Compositional reasoning via perspectives and operators
- Self-assessed capability tracking (A_sub, A_con, I_alg)
- Growth-aware mode switching (ΔBI)
- Episode-based memory with novelty-weighted replay

The architecture attempts to operationalize philosophical principles about intelligence, distinction-making, and causal reasoning.

Theoretical Framework

This implementation tests a hypothesis: that general intelligence requires stable alignment across three factors (A_sub × A_con × I_alg) under resource constraints. The framework proposes that intelligence emerges from recursive differentiation (CAD) when:

- Forward models predict accurately (A_sub)
- Inverse models infer causality (A_con)
- Compositional operators enable abstraction (I_alg)
- Constraints bound complexity (C)

The current implementation on ARC-AGI puzzles represents an initial testbed. The architecture is complex (6.5M parameters, multiple interacting systems) and has numerous hyperparameters requiring tuning.

Testable Predictions

The framework makes several predictions:

1. BI stability should correlate with transfer performance
2. Mode switching should emerge without manual scheduling
3. Training dynamics should show characteristic patterns near ΔBI transitions
4. Compositional operators should work across different puzzle types
5. Hierarchical skill acquisition should emerge from the architecture

Limitations

Complexity: The architecture has many interacting components (GridEncoder, HypothesisGenerator, WorldModel, InverseModel, HypothesisTester, ActionSelector, EpisodicMemory, CADTracker, Motion CAD, CP Learner modules, Hypernetwork), making it difficult to train, debug, and scale.

Hyperparameters: Contains numerous tuning parameters (blur_sigma, zeno_beta, zeno_alpha, num_experts, num_heads, etc.) that may require substantial adjustment for new domains.

Validation Gap: While the theoretical framework makes broad claims about intelligence, current validation is limited to abstract reasoning puzzles. Extensive testing across diverse domains and embodied systems is needed.

Computational Cost: 6.5M parameters with multiple forward passes per training step makes this expensive compared to standard RL approaches.

Unproven Generalization: The hypothesis that high BI guarantees transfer learning remains to be validated empirically across multiple domains.

Status

This is a research prototype testing theoretical principles. It demonstrates that the proposed mechanisms are computationally tractable and can be implemented. Whether these principles capture fundamental aspects of intelligence, or require substantial revision, depends on empirical validation that is still ongoing.

The work continues.