ChatGPT

Sobreposición de Capas y Pensamiento en MoA para un sistema homeostático-estratificado

Hallazgos nucleares sobre colaboración en tiempo de inferencia

Tu intuición de "no aislar expertos" tiene un análogo técnico muy directo: **colaboración a nivel de token** durante el *decoding*, donde la decisión "¿quién genera el próximo token?" no es fija ni por turnos, sino una **variable latente** que se resuelve dinámicamente en inferencia (o se aprende para inferencia). El paper más alineado con lo que describes ("Experto A genera mientras B/C validan o intervienen") es **Learning to Decode Collaboratively with Multiple Language Models** (ACL 2024): propone enseñar a varios LLMs a colaborar **intercalando sus generaciones token a token**, modelando la elección del LLM que emite el siguiente token como variable latente; el modelo base aprende **cuándo generar por sí mismo y cuándo "llamar" a un modelo asistente experto** sin supervisión directa. [1]

Esto es importante porque desplaza el problema desde "pipeline multiagente" (secuencial/paralelo por turnos) hacia algo más cercano a tu "**sobreposición**": un **sistema de decodificación conjunta** donde, en cada paso, varios módulos influyen el resultado.

Un segundo bloque de evidencia clave es que hoy existen técnicas robustas para hacer que **memoria y seguridad actúen como sesgos latentes durante la generación**, no como filtros post-hoc, combinando distribuciones de probabilidad/logits en tiempo real:

- **DExperts** (ACL 2021) formaliza el control en decodificación como un **product of experts**: combinando un LM base con LMs "expertos" y/o "anti-expertos", de modo que un token recibe alta probabilidad solo si es consistente con la(s) experticia(s) y penalizado por anti-expertos (p.ej., detox). [2]
- **GeDi** guía la generación con un "discriminador generativo" (un LM pequeño entrenado condicionalmente) para reponderar tokens hacia atributos deseados (p.ej., menos toxicidad), **en cada paso de decodificación**. [3]
- **FUDGE** aprende un predictor del atributo en secuencias parciales y usa esa señal para **ajustar los logits del generador** (condicionamiento modular, componible). [4]
- Para memoria como sesgo "latente" real (más allá de RAG/prompt), **kNN-LM** interpola la distribución del LM paramétrico con una distribución basada en *nearest neighbors* en un *datastore* externo, mejorando generación mediante **memorization at inference-time** (la memoria literalmente entra como probabilidad de siguiente token). [5]

En conjunto, estos resultados soportan la plausibilidad de tu hipótesis "no sumar votos, sino **capas de restricción** que condicionan logits": ya hay evidencia de *logit mixing* y *product-of-experts* aplicados a control y safety, y de memoria como componente probabilístico del decodificador.

Inter-querying durante CoT y verificación “en línea” sin romper el flujo

Inter-querying como patrón de “razonar–actuar–consultar”

La forma más establecida (y portable a tu caso) de “pausar la generación para consultar algo y continuar” es el patrón tipo **ReAct**, que intercala trazas de razonamiento con acciones (p.ej., consultas a Wikipedia/herramientas) para reducir alucinación y mejorar planificación. Aunque ReAct consulta herramientas, el patrón se traslada directamente a “consultar otro modelo experto”: el *call* a un experto es una “acción” dentro del loop. [6]

A nivel de *framework*, **AutoGen** está diseñado precisamente para definir conversaciones multiagente y patrones flexibles de interacción (turnos, herramientas, humanos, etc.), por lo que implementa naturalmente “micro-consultas” a expertos en medio del razonamiento. [7]

Verificación paso a paso como “subconsciente” evaluador

Si lo que quieres es que B/C no solo respondan cuando se les pregunta, sino que estén **evaluando la coherencia del razonamiento a medida que se produce**, hay una línea muy fuerte: **process supervision y process reward models (PRMs)**.

- **Let’s Verify Step by Step** (2023) contrasta supervisión por resultado versus supervisión por proceso: los **PRMs** reciben feedback por cada paso del razonamiento, lo que mejora localización de errores y la fiabilidad del razonamiento largo. [8]
- En tu arquitectura, esto encaja como: “Experto generador” produce pasos; “capa estructural” (seguridad/memoria/reglas) opera como **verificador por pasos** que puntúa o frena cuando detecta incoherencias.
- Para factualidad/hallucination, **Chain-of-Verification (CoVe)** usa un loop donde el modelo primero redacta, luego genera preguntas de verificación y contesta esas preguntas de manera separada para evitar sesgo, y finalmente reescribe con verificación. Es secuencial, pero es evidencia clara de “inter-querying” interno para validar premisas. [9]

Más cerca aún de tu “validación en tiempo real” es la idea de **decoding con auto-verificación durante la generación**: - **DSVD (Dynamic Self-Verify Decoding)** reporta explícitamente que la verificación “real-time” durante generación puede mejorar fidelidad y combinarse con otros métodos de *faithful decoding*. [10]

“Background inference” práctico: streaming + verificador + backtracking

En un transformer estándar no existe un “subproceso” escondido dentro del forward pass, pero sí puedes implementar **equivalentes funcionales** de background inference a nivel de sistema:

- **Verificación en streaming**: mientras el generador emite tokens (stream), un verificador consume el prefijo y evalúa consistencia/seguridad. DSVD formaliza precisamente esa filosofía (verificar durante generación). [10]
- **Backtracking/búsqueda**: en vez de comprometerte a una sola trayectoria, usas búsqueda sobre “unidades de pensamiento” y evalúas alternativas. **Tree of Thoughts** propone explorar múltiples caminos y auto-evaluarlos, permitiendo lookahead y backtracking. [11]
  En la práctica, esto te permite que el “subconsciente” (capa estructural) mate ramas inconsistentes temprano, y que el generador rehaga desde un checkpoint.

- **Graph of Thoughts** generaliza esta idea a grafos de dependencias entre “pensamientos”, útil si tu sistema RSI requiere que *Real/Simbólico/Imaginario* interactúen con dependencias no lineales. [12]

Biomimética útil y cómo aterriza en una lectura RSI

Global Workspace como “workspace compartido” para expertos

Tu “sobreposición” se parece más a una **arquitectura tipo Global Workspace** que a un simple MoA por votación.

- La teoría de **Global Workspace (GWT)** (Baars) describe un “espacio de trabajo” de capacidad limitada donde compiten procesos especializados; lo que “gana” se vuelve globalmente accesible/broadcast a otros módulos. El modelo LIDA (Baars & Franklin) presenta esta idea como arquitectura cognitiva con competencia y difusión global. [13]
- En neurociencia, el **Global Neuronal Workspace** enfatiza “ignición” no lineal y acceso global a procesadores locales, de nuevo alineado con “expertos activos” que se sincronizan mediante un estado broadcast. [14]
- En deep learning moderno, hay trabajo explícito sobre **canales compartidos de comunicación** inspirados en global workspace: Goyal et al. proponen coordinación entre módulos mediante un canal compartido y limitado (OpenReview). [15]
- A nivel multiagente LLM, aparece el retorno de la **blackboard architecture**: un “tablero” central donde agentes postean solicitudes y responden según capacidades, reduciendo rigidez del master-slave. Esto es coherente con tu idea de “subconsciente” siempre leyendo/escribiendo un estado común. [16]

Este bloque sugiere una traducción concreta: **tu capa estructural (Simbólico)** puede implementarse como *workspace/blackboard* que controla qué información y restricciones son “broadcast” al generador y a los verificadores en cada paso.

Homeostasis formalizable: de “hambre” a objetivos y setpoints computables

La noción de “Hambre” como utilidad/drive tiene respaldo formal en dos familias:

- **Homeostatic Reinforcement Learning**: Keramati & Gutkin formalizan un marco donde el agente regula estados internos y el refuerzo se integra con homeostasis; su línea conecta drive y reward en un espacio homeostático. [17]
- Revisiones recientes conectan explícitamente homeostasis con RL y conducta motivada (HRRL), definiendo reward como reducción de drive respecto al óptimo homeostático. [18]
- **Active Inference / Free Energy Principle**: Friston propone el principio de energía libre como teoría unificada de cerebro; en active inference se formulan preferencias (p.ej., estados sensoriales preferidos) que guían políticas para mantener al agente en estados viables (homeostasis/allostasis). [19]

En términos de ingeniería: tu “Hambre/Real” puede implementarse como un **modelo de preferencias/valor (reward o expected free energy)** que no solo evalúa outputs finales, sino que también actúa como *prior* o *regularizador* para decisiones intermedias (qué consultar, cuándo detenerse, cuánta exploración gastar).

RSI lacaniano: evidencia de formalización, pero aún mayormente interpretativa

Una implementación "literal" del **Nudo Borromeo RSI** en IA productiva es rara; sin embargo, existe trabajo explícito de formalización computacional que, aunque abstracto, valida que la idea de "tres órdenes acoplados" puede mapear a mecanismos de inferencia.

Un ejemplo directo (Frontiers in Psychology, 2025) propone un modelo **FEP-RSI** que trata Real/Simbólico/Imaginario como unidades de FEP y operacionaliza su interdependencia tipo Borromeo mediante **message passing de errores de predicción** con pesos de precisión; incluso reporta simulaciones donde perturbar el "Simbólico" propaga dinámica hacia Real e Imaginario según la fuerza de acoplamiento. [20]

Interpretación útil para tu arquitectura: los "pesos de acoplamiento" del paper son un análogo computacional sólido para tus **coeficientes de condicionamiento** (cuánto el Simbólico restringe al Imaginario; cuánto el Real empuja objetivos).

Determinar estratificación óptima y diseñar un "repechaje" sin colapsar la exploración

Tu pregunta adicional (cuánta estratificación conviene para evitar aleatoriedad, sin perder alternativas fuera de la estratificación) se parece a tres problemas clásicos: **routing**, **exploration–exploitation**, y **selección/reranking** bajo presupuesto.

Estratificación como routing: elegir qué expertos "entran" y cuándo

En MoE moderno, el routing se resuelve con una **network de compuertas (gating)** que selecciona un subconjunto de expertos por input. El paper fundacional de MoE "sparsely-gated" introduce **Noisy Top-K Gating**: añade ruido y sparsity antes del softmax y luego conserva solo top-k (resto a $-\infty$), lo cual sirve tanto para eficiencia como para evitar colapsos/degeneraciones de routing. [21]
Switch Transformers simplifica routing y hace escalable el MoE a gran tamaño. [22]

Traducción a tu MoA jerárquico: aunque tus "expertos" son agentes/modelos separados (no bloques dentro de un mismo transformer), puedes usar el mismo principio: un **router** que selecciona un top-k de expertos "estratificados" (por relevancia al usuario, dominio, historial) y mantiene **ruido controlado** para que algunos expertos extra entren ocasionalmente.

Además, en despliegue real existe una línea de investigación de **routing/cascading** entre modelos basada en presupuesto:
- **FrugalGPT** aprende un "LLM cascade" para decidir qué combinación de modelos usar por query, buscando trade-off costo–calidad. [23]
- Hay enfoques explícitos de routing como **multi-armed bandit** para seleccionar modelos dinámicamente con preferencias de costo/calidad en inferencia. [24]

Esto es muy relevante: tu "estratificación óptima" puede tratarse como una política que **aprende** cuánto presupuesto asignar a la vía estratificada vs la vía exploratoria.

El “repechaje” como inferencia con selección: Best-of-N, self-consistency y reranking diverso

Tu idea de “una parte pequeña del sistema reintroduce candidatos buenos que quedaron fuera” se alinea con métodos de **test-time scaling**:

- **Self-Consistency** (2022) mejora CoT muestreando múltiples trayectorias de razonamiento y seleccionando la respuesta más consistente (marginalizando trayectorias). Esto es un “repechaje” natural: trayectorias alternativas vuelven vía selección. [25]
- **Best-of-N** con reward model (estilo RLHF) es un patrón ampliamente usado: generas N candidatos y eliges el de mayor score según un evaluador (reward model). InstructGPT (Ouyang et al.) es una referencia canónica de pipelines con modelado de preferencias y evaluación por RM para seleccionar/checkpoints y outputs. [26]
- Pero hay un matiz importante para tu diseño: Best-of-N puede degradar cuando N crece por errores del reward model (“reward hacking”/selección adversarial). Hay evidencia directa de este fenómeno en trabajos sobre límites de Best-of-N. [27]
- Para hacer “repechaje” eficiente, aparece **Speculative Rejection** (2024) como algoritmo de alineamiento en inferencia que busca outputs con alto score de reward model, como Best-of-N, pero con mucha mayor eficiencia computacional. [28]
- Para mantener diversidad útil (no solo “más de lo mismo”), puedes usar reranking de diversidad tipo **MMR (Maximal Marginal Relevance)**, que combina relevancia con novedad/diversidad. Aunque viene de IR/sumarización, es un mecanismo matemático directo para “repechaje” diverso. [29]

Una receta operativa para “cuánta estratificación” conviene

En vez de fijar un número de estratos a priori, la literatura sugiere que es más robusto usar **presupuesto adaptativo** (compute adaptativo):

- **Adaptive Computation Time (ACT)** formaliza la idea de que el modelo aprende cuántos pasos computacionales ejecutar según la dificultad. [30]
- En routing/cascading de LLMs, la idea equivalente es “parar temprano” si ya tienes suficiente calidad, o escalar a más modelos si la señal de calidad es baja (FrugalGPT/cascades). [31]

Propuesta concreta (alineada con tu idea de dos caras de moneda): - Mantén un **canal estratificado** (exploitation) que usa router + memoria + reglas para seleccionar top-k expertos y generar 1–m candidatos. - Mantén un **canal exploratorio/repechaje** (exploration) que genera r candidatos adicionales con rutas menos estratificadas (p.ej., expertos fuera del top-k o sampling más diverso).
- Un **verificador** (PRM/DSVD/GeDi/FUDGE, según el tipo de restricción) decide:
- si el canal estratificado es suficiente (halt), o
- si se debe activar repechaje y/o más capas. [32]

El “óptimo” entonces no es un “número de capas fijo”, sino una política que ajusta (k, m, r) en función de señales: incertidumbre, conflicto entre expertos, violaciones de reglas, o baja puntuación del verificador.

Stack híbrido existente para simular el entrelazamiento sin partir desde cero

Orquestación y control de flujo

Para pasar de lineal/paralelo a "entrelazado", necesitas un runtime que soporte **grafos, estado compartido y loops de evaluación**:

- **LangGraph (LangChain)** se posiciona como framework para flujos controlables: single agent, multi-agent, jerárquico, secuencial, y con facilidad para añadir loops de calidad/moderación. [33]
- Las docs de LangChain describen patrones multiagente donde subagentes se invocan como herramientas, handoffs por estado, y rutas que cambian dinámicamente según variables de estado (esto te sirve para "interrumpir" y "reinyectar" outputs). [34]
- **AutoGen** es especialmente útil si quieres tratar expertos como agentes conversables con patrones de interacción programables (y luego migrar a "collaborative decoding" si necesitas token-level). [35]

Implementar "capas que condicionan logits" (no post-proceso)

Un blueprint plausible (combinando literatura) es un **decodificador compuesto**:

1) **Generador (Imaginario/Ego)**: el experto principal produce logits base.
2) **Memoria (Simbólico, parte)**: agrega distribución tipo kNN-LM desde un datastore personal/histórico, interpolada con el LM. [36]
3) **Reglas/Seguridad (Simbólico, parte)**: agrega penalizaciones o sesgos en logits usando guided decoding: - estilo DExperts (expert/anti-expert) [37]
- o estilo GeDi/FUDGE (discriminador/predictor que ajusta logits por atributo). [38]
4) **Hambre/Real (utilidad)**: agrega un término de valor/preferencia (reward model o expected free energy proxy). El rol "drive reduction" puede inspirarse en HRRL o active inference como formalismo de preferencias. [39]

Conceptualmente, en cada paso de decodificación: - logit_final(token) = logit_experto + α·logit_memoria + β·logit_seguridad + γ·logit_utilidad
(donde α,β,γ son "precisiones"/pesos de acoplamiento, análogos a los pesos de message passing del modelo FEP-RSI). [20]

Esto realiza exactamente lo que pedías: **restricciones y memoria como sesgo latente en generación**, no como filtro post-hoc.

Simular la "sobreposición" en tiempo real

Tienes dos rutas complementarias (y compatibles):

- **Token-level superposition (lo más cercano a tu objetivo)**: adoptar ideas de *collaborative decoding* donde el modelo base aprende cuándo "delegar" el token a expertos. [1]
  Aquí B/C no están "revisando después": están literalmente participando en la emisión del texto.
- **Verify-while-generating + backtracking**: usar un verificador que evalúa prefijos en streaming (DSVD) y, si detecta incoherencia/ilegalidad/inconsistencia, fuerza:
- reponderación de logits, o
- rollback a un checkpoint y exploración alternativa (ToT/GoT). [40]

Usar Sparse Autoencoders como “bias latente” aún más interno

Si tu objetivo es llevar “Simbólico/Real” desde logits hacia **activaciones internas** (un sesgo más “subconsciente” aún), hay evidencia emergente en interpretabilidad/steering:

- Los trabajos de Anthropic/ICLR sobre **Sparse Autoencoders (SAEs)** muestran que pueden extraer features interpretables de activaciones y vincularlas a comportamientos; se conectan explícitamente con la idea de *superposition* de features y su desentrelazado vía SAEs. [41]
- Aparecen métodos de **steering en el espacio SAE** (p.ej., Sparse Activation Steering) para reforzar o suprimir comportamientos inyectando/modulando features durante inferencia. [42]

Esto abre una vía para que tu capa estructural sea literalmente un “modulador” de estados internos, no solo de logits. Es más experimental, pero conceptualmente se alinea con tu intención de “restricción latente”.

Qué existe específicamente sobre MoA por capas

El MoA “clásico” (Wang et al., 2024) es una arquitectura en capas donde múltiples agentes generan en la primera capa y las capas siguientes refinan usando outputs previos como información auxiliar; logra mejoras empíricas fuertes en benchmarks conversacionales. [43]
Esto valida que **estratificar en capas mejora**; sin embargo, por diseño es más secuencial (refinamiento por capas) que tu objetivo de “sobreposición simultánea”. Por eso, la combinación MoA (macro) + collaborative decoding/guided decoding (micro) es un camino particularmente razonable.

Síntesis:
La “Sobreposición de Pensamiento” que propones es plausible si la redefinimos como una combinación de (i) **colaboración a nivel de token** (collaborative decoding) [44], (ii) **capas de sesgo latente** basadas en *product-of-experts / guided decoding* (DExperts, GeDi, FUDGE) [45], (iii) **memoria probabilística en el decodificador** (kNN-LM) [36], y (iv) **verificación paso a paso o en streaming** (PRMs, DSVD) [46].
La parte “RSI/Lacan” tiene menos implementaciones ingenieriles directas, pero ya hay intentos explícitos de formalización computacional vía FEP con acoplamientos tipo Borromeo que pueden servir como guía de diseño (pesos/precisiones y propagación de perturbaciones). [20]

---

[1] [44] [2403.03870] Learning to Decode Collaboratively with Multiple Language Models
https://arxiv.org/abs/2403.03870

[2] [37] [45] DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts
https://arxiv.org/abs/2105.03023?utm_source=chatgpt.com

[3] [38] GeDi: Generative Discriminator Guided Sequence Generation
https://arxiv.org/abs/2009.06367?utm_source=chatgpt.com

[4] FUDGE: Controlled Text Generation With Future Discriminators
https://arxiv.org/abs/2104.05218?utm_source=chatgpt.com

[5] [36] Generalization through Memorization: Nearest Neighbor ...
https://arxiv.org/abs/1911.00172?utm_source=chatgpt.com

[6] ReAct: Synergizing Reasoning and Acting in Language Models
https://arxiv.org/abs/2210.03629?utm_source=chatgpt.com

7 35 AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation
https://arxiv.org/abs/2308.08155?utm_source=chatgpt.com

8 [2305.20050] Let's Verify Step by Step
https://arxiv.org/abs/2305.20050?utm_source=chatgpt.com

9 Chain-of-Verification Reduces Hallucination in Large Language Models
https://arxiv.org/abs/2309.11495?utm_source=chatgpt.com

10 DSVD: Dynamic Self-Verify Decoding for Faithful ...
https://arxiv.org/html/2503.03149v1?utm_source=chatgpt.com

11 Tree of Thoughts: Deliberate Problem Solving with Large Language Models
https://arxiv.org/abs/2305.10601?utm_source=chatgpt.com

12 Graph of Thoughts: Solving Elaborate Problems with Large ...
https://arxiv.org/abs/2308.09687?utm_source=chatgpt.com

13 The LIDA Model of Global Workspace Theory
https://cse.buffalo.edu/~rapaport/Papers/Papers.by.Others/baars-franklin09.pdf?utm_source=chatgpt.com

14 The Global Neuronal Workspace Model of Conscious Access
https://www.antoniocasella.eu/dnlaw/Dehaene_Changeaux_Naccache_2011.pdf?utm_source=chatgpt.com

15 COORDINATION AMONG NEURAL MODULES THROUGH ...
https://openreview.net/pdf?id=XzTtHjgPDsT&utm_source=chatgpt.com

16 LLM-based Multi-Agent Blackboard System for Information Discovery in Data Science
https://arxiv.org/abs/2510.01285?utm_source=chatgpt.com

17 Homeostatic reinforcement learning for integrating reward ...
https://pmc.ncbi.nlm.nih.gov/articles/PMC4270100/?utm_source=chatgpt.com

18 39 Linking homeostasis to reinforcement learning: internal ...
https://www.sciencedirect.com/science/article/pii/S2352154625001305?utm_source=chatgpt.com

19 The free-energy principle: a unified brain theory? - FIL | UCL
https://www.fil.ion.ucl.ac.uk/~karl/NRN.pdf?utm_source=chatgpt.com

20 Formalizing Lacanian psychoanalysis through the free energy principle - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC12180394/

21 Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer
https://arxiv.org/abs/1701.06538?utm_source=chatgpt.com

22 Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
https://arxiv.org/abs/2101.03961?utm_source=chatgpt.com

23 31 FrugalGPT: How to Use Large Language Models While ...
https://arxiv.org/abs/2305.05176?utm_source=chatgpt.com

24 Cost-Efficient LLM Generation via Preference-Conditioned ...
https://arxiv.org/abs/2502.02743?utm_source=chatgpt.com

25 Self-Consistency Improves Chain of Thought Reasoning in Language Models
https://arxiv.org/abs/2203.11171?utm_source=chatgpt.com

26 Training language models to follow instructions with ...
https://arxiv.org/abs/2203.02155?utm_source=chatgpt.com

27 Is Best-of-N the Best of Them? Coverage, Scaling, and ...
https://arxiv.org/abs/2503.21878?utm_source=chatgpt.com

28 Fast Best-of-N Decoding via Speculative Rejection
https://arxiv.org/html/2410.20290v2?utm_source=chatgpt.com

29 The Use of MMR, Diversity-Based Reranking for ...
https://www.cs.cmu.edu/~jgc/publication/The_Use_MMR_Diversity_Based_LTMIR_1998.pdf?utm_source=chatgpt.com

30 Adaptive Computation Time for Recurrent Neural Networks
https://arxiv.org/abs/1603.08983?utm_source=chatgpt.com

32 46 Let's Verify Step by Step 1 Introduction
https://cdn.openai.com/improving-mathematical-reasoning-with-process-supervision/Lets_Verify_Step_by_Step.pdf?utm_source=chatgpt.com

33 LangGraph
https://www.langchain.com/langgraph?utm_source=chatgpt.com

34 Multi-agent - Docs by LangChain
https://docs.langchain.com/oss/python/langchain/multi-agent?utm_source=chatgpt.com

40 DSVD: Dynamic Self-Verify Decoding for Faithful ...
https://aclanthology.org/2025.emnlp-main.1050.pdf?utm_source=chatgpt.com

41 SPARSE AUTOENCODERS FIND HIGHLY INTER
https://proceedings.iclr.cc/paper_files/paper/2024/file/1fa1ab11f4bd5f94b2ec20e794dbfa3b-Paper-Conference.pdf?utm_source=chatgpt.com

42 Steering Large Language Model Activations in Sparse ...
https://arxiv.org/abs/2503.00177?utm_source=chatgpt.com

43 Mixture-of-Agents Enhances Large Language Model Capabilities
https://arxiv.org/abs/2406.04692?utm_source=chatgpt.com