# **Governing Faculty Blueprint: Comprehensive Cognitive Architecture**

## **I. Algorithmic Instantiation**

In this section, we formalize each component of the Governing Faculty Blueprint as an algorithmic module with specified inputs, processing rules (including mathematical models), outputs, and temporal dynamics. We detail five key modules: (1) the Comparator (evidence accumulation mechanism), (2) the Assent Gate (threshold gating of actions), (3) the Recursive Adjudication Agent (RAA) – the meta-controller orchestrating recursive evaluation, (4) the Norm Repository (store of normative rules/values), and (5) the Norm Conflict Resolver (mechanism for adjudicating between conflicting norms). We also describe the temporal dynamics (e.g. discrete cycles) linking these modules, and provide pseudocode and a flowchart-style outline for clarity.

### **Comparator Module: Multi-Attribute Drift-Diffusion Model (MDDM)**

Function: The Comparator integrates multiple factors (impulse salience, urgency, norm congruence) into a running evaluation of a candidate action or impulse. It continuously accumulates “evidence” for or against allowing the impulse, analogous to a drift-diffusion process, until a decision threshold is reached. This implements a form of conflict monitoring and evidence accumulation .

Input Format: The inputs to the Comparator are a set of impulses (proposed actions or responses) along with associated attributes: (a) Salience (how attention-grabbing or strong the impulse is, e.g. an “urge strength” scalar), (b) Norm Congruence (a score indicating consistency of the impulse with each active norm, which can be positive if the impulse aligns with norms or negative if it violates norms), and (c) Urgency (a time-pressure or need metric indicating how urgent it is to act on this impulse). Each impulse’s inputs can be represented as a vector (salience\_i, norm\_congruence\_i, urgency\_i) that may evolve over time (e.g. urgency increasing if unresolved).

Processing Rules: The Comparator uses a multi-attribute drift-diffusion model to process these inputs. In a standard drift-diffusion model (DDM), evidence accumulates over time toward an upper or lower bound, with a drift rate determined by input quality and a stochastic noise term . Here, the drift rate v\_i for impulse i is computed as a weighted combination of the attributes:

* Drift Rate: v\_i = w\_s \* S\_i + w\_n \* N\_i + w\_u \* U\_i + ξ, where:  
  + S\_i = salience of impulse i (normalized, e.g. between 0 and 1),
  + N\_i = net norm congruence of impulse i (we define this as positive if norms permit or encourage the action, negative if norms forbid it; details below),
  + U\_i = urgency level for impulse i (which may increase each time step if unresolved, simulating rising pressure),
  + w\_s, w\_n, w\_u are weight parameters tuning the influence of salience, norm congruence, and urgency on the evidence accumulation.
  + ξ is a noise term (e.g. Gaussian noise per time-step) reflecting stochastic variability in evidence sampling.

Thus, an impulse that is highly salient (S\_i high) and norm-congruent (N\_i high) with urgent need (U\_i high) will have a large positive drift, meaning evidence accumulates quickly in favor of allowing/acting on the impulse. Conversely, an impulse that strongly violates norms (N\_i very negative) would produce a negative drift (evidence accumulating against the action). The multi-attribute DDM thus continuously tallies the “case for” vs “case against” an impulse by integrating these factors. This extends standard DDMs to multi-criteria decisions . We implement separate accumulators for each impulse (if multiple impulses are being considered simultaneously), or alternately a single accumulator that compares the impulse against a “do nothing” baseline.

Output Format: The primary output of the Comparator for each impulse is the accumulated decision variable (the evidence total) and a flag if it has reached either an upper bound (indicating a “Yes, allow action” decision) or a lower bound (“No, veto action”). The Comparator might also output intermediate values such as current drift rates or conflict signals. If multiple impulses are present, the Comparator could output the one with the highest evidence or any that hit threshold.

Equations – Multi-Attribute Drift and Norm Influence: To formalize norm congruence: suppose the Norm Repository provides a set of relevant norms for the situation (see Norm Repository below). For impulse i, we compute a normative congruence score N\_i = Σ\_j (α\_j \* compliance\_{ij}), where compliance\_{ij} measures how well impulse i satisfies norm j (ranging from +1 for fully compliant to –1 for fully violating), and α\_j is the salience/weight of norm j (which could be the norm’s importance or activation level). In effect, N\_i is positive if the impulse mostly aligns with active norms, and negative if it transgresses them. This N\_i enters the drift as shown above. Urgency U\_i can be modeled as an increasing function over time if no decision is made – for example, U\_i(t) = U\_{i0} + k \* t (linearly growing) or an asymptotic function as time in the decision cycle increases, reflecting the idea that the longer an urge remains unresolved, the stronger the drive to resolve it (to avoid indefinite indecision). This implements an urgency signal that biases the model toward making a choice as time passes . Salience S\_i might be set initially by perceptual factors or internal drive strength and could also evolve (for instance, stimuli that attract attention longer effectively increase salience). All these factors modify drift in each discrete timestep of the simulation.

Temporal Dynamics: The Comparator operates in discrete timesteps (e.g. ∆t = 100 ms per cycle). At each timestep, for each active impulse i, it updates the evidence E\_i by: E\_i <- E\_i + v\_i \* ∆t + noise. The process continues until E\_i hits the upper decision threshold +Θ or lower threshold –Θ for that impulse. The thresholds Θ are set by the Assent Gate module (see below) and can vary dynamically. Importantly, the evidence E\_i can carry over between cycles if no decision was reached (unless it decays; see “decay/persistence” below). If multiple impulses are being accumulated in parallel, the one that hits its bound first would “win” the competition for attention/behavior (assuming only one action can be taken at a time), or potentially multiple can reach decision if they pertain to different modalities.

Decay or Persistence: If an impulse fails to reach threshold in one cognitive cycle (e.g. after a certain number of timesteps or by a certain deadline), the model can either persist the partial accumulation or decay it. We incorporate a leaky accumulation: unchosen impulses’ evidence values E\_i decay by a factor (e.g. multiply by 0.9) each cycle if they haven’t reached threshold, preventing stale proposals from lingering indefinitely. However, the impulse is not forgotten entirely – a remnant may carry into the next cycle, implementing persistence of unchosen impulses. This ensures that an impulse that repeatedly arises or remains relevant will gradually accumulate evidence (especially via rising urgency) even if initially suppressed.

Example of a drift-diffusion evidence accumulation process. Each colored trace shows the stochastic evolution of an evidence signal over time until hitting an upper (blue dashed) or lower (red dashed) threshold. A higher threshold (blue line) requires more evidence and yields slower, more controlled decisions, whereas a lower threshold (red line) leads to faster, more impulsive decisions. In the Comparator’s multi-attribute drift-diffusion model, similar accumulation occurs with drift rates influenced by urgency, norm alignment, and salience. In our Comparator, a norm-congruent, high-urgency impulse would produce a steep upward drift, reaching the “Yes” bound quickly, whereas an impulse that violates a strong norm could drive the evidence downward to the “No” bound instead. This mechanism effectively monitors conflicts between impulses and norms: when norms oppose an impulse, the drift may be near zero or negative, reflecting a conflict that slows or prevents reaching the “go” threshold. Such a system aligns with psychological models where a dedicated monitor (often associated with ACC) registers conflict between competing responses and accumulates evidence until a control signal is issued.

Pseudocode (Comparator): Below is pseudocode outlining the Comparator’s operation per cycle:

# Comparator Module: Multi-Attribute Drift-Diffusion

# Inputs: impulses list with attributes (S\_i, N\_i, U\_i)

# Outputs: decision outputs per impulse (None if no threshold reached yet)

for impulse i in impulses:

# Update urgency over time

impulse.U = impulse.U + k \* Δt # urgency increase each timestep

# Compute drift rate from current attributes

drift\_rate = w\_s \* impulse.S + w\_n \* impulse.N + w\_u \* impulse.U

# Add stochastic noise

drift\_rate = drift\_rate + random\_gauss(0, σ)

# Update evidence accumulation

impulse.E = impulse.E + drift\_rate \* Δt

# Leak/decay unchosen evidence

impulse.E = impulse.E \* decay\_factor # e.g. 0.99 (slight decay) if needed

# Check decision bounds

if impulse.E >= Theta\_i\_upper:

output\_decision(i, "YES") # signal impulse i as approved

elif impulse.E <= Theta\_i\_lower:

output\_decision(i, "NO") # signal impulse i vetoed

(The thresholds Theta\_i\_upper/lower are provided by the Assent Gate and may be impulse-specific or globally set. The Norm Conflict Resolver may also influence the drift or directly override decisions; see below.)

### **Assent Gate Module: Threshold Modulation and Gating**

Function: The Assent Gate serves as a dynamic gatekeeper that determines whether the accumulated evidence in the Comparator is sufficient to trigger an action (i.e. to “give assent” to the impulse). It implements variable decision thresholds that can be modulated by neuromodulator-like signals (analogous to serotonin) and emotional state. Essentially, while the Comparator accumulates evidence, the Assent Gate decides how much evidence is required to commit to a “Yes” decision (acting on the impulse) or to definitively “No” veto it. By adjusting thresholds, the Assent Gate controls impulsivity vs. restraint – a higher threshold means more evidence (and thus time) is required, promoting caution, whereas a lower threshold allows quicker, impulsive action .

Input Format: Inputs include the current baseline thresholds (which could be preset or context-dependent) and any modulatory signals that adjust these thresholds. We introduce a Serotonin-like signal 5HT\_level as a key modulator: this is a scalar representing the influence of a serotonin analog in the system (with higher values corresponding to calmer, more inhibition-oriented state, and lower values corresponding to impulsive, low-inhibition state). Additionally, the emotional context (from the Emotion-Affect system) can input signals to raise or lower thresholds (for example, fear might raise threshold to avoid rash actions, whereas anger might lower it). The Assent Gate may also receive instructions from the RAA if special conditions apply (e.g. if the RAA decides to tighten control in high-stakes situations, it could command a threshold increase).

Processing Rules: The Assent Gate computes an effective decision threshold Θ for the current cycle for each Comparator accumulator (or globally). We model the threshold as variable: Θ = Θ\_base + ΔΘ(5HT\_level, emotion). For instance:

* Serotonin Modulation: We posit ΔΘ is positively correlated with 5HT\_level. In other words, higher serotonin-like tone raises the threshold, requiring more evidence to act. This is consistent with serotonin’s role in promoting behavioral inhibition and patience . Conversely, low 5HT\_level lowers the threshold, making it easier for impulses to trigger a “go” decision (leading to impulsivity). For example, if baseline Θ\_base is 1.0 (in normalized units), a high 5HT state might set Θ = 1.2, whereas a low 5HT state might set Θ = 0.8.
* Emotion Modulation: Emotional signals can adjust thresholds. We will detail emotional effects in Section III, but briefly: “Hot” emotions (high arousal, e.g. anger, excitement) tend to lower the threshold (impatience, urgency to act), while “Cold” emotions (low arousal or deliberate mood, e.g. sadness, cautiousness) tend to raise thresholds (promoting deferral of action). Thus, ΔΘ might also include terms for these: e.g. a hot emotion might contribute a –0.2 to the threshold, a cold emotion a +0.2, etc. If multiple impulses are present, the threshold might be adjusted individually per impulse type or context (for instance, fear might specifically raise threshold for dangerous actions).

The Assent Gate thus implements a kind of gain control on the decision process, analogous to how speed-accuracy tradeoff is modulated in the brain by threshold adjustments . A high threshold enforces that only very well-justified impulses (with strong accumulated evidence) pass through (i.e. the agent assents only after careful consideration), whereas a low threshold means the agent will assent readily to impulses with minimal evidence.

Output Format: The output of the Assent Gate is the set of decision thresholds (upper and lower bounds) at the current moment, which the Comparator uses to judge decisions. It may also output a final “assent decision” in cases where threshold is reached: e.g. it could produce a binary “Yes/No” signal indicating whether the impulse is allowed or rejected. In practice, once the Comparator’s evidence crosses the threshold, that event can be considered the output (“Yes, trigger action” or “No, suppress action”). The Assent Gate could further gate this output if needed (for example, a final safety check).

Temporal Dynamics: The threshold can be updated each timestep or each cycle. We simulate that at each 100ms cycle, the Assent Gate recalculates thresholds based on the current modulation signals. If serotonin level or emotional state changes mid-decision, the threshold can dynamically shift. (E.g., if suddenly a calming influence occurs, threshold might rise, possibly preventing an impulse from reaching it as quickly.)

Integration with Comparator: Initially, before evidence accumulation begins, the Assent Gate sets Theta\_upper = +Θ and Theta\_lower = –Θ for each accumulator. The Comparator checks these as it integrates. When evidence ≥ Θ or ≤ –Θ, that indicates an assent or rejection decision. If the threshold is dynamic, one way to implement it in the accumulation process is to normalize evidence by threshold or to compare to a time-varying bound (some DDM frameworks allow collapsing bounds over time; here bounds could potentially change if modulators shift).

Serotonin analogy: This design aligns with findings that increasing serotonin promotes waiting and inhibits premature responding , whereas serotonin depletion can increase impulsive (premature) actions . In our model, a high serotonin analog raises thresholds (making “premature” crossings less likely), and low serotonin lowers thresholds (making it easier for evidence to trigger a decision quickly). This mimics how SSRIs (which elevate serotonin) have been observed to enhance behavioral inhibition and impulse control , and how low serotonin states correspond to greater impulsivity . It effectively implements a neuro-inspired “braking” mechanism on the Comparator.

Pseudocode (Assent Gate):

# Assent Gate Module: Threshold modulation

# Inputs: base\_threshold, serotonin\_level, emotion\_state

# Output: Theta\_upper, Theta\_lower for decision bounds

# Base threshold (could depend on context difficulty)

Theta = base\_threshold # e.g. 1.0

# Modulate by serotonin (5HT) level

Theta\_mod\_5HT = k\_ser \* (serotonin\_level - normal\_level)

Theta += Theta\_mod\_5HT # increase if 5HT above normal, decrease if below

# Modulate by emotional state

if emotion\_state is "hot": # high arousal, impulsive

Theta += Theta\_mod\_hot # Theta\_mod\_hot might be negative (lower threshold)

elif emotion\_state is "cold": # low arousal, cautious

Theta += Theta\_mod\_cold # Theta\_mod\_cold positive (raise threshold)

# (masking emotions or others could also adjust threshold or temporarily disable outputs)

# Ensure Theta stays within sensible bounds

Theta = max(min(Theta, Theta\_max), Theta\_min)

# Set symmetric upper/lower bounds (could also allow asymmetry if needed)

Theta\_upper = Theta

Theta\_lower = -Theta

output(Theta\_upper, Theta\_lower)

In the above, k\_ser is a scaling factor for serotonin’s effect. The emotion adjustments Theta\_mod\_hot and Theta\_mod\_cold are determined by the Affect system (Section III). We impose limits Theta\_min, Theta\_max to avoid extremes. The output thresholds are then used by the Comparator in its next accumulation cycle.

### **RAA Module: Recursive Adjudication Agent (Meta-Control Layer)**

Function: The Recursive Adjudication Agent (RAA) is the meta-controller of the architecture, responsible for overseeing the iterative decision process, handling unresolved conflicts, and orchestrating multiple passes of evaluation (hence “recursive”). It plays the role of the executive that can call for additional deliberation, query higher-order norms, or halt the process if needed. Conceptually, this corresponds to the system’s self-reflective layer – analogous to a supervisory attention or executive function in cognitive terms – that adjudicates when the automatic processes (Comparator + Assent Gate) are uncertain or when norms conflict. We will later refer to this as the Recursive Adjudication Layer (RAL) or Normative Executive System (NES) to emphasize it as the emergent executive (see Section V), but here we describe its algorithmic role.

Input Format: The RAA monitors signals from other modules: it takes input about the state of the Comparator (e.g. evidence levels, whether multiple impulses are active, whether conflict exists between high-value norms), outcomes of the Norm Conflict Resolver (if a conflict was detected and resolved or if resolution is indeterminate), and the time/iteration count of the current decision process. It also is aware of contextual information like the importance or stakes of the decision (which might influence how many recursive cycles to allow, or whether to apply stricter norms). Essentially, the RAA observes meta-information: e.g., “no decision has been reached yet and two cycles have passed,” or “impulse A and impulse B are in conflict with different norms,” or “the agent has attempted to veto an impulse but it keeps reappearing.”

Processing Rules: The RAA employs policies to determine when to iterate or intervene. Key functions include:

* Conflict Detection & Recursion: If the Comparator evidence is oscillating or indecisive due to conflicting influences (e.g., norms pulling one way, salience another), the RAA detects this as a conflict condition. It can then trigger a recursive evaluation. In practice, this means the RAA will reset or continue the accumulation process for another round, possibly after engaging the Norm Conflict Resolver for guidance. For example, suppose impulse X has strong salience but violates an important norm Y; the Comparator’s drift might hover around zero (stalemate). The RAA notices that after a typical decision time no threshold was reached and conflict flags are raised. It could then do the following recursively: consult the Norm Conflict Resolver (to see if norm Y absolutely forbids X or if there’s a permissible exception), potentially adjust weights (e.g. increase the weight of norm congruence to break the tie), and allow another run of accumulation. This iterative checking embodies a reflective loop, where the system “thinks twice” about a tricky impulse.
* Arbitration of Multi-Impulses: If multiple impulses compete (say impulses A and B concurrently), the RAA can allocate processing (like attention) between them. It might run the Comparator for impulse A first, or run them in parallel but, upon a conflict, focus on the more critical one. It effectively arbitrates priority of decision if needed at a meta-level.
* Drift Rate or Threshold Adjustment: On subsequent iterations, the RAA can modify parameters to help reach a resolution. For example, it might escalate urgency signals (if an action must be decided, RAA can inject a higher urgency in the next cycle to ensure a decision occurs). Or it could adjust thresholds temporarily (e.g. if too strict, loosen them to allow a decision, or vice versa if a dangerous impulse is about to slip through, raise threshold further). These adjustments are analogous to allocating more or less control as per expected value considerations – much like the brain’s executive might allocate control based on expected value of control .
* Stopping Criteria: The RAA also decides when to stop the recursion. For instance, it might allow at most N cycles of deliberation (to avoid infinite loops). If after N cycles no “Yes” decision is reached for an impulse, the RAA might force a “No (withhold action)” by default (the system chooses inaction as the safest default if it cannot confidently assent). This ensures the agent eventually acts or withholds rather than hanging indefinitely. Alternatively, the RAA could escalate the decision to another system or ask for external input if applicable (though in an autonomous agent, that might mean deferring the decision entirely).

Output Format: The RAA does not directly output a yes/no on the impulse; rather, it outputs control signals to other modules: e.g., a command to restart the Comparator for another cycle, updated weights or thresholds, or a call to engage the Norm Conflict Resolver. One can think of the RAA’s output as an action plan for the cognitive process itself (meta-actions). It may also log a final outcome reason (e.g., “impulse rejected due to conflict between Norm1 and Norm2”).

Temporal Dynamics: The RAA operates at the cycle boundaries. Suppose each cycle is 100ms; the RAA checks at the end of each cycle whether a decision was made. If yes, RAA lets it proceed to action execution. If not, RAA evaluates if the situation warrants another cycle. The notion of persistence of unchosen impulses ties in: the RAA can carry forward the partial state (with or without decay as described). If conflict persists across cycles, RAA might attempt resolution at a higher level (like engaging more abstract norms or considering long-term implications). This introduces a potentially variable decision latency: simple decisions (no norm conflict) finish in 1–2 cycles; complex conflicts might take several cycles as RAA repeatedly engages conflict resolution and re-evaluation. This corresponds to the idea that difficult moral or self-control decisions involve more prolonged deliberation (which we will link to behavioral predictions in Section IV).

Pseudocode (RAA):

# RAA Module: Recursive meta-control

# Called at end of each cycle or upon detection of conflict condition.

inputs = (impulses, norm\_conflict\_flags, cycle\_count, max\_cycles)

# Check if any decision reached in Comparator

if any impulse has output\_decision:

allow those outputs to proceed (assent or veto)

terminate processing for those impulses

else:

# No decision yet – possibly conflict or just not enough evidence

if conflict\_detected or cycle\_count >= max\_cycles:

# Engage Norm Conflict Resolver if not already done

conflicts = detect\_conflicting\_norms(impulses)

if conflicts:

resolution = NormConflictResolver.resolve(conflicts)

if resolution == "veto":

force\_rejection\_of(conflicts.affected\_impulse)

return

elif resolution == "allow":

bias\_in\_favor\_of(conflicts.affected\_impulse)

# e.g., reduce threshold or boost drift for that impulse

# If still unresolved or no explicit resolution:

if cycle\_count >= max\_cycles:

# Safety cutoff: if too many cycles, default action

default\_decision = "NO" # e.g., withhold by default

output\_decision(conflicts.or\_all\_impulses, default\_decision)

return

# Prepare for next cycle (adjust parameters if needed)

if conflict\_detected:

# adjust drift/threshold to break tie on next cycle

for impulse in impulses:

if impulse.in\_conflict:

impulse.U \*= 1.2 # increase urgency to push a decision

# optionally adjust thresholds or weights here

# Loop for another cycle of evidence accumulation

continue\_processing\_next\_cycle()

In this pseudocode, conflict\_detected could be a flag set if norms for a single impulse disagree (one norm says “must do” while another says “must not do”), or if multiple impulses are mutually exclusive but both have support. The RAA uses NormConflictResolver (detailed in its section) to attempt a logical resolution. If that yields a clear veto or allowance, the RAA implements it (overriding the normal drift outcome if necessary). If not resolved, the RAA potentially modifies parameters and continues. There’s a safeguard that if the cycle count exceeds max\_cycles (for instance, max\_cycles might be 5 or some function of context urgency), it will force a decision (in this example by default denying the impulse to err on the side of caution – one could also choose a default action in line with highest norm priority).

In summary, the RAA ensures the decision process can iterate and handle edge cases of indecision or norm stalemate, rather than requiring a single pass. This recursiveness addresses the “homunculus” concern by distributing control: RAA doesn’t itself decide the content of action, but rather adjusts the parameters and calls of the decision process until coherence emerges (more on this in Section V).

### **Norm Repository Module: Norm Library and Retrieval**

Function: The Norm Repository is a structured memory store containing the agent’s known norms, rules, and principles that guide behavior. It provides the norms relevant to the current context for use by other modules (Comparator and Norm Conflict Resolver) and updates these norms over time through learning. In this section (Algorithmic Instantiation), we focus on how norms are represented and retrieved (the static usage); Section II will extend this to how norms are acquired and evolve. Essentially, the Norm Repository acts like a knowledge base of “shoulds and shouldn’ts” – analogous to a conscience or rule library – which can be queried given a situation or impulse.

Input Format: Inputs to the Norm Repository include the current contextual situation and candidate actions/impulses. Context can be encoded as features (location, social setting, roles, etc.), and the impulse is a proposed action with attributes (e.g. “eat the cake” with context “in diet program” or “lie to friend” with context “friend asks for honest opinion”). A query to the repository might look like: query(context, impulse) asking “what norms apply here?” The repository might also accept authority inputs or learning signals as inputs when updating (discussed in Section II, Norm Acquisition).

Processing Rules (Retrieval): The repository performs a pattern-match or lookup to find norms whose applicability conditions match the current context and impulse. We assume each stored norm has:

* Content: a representation of the rule (e.g. “Do not lie” or “Eat healthy foods”).
* Activation condition: conditions or context tags when the norm is relevant (e.g. “when interacting with friends” or “when trying to stay healthy”).
* Priority metadata: such as a weight or importance value, an obligatoriness flag (distinguishing absolute duties vs. guidelines), and possibly tags like deontic type (prohibition, obligation, permission).
* Exceptions: any known exceptions or sub-conditions where the norm does not apply or is overridden.

Upon query, the Norm Repository yields a set of norms that are triggered. For each norm, it might also compute a relevance score depending on how well the context matches and how broad/narrow the norm is. For example, a very general norm (“Be kind”) might always be somewhat relevant, but a specific norm (“Do not eat cake on Tuesdays while dieting”) might only trigger if context exactly matches. The output is typically the set of applicable norms along with their weights/priority.

Output Format: The output is a list of (norm, weight, type, context\_match) for each applicable norm. For integration with the Comparator, these outputs are used to calculate the norm congruence N\_i for each impulse as described earlier. For integration with the Norm Conflict Resolver, the repository might specifically highlight if any norms directly conflict (e.g. one norm says “do X” another says “do not do X” in the given context).

Representation and Equations: We can formalize a simple model: Each norm j in the repository can be thought of as a predicate $norm\_j(context) \rightarrow (action, deontic, strength)$ indicating that in a given context, a certain action (or category of actions) is obligatory/forbidden/allowed with a certain strength. For example, norm “No stealing” might be represented as: if context contains property(ownership) and action == "take", then deontic = FORBID, strength = high. The repository’s query essentially filters norms by checking these conditions against the current context and impulse.

We can define the norm congruence contribution of a single norm j to impulse i as:

$$

congruence\_{ij} =

\begin{cases}

+1 \* \text{(norm weight}\_j) & \text{if norm $j$ prescribes impulse $i$ (obligatory or positively encouraged)}\

-1 \* \text{(norm weight}\_j) & \text{if norm $j$ forbids impulse $i$}\

0 & \text{if norm $j$ not applicable or neutral toward impulse $i$}

\end{cases}

$$

The Norm Repository essentially provides these congruence values (and weights) which the Comparator sums up. If a norm has an absolute veto nature (e.g. a deontic veto), it might also flag that for the Norm Conflict Resolver’s use.

Temporal Dynamics: In a single decision cycle, norm retrieval is assumed to be fast relative to the 100ms cycle (it could be considered happening in the initial phase of the cycle: when an impulse arises, immediately relevant norms are fetched). If context changes or a new impulse type appears mid-process, the repository can be queried again. Typically, the set of relevant norms remains constant throughout one decision on an impulse, unless the RAA triggers a shift (for instance, if initial resolution fails, the RAA might broaden the search to higher-level norms or meta-norms).

For now, we assume a static set per scenario, with updates to the repository happening between scenarios or when learning (discussed in Section II). The retrieval within one cycle does not modify the norms; it just reads them.

Pseudocode (Norm Repository retrieval):

# Norm Repository: Retrieve relevant norms for given context & impulse

function retrieve\_norms(context, impulse):

applicable\_norms = []

for norm in NormLibrary:

if matches(norm.condition, context, impulse):

# Calculate relevance weight

w = norm.priority\_weight

deontic = norm.type # e.g. "FORBID" or "OBLIGE"

applicable\_norms.append((norm, w, deontic))

return applicable\_norms

# Example usage:

norms = retrieve\_norms(current\_context, candidate\_impulse)

# Compute norm congruence score for impulse using returned norms

N = 0

for (norm, w, deontic) in norms:

if deontic == "FORBID":

N -= w # subtract weight if impulse violates a prohibition

elif deontic == "OBLIGE" or deontic == "ENCOURAGE":

N += w # add weight if impulse fulfills an obligation/recommendation

# (If "PERMIT" type norms, they might contribute 0 or a small positive if needed)

This pseudocode shows a simple loop checking each stored norm. In a real system, this would be optimized (using indexes or tags) rather than brute force. The output N would then be used in the Comparator’s drift. The matches function would evaluate context conditions (which could be logical expressions or pattern rules). For example, matches might check if impulse.action == norm.action or falls under its category, and if all required context features (like roles or time) align.

### **Norm Conflict Resolver Module: Resolving Conflicting Norms**

Function: The Norm Conflict Resolver is invoked when two or more relevant norms conflict in their prescriptions, creating ambiguity about what action to take. Its role is to algorithmically adjudicate between norms – essentially a mini decision-making system about norms themselves. It outputs a resolution that either prioritizes one norm over another or finds a compromise, according to a specified policy. The blueprint specifies weighted utility voting with deontic veto as the mechanism: this means each norm “votes” for or against the impulse with a weight (utility), but any norm that constitutes a hard veto (an absolute prohibition) can override the others if applicable.

Input Format: The inputs are the set of applicable norms (retrieved from the Norm Repository) that are in conflict regarding a particular impulse or decision. For instance, Norm A says “Oblige action X” while Norm B says “Forbid action X.” We assume the Norm Repository tags such conflicts or the RAA detects them. The input will include details of each norm: its weight (importance), its deontic stance (oblige/forbid), and possibly the context specifics. If multiple impulses are considered, conflict could also mean two impulses each supported by different norms that cannot both be satisfied (e.g. one norm says “Always tell the truth” supports impulse “confess truth”, another norm “Do no unnecessary harm” might support an impulse to withhold a hurtful truth – a direct conflict). The input might be structured as pairs or sets of norms in tension.

Processing Rules: The Norm Conflict Resolver applies a voting algorithm among the norms:

1. Weighted Voting: Treat each norm’s recommendation as a “vote” with strength equal to its weight or priority value. For a given impulse, sum up the weights of norms that favor allowing it (obligations or positive permissions) and sum up the weights of norms that oppose it (prohibitions). Compare the totals:  
   * Let W\_yes = Σ(weights of norms supporting action).
   * Let W\_no = Σ(weights of norms forbidding action).
2. Deontic Veto: Before finalizing the vote, check if any norm in the opposing set is marked as absolute (a deontic veto). A deontic veto norm is one that by its nature should not be violated regardless of utilities (for example, a moral rule like “Never kill an innocent” might be treated as absolute). If any such norm is present on the “No” side, then it can veto the action outright, irrespective of the numerical weights on “Yes”. In practice, this means if W\_no includes a norm with veto status, the resolver immediately returns “reject impulse” (unless perhaps an equally absolute norm on the “yes” side exists, leading to a stalemate that might require external resolution or meta-norms; such cases are rare and complex).
3. Resolution Outcome: If no veto is exercised, then compare W\_yes vs W\_no.  
   * If W\_yes > W\_no, the resolver leans towards allowing the action (norms in favor outweigh those against).
   * If W\_no > W\_yes, it leans towards rejecting the action.
   * If they are equal or nearly equal, that indicates a strong conflict deadlock. In such a tie, the RAA might then need to step in with a meta-decision or ask for more information (like a meta-norm could apply: for example, “in tie, prefer the more restrictive course (caution)” or “defer to a higher authority norm”).
4. Deontic Hierarchies: The weighted approach can be refined by recognizing norm types: obligations might carry different weight than recommendations. One way is to give obligations inherently higher weight ranges than mere advisories. Another is to implement lexicographic ordering: e.g., treat any “obligatory vs forbidden” conflict as higher importance than “recommended vs recommended” conflict. For simplicity here, we treat the weight as already reflective of importance, but in practice the Norm Repository could assign higher weights to obligations and lower to preferences.

The Norm Conflict Resolver essentially does a mini calculation of utility (where utility here is moral or rule-based utility, not just reward) for following each side. It is akin to how a person might weigh principles – e.g. honesty vs kindness – in a given situation. It can be related to multi-criteria decision algorithms or to defeasible deontic logic, where some norms can override others based on a hierarchy .

Output Format: The output is a decision or recommendation: e.g. “Allow impulse” or “Veto impulse”, possibly with a reason code (like which norm was decisive). This output is fed to the RAA and can directly influence the Assent Gate or Comparator:

* If the resolution is “Veto”, the RAA/Assent Gate can immediately enforce a “No” (e.g. set the evidence of that impulse to a large negative value or set threshold such that it can’t be reached).
* If the resolution is “Allow”, the system might lower the threshold for that impulse or boost its drift (to ensure it crosses the line).
* If it’s a tie or unclear, the RAA knows the conflict isn’t resolved and may require further recursion or external input.

Temporal Dynamics: The Norm Conflict Resolver’s operation is called when a conflict is detected, typically this would be during the deliberation if the RAA notices stagnation or contradiction. It might only need a few milliseconds to perform the calculation (since it’s just arithmetic over a handful of norms). It could be called once per conflict; if the context or weights change, it could be called again. For example, if after recursion one norm’s weight is adjusted (learning or reframing), a second call might yield a different result.

Pseudocode (Norm Conflict Resolver):

# Norm Conflict Resolver: Weighted voting with veto

# Input: list of norms (for current impulse) with each norm having (weight, stance, veto\_flag)

# Output: decision recommendation "ALLOW" or "REJECT" (or "TIE")

function resolve\_norm\_conflict(conflict\_norms):

total\_yes = 0

total\_no = 0

veto\_flag\_no = False

veto\_flag\_yes = False

for norm in conflict\_norms:

if norm.stance == "FORBID":

total\_no += norm.weight

if norm.veto\_flag == True:

veto\_flag\_no = True

elif norm.stance == "OBLIGE" or norm.stance == "ENCOURAGE":

total\_yes += norm.weight

if norm.veto\_flag == True:

veto\_flag\_yes = True

# (If stance == "PERMIT" meaning it allows but doesn't require, it might not add to either, or slight to yes.)

# Apply deontic veto logic

if veto\_flag\_no:

return "REJECT" # an absolute 'no' norm present, veto action

if veto\_flag\_yes and total\_no == 0:

# (If there's an absolute obligation and no forbidding norm, we should allow)

return "ALLOW"

# If no veto determined outcome, do weighted comparison

if total\_yes > total\_no:

return "ALLOW"

elif total\_no > total\_yes:

return "REJECT"

else:

return "TIE"

In this pseudocode, veto\_flag\_yes would rarely be used; it represents an absolute obligation (e.g. “You must do X”). If an absolute obligation norm is present and no opposing norms block it, we’d recommend allow (however, if an absolute obligation conflicts with an absolute prohibition, that’s a true dilemma – the code above would veto because it catches veto\_flag\_no first; a more nuanced approach might recognize that situation as unsolvable without external input, possibly returning “TIE” and letting the RAA handle it via meta-norm or external input). The “PERMIT” norms (which say an action is allowed but not required) typically don’t force anything; they might not need to appear in conflict resolution except to note that doing the action wouldn’t violate that norm.

Example: Imagine an agent considering whether to tell a friend a painful truth. Norms: N1: “Be honest” (obligation, weight 5), N2: “Do no harm” (obligation, weight 5). Telling the truth will hurt (minor harm) but is honest. N1 votes yes (with 5), N2 votes no (with 5). Neither has a veto flag (both are important but not absolute “never” – though one could argue “do no harm” might be near absolute, but assume not here). The resolver finds total\_yes = total\_no = 5, no veto. It returns “TIE”. The RAA seeing a tie might then look for a higher-order norm or context (maybe a meta-norm like “Honesty is the best policy in friendship” could break tie or the agent might delay the decision). If one norm had slightly higher weight, that side would win. If “do no harm” was marked as absolute (veto\_flag\_no True), then regardless of honesty’s weight, it would return “REJECT” telling the agent not to tell (because causing harm is absolutely forbidden in this hypothetical stance).

This Norm Conflict Resolver ensures that absolute prohibitions (deontic vetoes) are respected in the architecture, aligning with the idea of certain inviolable rules (deontological ethics), while otherwise using a utilitarian-like weighted sum for flexible decision-making . It also aligns with how humans often resolve norm conflicts by either identifying an overriding principle or by weighing the importance of each principle (e.g., deciding that in this case honesty is slightly more important than avoiding hurt feelings, etc.).

Interaction with RAA: The RAA triggers this resolver when needed and then acts on its output. If “REJECT,” the RAA might directly stop the impulse. If “ALLOW,” RAA might boost that impulse’s evidence to ensure it succeeds. If “TIE,” RAA may initiate further reflection (e.g., consider long-term consequences or seek additional input).

Flow Integration: Putting it all together in a flowchart-style summary:

1. Impulse arises → Query Norm Repository for relevant norms.
2. Norms returned → Comparator starts accumulating evidence (drift from salience, norm inputs, etc.) with current thresholds from Assent Gate.
3. If evidence crosses threshold → Decision (Yes or No) → proceed to act or inhibit.
4. If conflict indicators (contradictory norm inputs) or no decision within time:  
   * RAA engages Norm Conflict Resolver.
   * Resolver outputs decision or tie.
   * If decision given (yes or no), RAA biases system accordingly (e.g. force outcome or adjust parameters) and can terminate the loop with that outcome.
   * If tie/unsure, RAA may adjust parameters (urgency++, threshold changes, or seek a higher norm) and loops back to continue evidence accumulation.
5. Repeat accumulation with adjusted parameters → eventually reach a decision or hit a recursion limit. If recursion limit hit with no resolution, default to safest course (often No action).
6. Output final decision (with explanation if needed for logging).

This completes the algorithmic instantiation of each module and their interplay. Next, we expand on how the Norm Repository is developed and how norms are learned and updated over time.

## **II. Norm Repository Development and Learning**

In the initial blueprint, norms were treated as a static set of rules. We now extend the Norm Repository into a dynamic Norm Library that can grow and change through learning, and that can handle norm evolution, drift, and meta-evaluation. We define mechanisms for norm acquisition (how new norms are added), norm updating (how existing norms strengthen or weaken based on experience), and conflict management through meta-norms or priority tags. This makes the system adaptable and capable of normative development akin to a human learning social and moral rules over time .

### **Norm Acquisition Mechanisms**

An intelligent agent must be able to acquire new norms rather than only relying on pre-programmed ones. We propose multiple pathways for norm acquisition:

* Authority Input: The agent can receive explicit instructions or teachings from an authority or teacher figure. This could be a parent telling a child “Don’t hit others,” or a system admin updating a robot with a new rule. In our architecture, an authority input would directly add a new norm to the Norm Repository with a high initial weight (since authorities impart rules often considered important). For example, if an authority states a rule, the repository’s update function creates a new norm entry with appropriate context and weight. The context could be broad or specific depending on the instruction.
* Reinforcement and RPE Minimization: The agent can infer norms from reinforcement learning signals, particularly by minimizing Reward Prediction Error (RPE) in social contexts. For example, if the agent performs an action and gets unexpected negative feedback (punishment or social disapproval), that discrepancy (RPE) can prompt the agent to infer “ah, that action must violate a norm.” Over repeated experiences, the agent internalizes a norm that doing that action leads to bad outcomes, effectively creating a norm like rule. Computationally, this could work as follows: when an action leads to a large negative RPE, the agent checks if there is a known norm covering that situation. If not, it may hypothesize a new norm “maybe I shouldn’t do that.” It then stores that with some tentative weight. Conversely, positive surprises could indicate a praiseworthy action that might be a norm (“helping others yielded unexpected reward—perhaps ‘help others’ is a norm”). This aligns with models of social norm learning that involve reinforcement and internalization . Over time, as RPE is minimized (agent’s predictions of outcomes become accurate), it reflects that the agent has adjusted its internal norms to match the environment’s expectations.
* Bayesian Updating and Imitation: The agent can also observe others’ behavior and outcomes to update its norms. Using Bayesian inference, the agent can treat the presence of punishments or rewards in the environment as evidence for or against certain norms. For instance, if the agent observes others being punished for a behavior, it increases its belief that “that behavior is disallowed (norm exists against it).” The Norm Repository could maintain a belief distribution over possible norms and update these probabilities with evidence. A concrete example: Initially the agent might not know if a certain rule (“no loud noise in library”) exists. If it observes someone being scolded for loud noise in a library, it updates probability of that norm being in force to high and potentially adds it explicitly as a norm with some confidence weight. Bayesian norm learning can also account for context: the agent updates belief that “in context library, quiet is expected” .

These mechanisms ensure the Norm Repository is not static. For implementation, the Norm Repository would have an update function that can be called with new experiences:

* add\_norm(norm\_spec) to add a brand new norm (for authority inputs or discovered rules).
* update\_norm\_weight(norm\_id, delta) to strengthen or weaken an existing norm’s weight based on outcomes (for reinforcement signals).
* Possibly disable\_norm(norm\_id) if evidence suggests the norm no longer holds (e.g., if environment changes or authority rescinds it).

### **Norm Drift and Strengthening**

Norms are not binary present/absent; they have gradations of strength and can shift over time:

* Norm Strengthening: Each time a norm is followed and results in expected positive outcomes (or the agent observes broad compliance by others with positive outcomes), the norm’s weight can incrementally increase. This reflects internalization: the norm becomes more deeply ingrained. Psychologically, this corresponds to the agent valuing the norm more after seeing it confirmed or rewarded repeatedly . Implementation: if the agent chooses an action in line with a norm and receives reward or no punishment (outcome as expected), it can perform a small upward update to that norm’s weight (positive reinforcement). Similarly, if it violates a norm and experiences punishment, it will strengthen the weight of that norm (making it more salient in future decisions to avoid violation).
* Norm Drift (Weakening or Change): If a norm is repeatedly not enforced or following it yields no benefit (or yields harm), the agent may slowly reduce the weight of that norm. For example, if the agent has a norm “always yield to seniors” but finds in a new culture that this behavior is not expected and sometimes even counterproductive, over time the weight of that norm would drift down. Another form of drift is if society’s standards change: e.g., a norm that was taught strongly in the past might no longer be reinforced. The agent’s norm weight will gradually decay in the absence of reinforcement. Implementation: incorporate a small decay on norm weights over long times if not actively reinforced, or negative updates if outcomes contradict the norm (if obeying the norm leads to negative outcomes, the agent might question it).
* Conflict Emergence: As new norms are added or weights change, conflicts can emerge even if none existed initially. For instance, an agent might acquire a new norm that wasn’t considered before and find it clashes with an existing one. The system should detect when two high-weight norms frequently apply to the same context with opposite directives. This could trigger a norm conflict learning event, where the agent realizes it needs to prioritize one over the other. The Norm Conflict Resolver might handle individual decisions, but at a meta-level the agent might decide “I keep encountering conflict between Norm A and Norm B; I should establish a priority: e.g., decide Norm A > Norm B generally.” This could be stored as a meta-norm or a priority ranking in the repository (like an ordering: A outranks B).

To handle drift and conflicts, we introduce a priority tagging system in the norm metadata:

### **Priority Tagging System for Norms**

Each norm in the repository will have additional tags:

* Value Weight: a numerical weight indicating its current strength/importance (as discussed). This weight is dynamic.
* Obligatoriness (Deontic Strength): a categorical tag such as {Strict Obligation, Prohibition, Guideline, Preference}. Strict obligations/prohibitions might also be marked with veto\_flag=True (for Norm Conflict Resolver to know). Guidelines or preferences have veto\_flag=False and typically lower weight.
* Context scope: a list of contexts where it applies (could be hierarchical or specific).
* Source: how it was acquired (authority, learned from experience, etc.), which might influence how readily it’s changed (e.g., an authority-given norm might not drift easily unless authority gives new info).
* Last reinforced time: timestamp or count of when it was last strongly reinforced, for decay calculations.

Using these tags:

* The system can preferentially maintain certain norms: e.g., never decay strict obligations to below a certain floor weight unless a very strong reason.
* When conflicts arise often between a strict obligation and a guideline, the strict obligation’s priority tag ensures it usually wins (even aside from weight).
* Context activation can help avoid conflicts: some norms only activate in certain contexts, preventing false conflicts (the repository should ideally not list two norms as conflicting if their contexts differ, but the agent might in transitional contexts feel both apply—then context tags help clarify).

### **Recursive Reflection (Meta-Evaluation of Norms)**

Beyond just adjusting weights, the agent can engage in meta-cognitive evaluation of its norms. This is akin to a person reflecting “Is this rule really good? Maybe I should relax it in this case.” Our RAA can facilitate this by, for example:

* Noticing if following a norm leads to consistently bad outcomes or severe internal conflict. The RAA could flag that norm for review.
* The agent might consult a meta-norm: e.g., a principle like “prefer norms that promote overall well-being” or “if two norms conflict, prefer the more specific one” – these are second-order rules guiding how to choose between norms. We can include such meta-norms in the repository (they would guide priority assignment).

In practical terms, a meta-evaluation might result in norm revision: the agent could either modify the content of a norm (adding an exception) or demote one norm in the hierarchy. For example, if an agent has norms “Always tell the truth” and “Never hurt others’ feelings,” a meta-evaluation after several painful experiences might lead it to adopt an exception: “It’s okay to omit the truth to spare serious harm to feelings in some cases.” This new conditional norm (an exception) would be stored, effectively refining the rule set to reduce future conflict .

We can implement a simple mechanism: if two norms conflict frequently, introduce an exception in the lower-priority norm for cases where the higher-priority norm applies. E.g., add to “Never hurt others’ feelings” an exception “(unless honesty norm applies strongly).”

The architecture can periodically (or when triggered by a major conflict event) run a routine to adjust norm relationships:

* E.g., adjust\_norm\_hierarchy(normA, normB, outcome) that says if normA overrode normB in conflict, increase priority of normA relative to normB.

Pseudocode for Norm Updates:

# Norm Repository Update mechanism (called after each action or periodically)

function update\_norms(chosen\_action, context, outcome):

# outcome contains reward/punishment signals, possibly social feedback

applied\_norms = retrieve\_norms(context, chosen\_action)

for (norm, w, deontic) in applied\_norms:

if outcome.punishment and deontic == "FORBID":

# Violated a forbidden norm and got punished -> strengthen norm

norm.weight += lr \* positive\_update # lr: learning rate

if outcome.reward and deontic == "OBLIGE":

# Followed an obliged norm and got reward -> strengthen norm

norm.weight += lr \* positive\_update

if outcome.punishment and deontic == "OBLIGE":

# Followed an obliged norm but got punished (maybe norm is not correct or too strong)

norm.weight += lr \* negative\_update # perhaps decrease weight (norm might be too strict)

if outcome.reward and deontic == "FORBID":

# Violated a "forbid" norm but got rewarded or no bad outcome -> perhaps norm is less absolute

norm.weight += lr \* negative\_update # i.e., decrease weight (negative\_update is negative)

# Decay norms that were not involved

for norm in NormLibrary:

if norm not in applied\_norms and time\_since\_last\_update(norm) > decay\_time:

norm.weight \*= decay\_factor # slightly reduce weight

# Detect frequent conflicts

conflicts = detect\_common\_conflicts() # maybe track conflict occurrences

for (normX, normY) in conflicts:

if conflict\_count(normX, normY) > threshold:

# decide priority: e.g., if normX had higher utility outcomes historically than normY

if normX.weight > normY.weight:

normY.priority\_level = normY.priority\_level - 1 # lower priority or add exception

else:

normX.priority\_level = normX.priority\_level - 1

# (Alternatively, create a new exception rule norm)

In this pseudocode, positive\_update could be a fixed increment, and negative\_update a fixed decrement (or a fraction of current weight). detect\_common\_conflicts() would analyze recent decision logs to see which norms conflicted often. We adjust priority by maybe a separate field priority\_level or directly via weight or veto flags. (A more sophisticated approach might maintain a conflict matrix and use something like an algorithm to find a consistent ordering that minimizes conflict).

Norm Library Architecture: Internally, we can imagine the Norm Repository as containing layers or categories of norms (personal habits, social conventions, moral principles, etc.) possibly in a hierarchy. For fast lookup and update:

* It can be organized by context (like a tree: norms associated with specific contexts under general ones).
* Each norm entry holds its metadata as described.

### **Example of Norm Dynamics:**

Suppose the agent is in a new environment and initially has a norm “Always follow orders from supervisors” (from authority input) and it learns a norm “Be creative and take initiative” from observing that creativity is rewarded. At first, these might conflict (following orders vs taking initiative can conflict if initiative means bending rules). Through experiences:

* It might occasionally break an order to pursue a creative solution and if it gets praised (reward), the weight of “take initiative” increases relative to “follow orders.” If no punishment came for breaking the order, maybe “follow orders” weight drifts down a bit.
* However, if in another instance it breaks an order and gets reprimanded, that reinforces “follow orders” (increasing its weight) and maybe indicates an exception “don’t take initiative if it means disobey direct orders.” The Norm Conflict Resolver would have handled each instance, but over time the Norm Repository encodes the refined principle: follow orders is primary, but minor deviations for creativity might be tolerated unless expressly forbidden.

By developing the Norm Repository in this way, the agent gains a more nuanced, context-sensitive normative framework that can adapt. It mirrors human norm learning: initial strict rules that become more contextually nuanced with experience, internalizing social norms, and occasionally reforming one’s values through reflection .

Finally, the Norm Repository must support meta-norm queries: e.g., the RAA may ask “Is there a norm about how to resolve norm conflicts?” If yes, it might retrieve something like “In case of conflict, prioritize the one that prevents greater harm.” Such a meta-norm would then guide the Conflict Resolver or RAA’s choices. We ensure the repository can return such meta-norms when needed (they could be stored as norms with context = “when norms conflict”).

With this evolving norm library in place, the agent’s behavior is governed not by a fixed code but by an adaptive set of norms that grow and change – crucial for long-term autonomy and alignment with changing human values.

Now we will deepen the emotion and affect system which interplays with these norms and decisions.

## **III. Emotion and Affect System Deepening**

We now provide a more detailed model of the Emotion-Affect subsystem and how it interacts with decision-making. Emotions modulate decision parameters (drift rates, thresholds) and working memory, influencing how norms are applied. We distinguish different classes of emotions (hot vs cold vs masking) and define their effects formally, including their temporal decay profiles. This builds on psychological theories of hot/cool systems and mood-congruent cognition .

### **Emotion Classes and Their Decision Effects**

We classify emotions into three functional categories in this model:

* Hot Emotions: These are high-arousal, intense emotional states (e.g. anger, fear, excitement, panic) that tend to push for immediate action. Hot emotions usually increase drift rates in the Comparator (the urgency/salience is amplified) and lower the decision thresholds in the Assent Gate (reducing caution). In essence, a hot emotional state makes the agent more impulsive and action-biased – the cognitive system is saying “do something now.” For example, anger might dramatically increase the drift in favor of aggressive impulses and lower the threshold to act on them, unless checked by strong norms. Fear (acute fear/panic) might increase the urgency to escape, lowering thresholds for flight responses.
* Cold Emotions: These are low-arousal or subdued affective states and deliberate moods (e.g. sadness, calm contentment, contemplative mood). Cold emotions generally slow down drift rates (less push for immediate action) and raise decision thresholds (more evidence required) . A person in a calm or cautious mood will be more careful, allowing the Comparator to accumulate evidence longer and requiring stronger justification to act. For instance, sadness might cause indecision or lethargy, meaning even if an impulse arises, the evidence accumulates slowly and the threshold is high, so actions are delayed or deferred. A neutral or thoughtful mood similarly encourages waiting for more information.
* Masking Emotions: This category refers to emotional states that suppress or flatten emotional responses overall, often leading to indecision or a kind of emotional “numbness.” Examples might include shock (being stunned), confusion, or clinical states like dissociation or apathy. When a masking emotion is present, it can flatten the drift inputs – essentially reducing both positive and negative evidence signals, often by introducing noise or apathy – and can also distort threshold setting (sometimes leading to effectively very high thresholds because nothing feels compelling). In our model, a masking emotion adds a high noise term to the Comparator (making the evidence accumulation more random and less directed) and could either freeze thresholds at a moderate level or fluctuate them erratically. The net effect is that decisions become difficult; the agent neither strongly commits to action nor confidently rejects it. An everyday example is being in shock upon receiving bad news: one might feel “paralyzed” – which in our terms means the Comparator isn’t accumulating strongly toward any action, and the Assent Gate isn’t triggered because everything is blunted.

These categories align with psychology research on how affect influences control: e.g., Metcalfe & Mischel’s hot/cool framework describes a hot, “go” system that is emotional and impulsive, and a cool, “know” system that is rational and slow . Our “masking” category covers scenarios where emotional processing is disrupted (the person might feel nothing or be overwhelmed in a way that suppresses action, akin to a freeze response).

Quantitative Influence on Model Parameters:

Let’s formalize these influences:

* We introduce an emotion vector or state E = (E\_hot, E\_cold, E\_mask) with intensities for each aspect (one could also treat it as categorical, but sometimes multiple can overlap slightly; e.g., one can be both highly aroused and somewhat focused—though typically one dominates).
* The drift rate for impulse i we had as v\_i = w\_s S\_i + w\_n N\_i + w\_u U\_i. We now extend that: v\_i\_emotional = v\_i \* (1 + β\_hot \* E\_hot - β\_cold \* E\_cold) + noise\_mask, where:  
  + β\_hot and β\_cold are scaling factors representing how much a unit of hot or cold emotion affects drift. Hot emotion increases effective drift (if anger is high, evidence for a chosen course piles up faster, or evidence against is discounted).
  + β\_cold slows the drift (if in a cautious mood, the same impulse yields slower accumulation).
  + noise\_mask is an added noise term that is proportional to E\_mask (the intensity of masking emotion). For instance, if the agent is in shock (masking emotion strong), add a wide variance noise or effectively reduce signal-to-noise ratio of drift. This can be represented as increasing the noise variance σ in the drift-diffusion model: high E\_mask → high σ, meaning more random fluctuations in evidence .
* The threshold we had as Θ = Θ\_base + ΔΘ(serotonin, emotion). We refine ΔΘ for emotion as: ΔΘ\_emotion = -γ\_hot \* E\_hot + γ\_cold \* E\_cold + γ\_mask \* E\_mask. Here:  
  + γ\_hot (positive constant) means hot emotion subtracts from threshold (lowers it).
  + γ\_cold means cold emotion adds to threshold (raises it).
  + For γ\_mask, masking emotions might slightly raise threshold (if the agent is numb, they might not act without a very strong reason) or sometimes it could effectively freeze the threshold at a certain high value until mask subsides. We can model γ\_mask \* E\_mask as a positive addition (the person in shock requires extra strong signals to act).
* Additionally, emotions bias the urgency parameter: Hot emotions often come with a sense of urgency (especially panic or anger has a “act now” urgency). We can increase U\_i (urgency) when E\_hot is high. Cold emotions might decrease perceived urgency (nothing feels pressing). Masking might set urgency to zero or undefined (the agent might not feel any drive to act despite an objective need).

These modulations ensure that in a hot state, an impulse that normally might need, say, 1.0 evidence to act could effectively need only 0.7 (threshold lowered) and accumulate evidence 50% faster (drift increased), leading to quick action. In a cold state, the same impulse might require 1.3 evidence (threshold raised) and drift at half-speed, leading the agent to possibly not act at all if the impulse isn’t strong enough. In a masking state, the impulse’s evidence might wander randomly and threshold might be high, so likely no action until the state passes or some external input forces a decision.

### **Affect-Based Priority Modulation and Memory**

Emotions not only affect the “physics” of decision-making (drift/threshold) but also what information is considered and remembered:

* Emotion-Biased Norm Retrieval: Emotions can make certain norms more salient. This is analogous to mood-congruent memory in psychology, where a given mood makes related memories more easily recalled . In terms of norms, if someone is feeling compassionate (a warm empathetic state), norms related to care and helping others might be more readily activated from the repository, whereas punitive norms might be less forefront. Conversely, if someone is angry, norms about justice or retribution (“people should get what they deserve”) may become more salient, while norms about patience or forgiveness might recede momentarily. We model this by allowing the current emotional state to act as a filter or multiplier on norm weights during retrieval:  
  + For each norm, we can store an associated affective tag if applicable (e.g., norm “help others” might be positively tagged with empathy/compassion emotion).
  + During retrieval, if the agent’s emotion matches the norm’s tag (mood-congruent), we boost that norm’s weight . If it’s opposite, we might not retrieve it as strongly.
  + Example: In anger (a hot negative emotion), the norm “do not harm others” might unfortunately receive less weight in that moment, while “stand up for yourself” gets more weight. Our system could implement this by multiplying norm weights by a factor (1 + λ) if norm’s associated emotion is active, or (1 - λ) if a conflicting emotional state is present.
  + This helps replicate how emotional states bias moral judgment – e.g., empathy increases adherence to pro-social norms; anger might increase adherence to norms of justice/punishment but decrease ones requiring gentleness.
* Emotion Tags modulating Working Memory Persistence: Emotions influence what stays in working memory and for how long. Empirical studies show that emotionally charged items are remembered longer and given more attention, whereas irrelevant things are dropped . In our model, working memory holds the current goal, the impulse under consideration, and perhaps the norm context. A strong emotion can act as a tag that keeps certain thoughts active:  
  + If an impulse or a norm has an emotional tag, the RAA may carry it through more recursive cycles (persistence) rather than dropping it. For example, an insult (triggering anger) may keep resurfacing in working memory – the agent finds it hard to let it go, which means the impulse to retaliate persists across cycles (the decay is less because the emotional tag signals significance).
  + On the flip side, if the emotion is one of apathy or boredom (a cold disengaged emotion), even important tasks might slip from working memory quickly.
  + We can formalize: in the decay formula of evidence or in whether an unchosen impulse is dropped, include a factor for emotional salience. E.g., decay\_factor = base\_decay \* (1 - k \* emotional\_salience). Emotional salience high (like something that made you angry or excited) yields a decay\_factor closer to 1 (little decay, so memory persists). Emotional salience low (neutral or disengaged) yields a smaller factor (meaning memory decays faster).
  + Also, if an item in memory has an associated emotion (like a memory of a loved one evokes warmth), a corresponding mood can maintain it longer.
* Emotions also bias attention: a fearful state might make the agent highly attuned to threat-related cues (norm “ensure safety” gets priority). A happy state might broaden attention and allow more creative impulses.

These biases integrate with the rest of the system as follows: When the Norm Repository is queried, it can consider the current emotional state as part of context. When the RAA is deciding whether to continue thinking about an issue, it can check if it’s emotionally important (and thus maybe allow more cycles of recursion on it).

### **Emotion Decay Dynamics**

Emotions themselves have temporal dynamics. We model differences in how hot vs cold emotions decay:

* Hot Emotion Decay: Hot emotions typically have a rapid rise and relatively fast decay (they are acute). For instance, acute fear or anger might spike in seconds and subside within minutes once the stimulus is gone (though they can re-trigger). We can model a hot emotion E\_hot with a decay like dE\_hot/dt = -α \* E\_hot with a relatively large α, meaning a short half-life. If no new triggers, the intensity diminishes exponentially. Additionally, once a hot emotion triggers an action, it might reduce (e.g., venting anger can sometimes lower anger, though not always – depending on outcome). We might incorporate a mechanism that if the impulse associated with the emotion is acted upon or resolved, the emotion dampens.
* Cold Emotion Decay: Cold emotions (or moods) are longer-lasting states. A sad or calm mood can persist for hours or days with slow change. We model E\_cold with a much smaller decay constant (if any decay at all on short timescales). Cold emotions might instead shift due to context changes or deliberate mood regulation rather than simply decaying. For simulation, we can treat them as constant over the short term of a decision process, or slowly drifting baseline.
* Masking Emotion Decay: A masking emotion like shock or confusion often is short-term (shock fades as one processes what happened), but during its presence it might completely dominate. We can give it a medium decay – e.g., shock might last a few minutes to an hour, gradually the person “comes out of shock”. Apathy (if we consider that a masking emotion) could be longer-term, but that borders on a trait or disorder rather than an acute emotion.

We also consider cascading effects: A hot emotion can cool into a cold mood. Example: intense anger (hot) might, after subsiding, leave a lingering cold resentment or sadness. In the model, once E\_hot decays, the agent might update E\_cold to represent the leftover mood. We won’t delve too deep, but acknowledge that transitions can happen (the architecture could have rules: if a strong hot emotion episode occurred, afterwards set a cold mood reflecting the outcome – e.g., anger episode might leave one exhausted (cold, low mood) or content (if justice achieved), etc.)

Affect-Impose Interaction Summary: The combined formal model ties together as:

* The Comparator’s drift function v\_i(t) now includes emotional multipliers.
* The Assent Gate’s threshold Θ(t) includes emotional offsets.
* The Norm Repository retrieval retrieve\_norms(context, impulse, emotion) uses emotion to modulate norm weights.
* The Working memory persistence uses emotion to decide whether to maintain or drop items.

This yields a system where emotion and cognition are tightly interwoven. For example:

* In a Stroop-like task (color-word interference) if the subject is stressed (a hot state), they might have difficulty inhibiting the prepotent impulse (drift for reading the word is high, threshold lowered, so more errors). In a calm state, they do better (drift of impulse is tempered, threshold high, so they wait to respond until sure).
* In a moral dilemma, an empathetic emotional response might raise the threshold to harming someone and increase drift toward self-sacrifice action, whereas a fear-driven response might lower threshold to protect oneself (we’ll explore such scenarios in next section).

Formal Affect-Impulse Interaction Model:

To encapsulate, we provide a concise model:

Let D = base\_drift(impulse) from Section I (using salience, norm alignment, urgency).

Let Θ = base\_threshold(impulse) from Section I (with serotonin modulation but no emotion).

Then with emotion state E:

* drift\_final = D \* (1 + β\_hot \* E\_hot - β\_cold \* E\_cold) + noise(0, σ\_base^2 + σ\_mask^2 \* E\_mask).
* threshold\_final = Θ + (-γ\_hot \* E\_hot + γ\_cold \* E\_cold + γ\_mask \* E\_mask).  
    
   These equations can be adjusted per emotion type and calibrated. For instance, if E\_hot = 1 (max hot), maybe drift\_final ~ 150% of D, threshold\_final ~ 80% of Θ. If E\_cold = 1, drift\_final ~ 50% of D, threshold\_final ~ 120% of Θ. If E\_mask = 1, noise variance might double and threshold\_final ~ 110% of Θ.

Additionally:

* Norm weight w\_j\_final = w\_j \* (1 + λ \* emotion\_congruence\_j), where emotion\_congruence\_j = +1 if norm j aligns with current emotion, -1 if it opposes, 0 if neutral. For instance, a norm “protect others” aligns with compassion (if agent feels compassion, that norm weight gets a boost).
* Working memory item decay rate decay = decay\_base \* (1 - κ \* emotional\_importance\_item).

Empirical support for parts of this model comes from studies:

* Emotions modulate cognitive control and attention (e.g., negative affect impaired working memory capacity, positive affect enhanced it, also high arousal reduced capacity). This fits our assumption that high arousal (hot) can impair some executive function (like keeping multiple things in mind) – which in DDM terms could equate to focusing on one impulse strongly and dropping others.
* Mood-congruent recall is well-documented , justifying our norm retrieval bias.
* The hot/cool system theory provides a conceptual backing for our drift/threshold adjustments : the cool system corresponds to high thresholds and slower accumulation (requiring more evidence, strategic), the hot system to low threshold quick responses (under stimulus control).

Pseudocode Snippet for Emotion Effects:

# Integrate emotion into decision parameters (executed each cycle)

for impulse in impulses:

# Compute base drift and base threshold as before

base\_v = compute\_base\_drift(impulse)

base\_theta = base\_threshold

# Emotion adjustments

drift\_factor = 1 + beta\_hot \* emotion.hot\_level - beta\_cold \* emotion.cold\_level

threshold\_adjust = - gamma\_hot \* emotion.hot\_level + gamma\_cold \* emotion.cold\_level + gamma\_mask \* emotion.mask\_level

# Apply to drift and threshold

impulse.current\_drift = base\_v \* drift\_factor

impulse.current\_noise\_sigma = base\_noise\_sigma \* (1 + sigma\_mask\_factor \* emotion.mask\_level)

Theta = base\_theta + threshold\_adjust

# Norm weight modulation

for norm in impulse.relevant\_norms:

if norm.emotion\_tag matches emotion.state:

norm.effective\_weight = norm.weight \* (1 + lambda\_emotion\_boost)

elif norm.emotion\_tag is opposite to emotion.state:

norm.effective\_weight = norm.weight \* (1 - lambda\_emotion\_suppress)

else:

norm.effective\_weight = norm.weight

This shows qualitatively how we might implement the influence each cycle: adjusting drift, noise, threshold, and norm weights according to emotion.hot\_level, etc., which are updated by an emotion dynamics function.

Emotion dynamics update would be something like:

# Emotion decay each cycle

emotion.hot\_level \*= exp(-alpha\_hot \* Δt)

emotion.mask\_level \*= exp(-alpha\_mask \* Δt)

# cold\_level might remain or slowly drift to baseline

emotion.cold\_level += -alpha\_cold \* (emotion.cold\_level - baseline\_cold) \* Δt

(where baseline\_cold might be 0 or a default mood if we consider neutral baseline).

In summary, the affect system adds a layer of modulation that can drastically change the behavior of the governing faculty in different emotional contexts, allowing the blueprint to account for how people sometimes make choices that violate their usual norms or differ from rational expectations under emotional influence. We now proceed to see how this comprehensive model generates predictions in various behavioral paradigms.

## **IV. Experimental Predictions and Behavioral Simulations**

With the full model specified (modules + norms + affect), we can simulate its behavior in various canonical tasks and situations. Here we outline expected outcomes in several domains:

1. Cognitive control tasks (e.g., the Stroop task for inhibition),
2. Intertemporal choice tasks (e.g., delay discounting for impulsivity/self-control),
3. Moral dilemmas (where norm conflicts occur).

We will describe how the model would perform and any measurable predictions (reaction times, error rates, choices) it makes. We also discuss the role of serotonin modulation and norm conflicts in these predictions, and propose toy simulation setups to illustrate the behavior.

### **Simulation 1: Stroop Task (Inhibitory Control)**

Task Description: In a Stroop task, the agent sees color words (e.g., “RED”, “BLUE”) printed in ink of a different color, and must name the ink color, not read the word. The habitual impulse (reading the word) conflicts with the instructed norm (task rule: say the ink color, inhibit reading). This is a test of inhibitory control and conflict monitoring.

Model Setup:

* Impulse: “read the word” (strong, salient impulse due to years of reading habit).
* Norm: “follow task instruction: name ink color” (explicit rule given by authority/experimenter, high priority during the task).
* Comparator: will accumulate evidence for the two possible actions: speaking the word vs. speaking the color. The stimulus salience strongly favors the word (reading is automatic). Norm congruence strongly favors the color (because the experiment instruction norm says doing otherwise is wrong).
* Conflict: There is a direct conflict (one norm or goal vs an automatic impulse). The Norm Conflict Resolver in this case is basically not needed as such because one is a task rule (which might be treated as absolute during the task). But effectively, yes, Norm Conflict Resolver would say “norm (follow instruction) vetoes the impulse to read word.”

Predictions:

* Reaction Time (Latency): The model predicts slower decision times on incongruent Stroop trials (word and color different) than congruent trials (word and color same). Why? In incongruent trials, the impulse to read (salience-driven drift) is pushing evidence one way, while the norm (task rule) pushes evidence the other way (through norm congruence input). The Comparator experiences conflict, resulting in a lower net drift rate and requiring more accumulation to reach a threshold . The RAA may even detect the conflict and ensure an extra moment of processing (possibly raising threshold temporarily to avoid error, akin to cognitive control adjustments).
* In congruent trials, impulse and norm align (the word is the same as the color), so drift is high in one direction with no conflict, threshold is reached faster.
* This aligns with classic Stroop effects where incongruent conditions have longer reaction times.
* Error Rates: If the Assent Gate threshold is not sufficiently high or if serotonin modulation is low (meaning the agent is in a state of low inhibitory control), the model may occasionally let the wrong impulse cross threshold. For example, with low serotonin (which lowers thresholds) or high cognitive load, the impulse to read might reach threshold before the norm-based evidence can catch up, resulting in an error (reading the word aloud). The model predicts more errors when the agent’s Assent Gate threshold is lowered (which could correspond to manipulations like stress or low serotonin).
* If we simulate a scenario with a serotonin antagonist (reducing effective 5HT level), the model would lower thresholds and potentially drift might even increase if stress is induced, yielding more impulsive errors – the agent fails to inhibit the prepotent response. This is consistent with observations that serotonin enhancement improves impulse control , so conversely depletion would impair it.
* Neural/Process Interpretation: The model’s Comparator+Assent Gate in Stroop plays a role analogous to the ACC detecting conflict and the executive raising control. Our model would generate something like: on an incongruent trial, as soon as evidence starts accruing for the wrong response, the Norm Conflict Resolver/RAA might detect the conflict (word reading is not aligning with instructed norm) and effectively implement a control signal (which in our model could be raising threshold or even injecting a negative bias to drift for the disallowed response). This mirrors the conflict-monitoring theory where ACC signals trigger increased control to bias toward the task-relevant response .

Toy Simulation: We can simulate multiple trials:

* Congruent trial: salience for correct = 1, norm for correct = 1, drift ~ say 0.5; threshold 1.0. Time to reach threshold ~ 2 units.
* Incongruent trial: salience for wrong = 1, norm for correct = 1. Suppose we have two accumulators: one for “say color” and one for “say word.” The “say word” has high drift from salience, but Norm Conflict Resolver effectively gives it a veto (or our model biases against it). If threshold is momentarily high or drift of “say color” gradually wins, reaction might take, say, 3-4 units, and sometimes if threshold was too low, the wrong one triggers (error).
* We’d expect our agent to maybe occasionally err if not optimally tuned.

Serotonin Modulation: If we explicitly toggle 5HT\_level:

* Normal/high 5HT: threshold is higher, agent rarely errors, but slower perhaps (it waits for full evidence of correct answer).
* Low 5HT: threshold low, the agent is faster on some but at cost of more errors (impulsive reads word quickly). This matches how certain impulsive states or disorders behave – speed at cost of accuracy, consistent with drift-diffusion accounts where lowering threshold speeds response but increases errors .

### **Simulation 2: Delay Discounting (Intertemporal Choice)**

Task Description: The agent is offered a small reward now vs. a larger reward after a delay (e.g., $1 now or $2 in a week). Choosing the larger later reward requires self-control, as the immediate impulse is to take the reward now (immediate gratification). This tests impulsivity and how the model handles future-oriented norms (like “be patient for larger reward”).

Model Setup:

* Impulse: “Take the immediate reward” – this is driven by the salience of immediate gratification (and possibly a basic drive of reward-seeking with no wait).
* Norm/Value: The agent might have an internal norm or value of maximizing utility or a learned principle “waiting is better for bigger reward” or even an instruction in an experiment “try to maximize total earnings.” We can model this as a cognitive bias/norm that encourages choosing the larger later reward (a rational norm).
* The urgency or time factor: The immediate reward has effectively an urgency (available now), whereas the later reward is abstract/future. We incorporate that as well: the impulse for immediate option has high initial salience and urgency (the opportunity is right here).
* Additionally, serotonin is known to be involved in patience. A hypothesis often is that higher serotonin makes organisms more willing to wait (less impulsive choice).

Predictions:

* Preference Reversal with Serotonin: The model predicts that when serotonin levels are higher (Assent Gate threshold higher, impulse control stronger), the agent is more likely to wait for the larger reward. With low serotonin (thresholds low, or drift favoring immediate due to impatience), the agent more often takes the smaller-sooner reward. So if we simulate an agent under two conditions:  
  + Normal condition: moderately often chooses larger later.
  + Tryptophan-depletion (low 5HT) condition: more frequent choice of immediate reward (impatience). This correlates with empirical studies suggesting low serotonin promotes impulsive choice (though interestingly, some studies find serotonin affects action inhibition more than choice; but many models still assume a role in temporal discounting).
* Effect of Time Horizon: As the difference between now and later increases, or the reward ratio changes, the model’s outputs shift. For example, if later reward is only slightly larger, even a rational norm might not overcome the drift for immediate reward. If later reward is much larger, then the norm (maximize total) has more weight (like a utility calculation might favor waiting). The model can simulate various delay durations: as delay increases, the urgency for the immediate option stays high while the normative/utility appeal of the later maybe doesn’t rise proportionally, so beyond a certain delay the model consistently chooses immediate. This replicates hyperbolic discounting behavior where beyond some delay, waiting isn’t worth it to the agent.
* Latency of Decision: If the model is conflicted (values patience but feels temptation), it might have a longer decision time or might engage RAA recursion (“Are you sure you want to give up the bigger reward?” as a self-check). An impulsive agent (low threshold) might decide very quickly “take now” with minimal deliberation. So we’d see shorter deliberation times for the impulsive choices, longer for the patient choices, potentially. Some studies indeed observe that more impulsive individuals make quicker choices in these tasks.

Toy Simulation:

We can set up a simple comparison:

* Option A: reward = $X now.
* Option B: reward = $Y later (Y > X).  
    
   We translate that to evidence:  
  + Norm/Value evidence for Option B might be proportional to Y (or Y minus X, net gain) but discounted by a factor for delay (the agent’s cognitive understanding might discount future somewhat).
  + Salience/impulse evidence for Option A is high because it’s immediate (we could give it an urgency factor representing temporal discounting: the longer the delay for B, the more urgency pushes toward A).
* For a given scenario, if $Y is significantly bigger and delay not too huge, a rational agent picks B. Our model will do so if norm congruence (like “maximize reward” norm) weighted by difference overcomes the immediate salience of A. If not, it picks A.

We then simulate varying serotonin:

* If we simulate an agent with low 5HT (threshold low), even if B is somewhat better, the threshold to choose A is low so the agent often triggers the decision for A before fully considering B’s evidence. Essentially the agent satisfices with the immediate option quickly.
* With high 5HT (threshold high), the agent waits longer, allowing more evidence accumulation; eventually the cumulative drift favoring B (since rationally bigger reward yields more evidence over time) can win out, so the agent chooses B.  
    
   This matches the intuition that patience (waiting longer) allows the evidence for the long-term option to dominate, whereas impatience cuts the decision process short in favor of now.

Serotonin Modulation Effects: This directly answers a key question: serotonin modulation effects on impulsivity. Our model predicts:

* Increasing serotonin (through the Assent Gate raising threshold and possibly slightly reducing drift for immediate urges) reduces impulsive choice, biasing toward delayed rewards . The agent becomes more willing to accumulate evidence (i.e., essentially integrate the value of waiting fully).
* Decreasing serotonin does the opposite, increasing impulsive choice . Notably, some experimental findings indicate serotonin depletion didn’t always change delay discounting in humans , focusing more on action impulsivity, but in many theoretical accounts it is expected to have some effect. Our model posits it should, which could be tested by simulation parameter tweaks.
* Also, emotional state matters: A hot state (like stress or hunger) could increase urgency for immediate reward (simulate as higher U for Option A), leading to steeper discounting (preferring now even more). A cold, reflective state might help waiting.

### **Simulation 3: Moral Dilemmas (Norm Conflict Adjudication)**

Scenario Description: A classic moral dilemma: e.g., the trolley problem. The agent must choose whether to sacrifice one person to save five (pull a lever to redirect a train). This pits a deontological norm (“do not kill”) against a utilitarian norm (“save the greater number of lives”). It’s a case of norm conflict requiring resolution.

Model Setup:

* Impulse might not be as relevant here (unless emotional impulses like empathy for the five or one). Instead, this is a conflict between two strong norms:  
  + Norm1: “Killing is forbidden (even as a means)” – deontic veto type, high weight.
  + Norm2: “Saving lives / minimize harm” – high weight utilitarian norm.
* The scenario activates both norms. The Norm Conflict Resolver is squarely in play, with Norm1 likely marked as a veto (especially in a personal dilemma scenario).
* Emotional influences: typically, personal harm dilemmas elicit strong aversive emotional response to the idea of harming someone (hot emotion like anxiety, empathy for the victim), which would further reinforce the “don’t kill” norm (or raise threshold to take that harmful action).
* The RAA might simulate internal debate: go through recursion if needed.

Predictions:

* Decision Outcome: The model, as configured, would likely refuse to push the man (in the footbridge variant) because Norm “don’t kill” has veto status. The Norm Conflict Resolver would see veto\_flag\_no from the deontic norm and output “REJECT action” – meaning do not take the action that kills one (thus the five die in that scenario). This corresponds to a deontological choice.
* If it’s the switch version (less personal), some models say people are more utilitarian there. In our model, the difference might be that the action of pulling a switch is less emotionally salient as “killing” (the norm “don’t kill” might be perceived slightly differently, or the emotion attached is lower), possibly not triggering a full veto (the agent might mark it as less personal, hence treat it as a more acceptable violation). If the veto flag is off (because perhaps it’s indirect), then the weighted voting happens: Norm2 (save five) weight might outweigh Norm1 (don’t cause one death) if Norm1 is no longer absolute. Then the model would allow the lever pull. This aligns with many people’s responses: they will pull a lever but not push a person, reflecting the presence or absence of an absolute constraint feeling.
* Decision Latency: In either case, the model would likely have a longer deliberation time than a straightforward scenario. The conflict means evidence accumulation is contested. The RAA might do multiple cycles: first leaning one way, then reconsidering (“But five people… but I’d be killing one…”). This extended recursion is predicted to manifest as a longer reaction time or hesitation. Humans indeed take longer on high-conflict moral dilemmas compared to low-conflict ones .
* If we had to measure “recursion frequency” (how many cycles the model goes through), a high-conflict dilemma (like footbridge) would cause more RAA cycles than a clear-cut scenario (like an easy choice where norms align). So norm conflict increases decision latency and recursion in the model. This could be analogous to increased brain activity in control regions during moral conflict, or simply more self-reported difficulty.
* Effect of Emotional Modulation: If we dial emotion settings:  
  + An extremely empathetic state (imagine the agent vividly empathizes with the one person as well as the five) might raise threshold to do any harm, making it even less likely to sacrifice one. If the agent were made very cold/calculating (suppressing emotion, say a “Spock-like” mode), then the weights might be considered more objectively and perhaps the utilitarian calculation wins (especially if no veto). Our model can simulate an emotionless version by setting E\_hot = E\_cold = 0, just use weights: it might then choose the utilitarian outcome if weights favor it. With normal emotional aversion, it chooses deontological.
  + This aligns with Greene’s dual-process theory: emotional response (deontological) vs cognitive evaluation (utilitarian) . Our Norm1 corresponds to the emotional/deontic side, Norm2 to the utilitarian side. In high-conflict personal dilemmas, emotional (deontic) tends to dominate (veto), causing longer RT if utilitarian reasoning is pushing opposite – exactly what our model does by needing the Norm Conflict Resolver to apply veto and RAA to take time.
* Variation: If we simulate a different moral dilemma, say “steal medicine to save a life” (norm conflict between property rights vs saving a life):  
  + The “do not steal” norm might be strong but perhaps not absolute veto like killing (depending on agent’s values). The “save a life” norm is strong. The resolver might find weights – many might say saving a life outweighs property norm. Our model could be tuned to reflect common moral intuitions by adjusting veto flags or weights of particular norms.
  + The prediction then is some dilemmas the model resolves in favor of utilitarian (especially if the inhibited norm is not absolute), and some it refuses (if a veto norm is present).

Toy Simulation:

* Set Norm1 weight = 9, veto=True (don’t murder).
* Set Norm2 weight = 8 (save lives).
* Conflict Resolver: sees veto on Norm1 -> outcome “## V. Theoretical Integration and Homunculus Rebuttal

Having detailed the blueprint’s mechanisms, we now situate this model in the landscape of cognitive science and philosophy, addressing potential criticisms (like the “homunculus problem”) and drawing parallels to established theories. We show that our “Normative Executive System” (NES, corresponding to the RAA+Assent Gate combination) is an emergent, distributed control process – not a mysterious little man in the head – and we map each component to concepts in contemporary frameworks, from conflict monitoring and cognitive control to global workspace and attention-schema theories.

### **Addressing the Homunculus Critique**

A common critique of models positing an internal “decider” or executive is that they risk simply creating a smaller homunculus (little person) inside the mind making decisions, leading to infinite regress. Our model avoids this by specifying mechanistic processes for each executive function, thus “breaking down the homunculus into an army of simpler processes” . Rather than a blank check “ruler” that magically approves impulses, we have concrete elements: a comparator implementing decision evidence, a threshold modulated by neurotransmitter-like signals, and a conflict resolver following explicit rules. As Verbruggen et al. note, many older theories used ill-defined homunculi for functions like “inhibition” or “updating” without explaining how . We combat that by providing a drift-diffusion mechanism for inhibitory control and a voting mechanism for norm arbitration, explaining how these functions occur (through accumulation and weighted veto) rather than simply labeling a box “inhibition homunculus.”

In essence, the executive in our model (NES/RAL) is not a single actor but an emergent result of these interacting modules. It leverages what Monsell & Driver metaphorically suggested: to banish a homunculus, organize “armies of such idiots to do the work” . Our “idiot” processes are the simple mathematical accumulators, threshold comparators, and rule-checkers, each of which on its own is dumb (e.g., an accumulator just follows a formula), but in concert they produce intelligent control. The recursion in RAA is strictly limited and rule-governed (it doesn’t lead to infinite regress because it stops after conflict resolution or a set number of cycles, at which point a decision is made or defaulted). There is no second hidden homunculus deciding when RAA stops; that is determined by the model’s parameters (e.g., urgency or a maximal cycle count) – essentially by another layer of simple logic (“if deliberation exceeds N cycles, abort”). This kind of layered control is common in engineered systems and doesn’t imply an infinite regress, it’s a hierarchical policy.

Furthermore, the NES can be thought of as implementing a form of self-monitoring without invoking a mysterious self. It monitors conflict signals (like ACC in the brain) and adjusts accordingly, akin to a thermostat regulating temperature – a clear feedback system, not an intelligent ghost. This ties into cognitive architectures like Norman & Shallice’s Supervisory Attention System, where a higher-level system biases lower-level actions when routine processes falter, but Norman & Shallice explicitly argued this need not be a homunculus if its operations are well-specified (which we have done).

In summary, the “governing faculty” is not a single centralized ghost, but a functional layer emerging from interactions of defined sub-modules. Each sub-module can in principle be implemented by neural circuits or computational routines, eliminating the need to postulate an inexplicable inner CEO. By detailing the algorithms (as we have with pseudocode and equations), we make the governing faculty transparent and mechanistic. This directly responds to the homunculus argument that earlier conceptual models faced – our blueprint provides a blueprint (in engineering terms) for the so-called homunculus, thereby explaining it in terms of smaller parts. The result is a model of executive control that is distributed across processes and recursive loops rather than a singular entity, consistent with modern theories that advocate for distributed cognition .

### **Alignment with Cognitive and Neuroscience Theories**

The table below provides a crosswalk between our blueprint components and concepts from major theories in cognitive science and philosophy, showing that our model is well-grounded in existing frameworks:

| **Blueprint Component** | **Analogous Theoretical Constructs & Findings** |
| --- | --- |
| Comparator (Drift-Diffusion) | Conflict Monitor (ACC): Detects response conflict and need for control , similar to how our Comparator registers norm-vs-impulse conflict via stalled drift. Evidence Accumulator: Implements a decision process akin to the drift-diffusion model (DDM) widely used for 2-choice decisions . The idea of accumulating evidence to a threshold maps onto neural models of decision-making (e.g., integrator neurons in parietal cortex). |
| Assent Gate (Threshold Gating) | Response Inhibition & Thresholding: Corresponds to the adjustable decision threshold (boundary separation) in DDM models, which reflects speed-accuracy tradeoff control . Neurologically, this relates to mechanisms in the subthalamic nucleus and PFC that raise decision thresholds under conflict or caution . Serotonergic Modulation: The effect of a serotonin-like signal on raising thresholds mirrors findings that increasing serotonin promotes behavioral inhibition and that low 5-HT leads to impulsive (premature) responding . In essence, our Assent Gate embodies the cognitive effect of neuromodulators on decision caution. |
| Norm Repository (Norm Library) | Long-term Memory & Schema: Functions like a semantic memory for rules and values. It aligns with psychological concepts of internalized social norms and Piaget/Kohlberg’s stages of moral internalization. Each norm is like a schema that can be activated by context (cf. scripts or frames in cognitive schemas). Value System (Superego analogy): Philosophically, the Norm Repository plays a role akin to the superego or an internalized moral code that guides the ego’s decisions, but in our model it’s updated by learning (authority, reinforcement) rather than fixed. This component also resonates with the Global Workspace Theory in that norms are part of the information that competes to shape the global decision – only the norms relevant (winning the competition) enter the “workspace” (i.e., influence comparator drift) . |
| Norm Conflict Resolver | Cognitive Control Arbitration: Relates to models of decision-making where multiple goals or constraints must be reconciled. Our weighted voting and veto is analogous to a cost-benefit analysis in the anterior cingulate cortex, which evaluates the value of exerting control vs. not . When it applies a veto, it’s like the brain saying “that action is off-limits” akin to how certain prepotent responses are completely suppressed in go/no-go tasks. Defeasible Deontic Logic: Technically, our conflict resolver echoes AI models of normative reasoning that allow exceptions and prioritization . It implements a simplified lexicographic ordering of norms (veto = lexically highest priority) which is discussed in ethical theories (e.g., lex talionis vs. utilitarian calculus debates). So it bridges rule-based (deontological) and utilitarian decision principles in a single mechanism. |
| RAA / Recursive Adjudication Layer (NES) | Supervisory Executive / Meta-cognitive Control: This corresponds to the notion of a central executive in cognitive theories (Baddeley) or the Supervisory Attentional System that intervenes in non-routine situations. It also parallels the Expected Value of Control (EVC) model by Shenhav et al., where the anterior cingulate cortex allocates control based on evaluating the payoff vs. effort . Our RAA effectively decides whether to exert additional control (do another cycle, enforce a norm) when conflict is detected, consistent with ACC’s proposed role in deciding “whether and how much control to allocate” . Global Workspace & Attention Schema: The RAA/NES can be seen as implementing a global workspace function – it takes inputs from various subsystems (percepts, norms, emotions) and ensures a coherent decision is broadcast (the chosen impulse). It models aspects of attention – e.g., it “pays attention” to conflicts and “shifts attention” to norms when needed. This resonates with Graziano’s Attention Schema Theory: the system has a simplified model of its own decision process (knowing when it is in conflict or not) , which it uses to regulate itself. In our case, RAA’s monitoring of evidence and conflicts is like an attention schema for internal control signals, enabling the system to adjust its focus (e.g., focus on norm considerations when impulse vs. norm conflict arises). |

(Sources: conflict monitoring ; drift diffusion and threshold ; serotonin and inhibition ; norm internalization ; global workspace ; deontic logic ; ACC/EVC ; attention schema .)

As shown above, our model incorporates elements of both dual-process theories (e.g., intuitive vs. deliberative processes in moral judgment ) and integrative theories (like EVC). For instance, in moral dilemmas our Norm Conflict Resolver’s behavior reflects the dual-process account: the deontological norm might act as an automatic veto (intuitive/emotional response), whereas the utilitarian norm adds up evidence for the opposite choice (deliberative calculation) . The outcome (and the latency) then depends on their interplay, which maps to Greene et al.’s findings that personal moral dilemmas engage emotional systems leading to longer RTs when pitted against utilitarian reasoning . Our architecture reproduces that dynamic: it will take longer and possibly require recursive runs when a strong “veto” norm conflicts with a strong utilitarian norm, mirroring empirical RT patterns .

### **Reframing the**

### **Hegemonikon**

### **as NES**

It is worth noting the philosophical lineage of the idea of a governing faculty. The Stoic philosophers used the term hegemonikon (Greek for “ruling principle”) to denote the rational executive of the soul that receives impressions and gives or withholds assent to impulses, resulting in action or suppression. This is remarkably analogous to our model: the Assent Gate literally implements assent to impulses, and the RAA/NES is the rational overseeing entity. However, whereas the Stoics conceived the hegemonikon as a single, unitary faculty (sometimes even likened to a fragment of the divine in us) , we reconceptualize it in modern, mechanistic terms as a Recursive Adjudication Layer. The NES performs the role of the hegemonikon – integrating perception, applying judgment (via norms), and controlling action – but does so through computational processes rather than an indivisible soul.

By calling it a Normative Executive System we emphasize that it’s a system: composed of sub-components (comparator, gate, conflict resolver) operating in concert. This dovetails with the Stoic idea that the mind is unified and rational , yet we allow that “rationality” to emerge from structured interactions rather than assume it as an irreducible given. In effect, we have built a model that fulfills the Stoic picture of decision-making (impressions -> assent by ruling faculty -> action) using tools of cognitive science. The homunculus critique did not trouble the Stoics because they were comfortable with a metaphysical soul; we, in contrast, have engineered the hegemonikon such that it can be implemented in a brain or AI, thereby demystifying it.

In doing so, we also provide a platform to explore conscious experience aspects: In Stoic theory, when the hegemonikon gives assent, that is an act of will aligning with reason. In our model, that moment corresponds to the Assent Gate threshold being hit for a norm-congruent action, which could be associated with a conscious commitment to the decision. The Global Workspace Theory connection suggests that when the NES “approves” an action, that content (the chosen impulse and rationale) enters the global workspace – in other words, it becomes the content of conscious thought leading to action . Thus, one could say the NES is the functional reality behind the intuitive notion of a will or executive ego.

### **Preemptive Rebuttal Summary**

To summarize our theoretical positioning: the Governing Faculty Blueprint is not positing a soul or homunculus, but a control system consistent with cognitive neuroscience principles. It stands on the shoulders of prior models: the conflict monitoring of ACC (we explicitly model conflict signals), the expected value of control (our RAA chooses when to loop or not, effectively a cost-benefit of more deliberation), global workspace (our decision result is globally broadcast, and attention/working memory are influenced by emotional salience as in workspace theory), and attention schema (the NES has a simplified internal model of its decision process state, akin to an attention schema, enabling self-regulation) . By integrating these, the blueprint aims to satisfy both functional and phenomenological accounts: functionally, it explains control and decision-making; phenomenologically, it could be extended to explain why it feels like we have a central self deciding – because the brain’s attention schema might be modeling this very NES as “me” making the choice.

In conclusion, the Governing Faculty Blueprint provides a coherent, mechanistically detailed account of the agent’s executive control over impulses, in harmony with major cognitive science theories and free of the logical fallacies that plague homunculus-style explanations. It reframes the ancient idea of a governing self into a modern control architecture: the Normative Executive System, an emergent ruler created not by fiat, but by the orderly interaction of computational elements bound by normative and affective logic. This “ruling system” can be implemented, tested, and refined – a significant step toward understanding and engineering autonomous moral cognition.