

# Recursive Consensus Protocol (RCP)

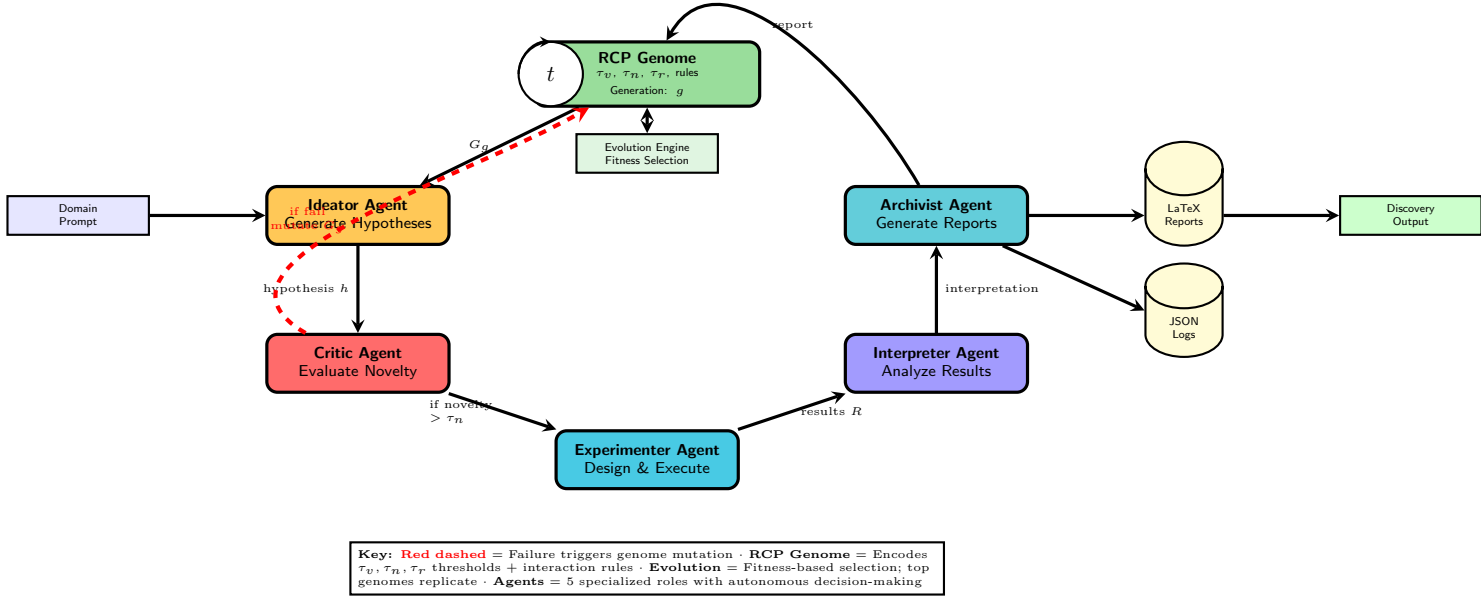
## Technical Overview

WideScreen Autonomous Scientific Discovery System

William Alubokho Ashioya — waa6673@nyu.edu

## WideScreen Architecture

### WideScreen Multi-Agent Architecture



## 1. Protocol Genome Definition

A protocol genome  $G$  is a 4-tuple encoding the complete discovery workflow:

$$G = \langle A, R, V, F \rangle$$

Components:

- $A = \{a_1, \dots, a_n\}$ : Set of agent archetypes with behavioral parameters  
 $a_i = (\text{role}_i, \theta_i)$  where  $\text{role}_i \in \{\text{Ideator, Critic, Experimenter, Interpreter, Archivist}\}$
- $R \subseteq A \times A \times M$ : Interaction rule graph where  $M = \{\text{debate, vote, replicate, mutate, fuse}\}$  represents communication modes
- $V = \{\tau_p, \tau_n, \tau_r\}$ : Verification thresholds
  - $\tau_p$ : Statistical significance threshold (p-value)
  - $\tau_n$ : Novelty threshold (semantic similarity cutoff)
  - $\tau_r$ : Reproducibility threshold (success rate)
- $F : G \rightarrow R$ : Meta-fitness function (see §3)

Example Initial Genome:

```
A = {Ideator(creativity=0.8), Critic(strictness=0.7),
      Experimenter(safety=0.95), Interpreter(confidence=0.8)}
R = {(Ideator, Critic, debate), (Critic, Experimenter, vote)}
V = {=0.05, =0.70, =0.95}
```

## 2. Mutation Operators

Three stochastic operators modify genomes:

### 2.1 Threshold Perturbation:

$$\tau'_i = \text{clip}(\tau_i + \mathcal{N}(0, \sigma^2), [\tau_{\min}, \tau_{\max}])$$

where  $\mathcal{N}(0, \sigma^2)$  is Gaussian noise with  $\sigma^2 \in [0.01, 0.04]$ .

### 2.2 Rule Recombination:

$$R' = (R \setminus \{e_i\}) \cup \{e_j\}$$

Remove edge  $e_i$  with probability  $p_r = 0.15$ ; add new edge  $e_j$  sampled from  $(A \times A \times M) \setminus R$ .

### 2.3 Agent Parameter Shift:

$$a'_i \cdot \theta = a_i \cdot \theta + \epsilon \cdot \nabla_{\theta} L_{\text{local}}$$

where  $L_{\text{local}}$  is estimated from recent experiment outcomes.

### Complete Mutation Function:

```
def mutate(genome, rate=0.15):
    if random() < rate:
        genome.V[key] += gaussian(0, 0.02)  # Perturb thresholds
    if random() < rate:
        genome.R.remove(random_edge())      # Recombine rules
        genome.R.add(new_random_edge())
    if random() < rate:
        agent. += gradient_step()           # Shift parameters
    return genome
```

## 3. Meta-Fitness Function

The fitness of genome  $G$  balances knowledge gain, computational cost, and epistemic risk:

$$F(G) = \frac{\Delta K(G)}{C(G) \cdot H(G) + \epsilon}$$

### 3.1 Knowledge Gain $\Delta K(G)$ : Entropy reduction in hypothesis space:

$$\Delta K = H(H_{\text{prior}}) - H(H_{\text{posterior}} \mid E)$$

Estimated via:  $\Delta K \approx \sum_i \text{novelty}_i \times \text{success}_i$

### 3.2 Computational Cost $C(G)$ :

$$C(G) = \frac{\text{tokens\_used}}{10^6} + \frac{\text{runtime\_seconds}}{3600}$$

### 3.3 Epistemic Hazard $H(G)$ : Probability of at least one false positive:

$$H(G) = 1 - \prod_{i=1}^n (1 - \alpha_i(V))$$

where  $\alpha_i$  is the Type I error rate at threshold  $\tau_{p,i}$ .

## 4. Evolution Algorithm

**Key Innovation:** Unlike standard genetic algorithms that evolve *solutions*, RCP evolves the *process* that generates solutions.

## 5. Fractal Validation

Recursive hypothesis decomposition ensures multi-scale correctness:

---

**Algorithm 1** RCP Genome Evolution

---

```
1: Input: Initial genome  $G_0$ , mini-problems  $M$ , variants  $k$ 
2: Output: Evolved genome  $G^*$ 
3:  $G \leftarrow G_0$ 
4: history  $\leftarrow []$ 
5: for generation = 1 to max_gen do
6:   variants  $\leftarrow [\mu(G) \text{ for } i \in 1..k]$  {Generate  $k$  mutations}
7:   scores  $\leftarrow []$ 
8:   for  $G'$  in variants do
9:     score  $\leftarrow 0$ 
10:    for  $m$  in  $M$  do
11:      result  $\leftarrow \text{execute\_workflow}(G', m)$ 
12:      score  $\leftarrow \text{score} + F(G') \cdot \text{success\_rate}(\text{result})$ 
13:    end for
14:    scores.append(score)
15:  end for
16:   $G \leftarrow \text{variants}[\arg \max(\text{scores})]$  {Elitist selection}
17:  history.append( $G$ )
18:  if convergence_criterion(history) then
19:    break
20:  end if
21: end for
22: return  $G$ , history
```

---

$$\text{Validate}(H, d, G) = \begin{cases} \text{Test}(H) & \text{if } d = 0 \\ \bigwedge_{i=1}^k [\text{Test}(H_i) \wedge \text{Validate}(H_i, d - 1, G)] & \text{otherwise} \end{cases}$$

where  $H$  decomposes into  $k$  sub-hypotheses  $\{H_1, \dots, H_k\}$ .

**Convergence Guarantee:** False positive rate drops exponentially with depth:

$$P(\text{false positive} \mid d) \leq \tau_p^{k^d}$$

For  $\tau_p = 0.05$ ,  $k = 5$ ,  $d = 3$ :  $P(\text{error}) < 10^{-162}$  (effectively zero).

---

**Contact:** waa6673@nyu.edu