Genoma de la IA, Tensor Global Distribuido y Blockchain: Pensamiento Multi–Arquitectura con Gates, Histeresis y Aprendizaje Continuo

Jaime Andres Menéndez Silva
Director Science & Technology
Departamento Teórico de I+D+i, Barion Technologies
25 de agosto de 2025 ©

***Abstract*—Se integra el *genoma de la IA* con un *tensor global distribuido* de métricas y una *capa blockchain* para garantizar seguridad, trazabilidad y aprendizaje continuo. El pensamiento de la IA se modela como interacción de múltiples arquitecturas (percepción, lenguaje, memoria, recuperación, planificación, acción y lógica simbólica), gobernadas por *gates* leídos del genoma con histeresis, control seguro y consenso. Se extiende (i) formula tensorial (dataset, pipeline, modelo, timestamp), (ii) anclaje on-chain de manifiestos y decisiones, y (iii) pseudocódigo para despliegue federado con promoción/rollback auditable.**

***Key Terms*—Genoma de IA, Tensor Distribuido, Blockchain, Auditoría, Histeresis, CVaR, Consenso, Aprendizaje Continuo**

I. Introducción y Motivación

El pensamiento general de una IA moderna emerge de la *composición* de arquitecturas heterogéneas. Para operarlas en producción con seguridad y aprendizaje constante se requiere: (i) una **especificación declarativa y operativa** (*genoma*) que fije reglas y *gates*; (ii) un **tensor global distribuido** $\mathcal{R}$ que mida calidad/seguridad/costo/latencia por región y nodo; y (iii) una **capa blockchain** que ancle manifiestos, métricas y decisiones (*promote/hold/rollback*) con firmas y, opcionalmente, pruebas de conocimiento-cero.

A. Relación con MLOps Tensorial

Extendemos MLOps Tensorial [1] a entorno federado, multi–arquitectura y con *ledger* para trazabilidad: el genoma gobierna; $\mathcal{R}$ verifica; la cadena certifica la historia de aprendizaje y despliegue.

II. Tensor Global Distribuido y Combinación con Timestamp

Definimos el tensor distribuido con tiempo:

$$\mathcal{R}_{t,\,i,\,j,\,k,\,m,\,g,\,s,\,v,\,rg,\,n} \in \mathbb{R}, \tag{1}$$

donde t es *timestamp* (ventana o marca UTC), i dataset, j pipeline, k modelo/arquitectura, m métrica, g slice, s etapa (shadow/canary/prod), v variante, rg región y n nodo.

a) Combinación tensorial con tiempo.: La combinación $\chi = (i, j, k, t)$ consume energía con una *configuración* evaluada:

$$R_{\chi,m,g,s,v,rg,n} = \mathbb{E}_{(x,y)\sim D_{i,t,g,s,rg}}\big[\ \ell_m\big(y,\ M_k(P_j(x))\big)\ \big].$$

El **registro** por χ incluye seeds, hashes, firmas y CIDs (ver seccion IV).

b) Agregación nodal y regional.:

$$\mathcal{R}^{(rg)}_{t,\dots} = \mathrm{Agg}_n(\mathcal{R}_{t,\dots,rg,n}), \quad \bar{\mathcal{R}}_{t,\dots} = \mathrm{Agg}_{rg}\big(\mathcal{R}^{(rg)}_{t,\dots}\big),$$

con $\mathrm{Agg} \in \{\mathrm{mean},\ \text{median-of-means},\ \mathrm{CVaR}_\alpha,\ \max\}$. Percentiles se obtienen con *sketches* (t-digest/HDR) mergenables.

III. Genoma Artificial y Capas Operativas

A. Espacios y notación

Sea $\mathcal{X}$ el espacio de entradas (posiblemente multimodal) y $\mathcal{Y}$ el de salidas. Denotemos por Arquetipo el conjunto de pipelines tipados $P : \mathcal{X} \to \mathcal{X}^{\mathrm{fusion}}$ y por Model el conjunto de modelos $M_\theta : \mathcal{X} \to \mathcal{Y}$, con $\theta \in \Theta$.

El tensor global distribuido de métricas vive en

$$\mathcal{R} \ \in\ \mathbb{R}^{T\times I\times J\times K\times M\times G\times S\times V\times (RG)\times N},$$

cuyos componentes escribimos como $\mathcal{R}_{t,i,j,k,m,g,s,v,rg,n} \in \mathbb{R}$.

a) Construcción atómica (suma de productos externos): Sea $\mathcal{U}$ el conjunto de *eventos elementales* (ejemplos o *requests*). Definimos mapeos de modos o estados

$$\pi_t,\ \pi_i,\ \pi_j,\ \pi_k,\ \pi_m,\ \pi_g,\ \pi_s,\ \pi_v,\ \pi_{rg},\ \pi_n : \mathcal{U} \to \mathcal{T}, \mathcal{I}, \mathcal{J}, \mathcal{K}, \mathcal{M}, \mathcal{G}, \mathcal{S}, \mathcal{V}, \mathcal{RG}, \mathcal{N}.$$

Sea $\{e^{(t)}_a\}_{a=1}^{T}$ la base canónica de $\mathbb{R}^T$ (análogamente $\{e^{(i)}_b\}$ para $\mathbb{R}^I$, $\{e^{(j)}_c\}$ para $\mathbb{R}^J$, …, $\{e^{(n)}_\ell\}$ para $\mathbb{R}^N$). Para $u \in \mathcal{U}$, definimos el escalar métrico elemental

$$r(u) \ =\ \ell_{\,\pi_m(u)}\Big(y(u),\ M_{\pi_k(u)}\big(P_{\pi_j(u)}(x(u))\big)\Big),$$

donde $P_{\pi_j(u)} \in \mathsf{Arquetipo}$ y $M_{\pi_k(u)} \in \mathsf{Model}$. Entonces, el tensor global se obtiene como superposición de tensores de rango 1:

$$\mathcal{R} = \sum_{u \in \mathcal{U}} r(u)\, e^{(t)}_{\pi_t(u)} \otimes e^{(i)}_{\pi_i(u)} \otimes e^{(j)}_{\pi_j(u)} \otimes e^{(k)}_{\pi_k(u)} \otimes e^{(m)}_{\pi_m(u)} \otimes e^{(g)}_{\pi_g(u)} \otimes e^{(s)}_{\pi_s(u)} \otimes e^{(v)}_{\pi_v(u)} \otimes e^{(rg)}_{\pi_{rg}(u)} \otimes e^{(n)}_{\pi_n(u)}.$$

Esta *construcción atómica* define $\mathcal{R}$ como suma de productos externos (uno por evento); cualquier agregación posterior (promedios, percentiles, CVaR_α, ventanas temporales) se implementa como transformaciones y reducciones sobre los modos correspondientes.

B. Estructura del genoma

El **genoma** es una tupla finita tipada

$$\mathbf{G} = \big(\mathsf{Arch}_{\text{arquitectura}},\ \Theta_{\text{hiperparámetros}},\ \mathsf{Pre}_{\text{pre}},\ \mathsf{Post}_{\text{post}},\ \mathsf{Tools}_{\text{permisos de herramientas}},\ \mathsf{Guard}_{\text{invariantes/budgets}},\ \mathsf{Gates}_{\text{restricciones}},\ \mathsf{Cons}_{\text{consenso}}\big).$$

cuyos componentes se detallan a continuación

a) Arquitectura Arch *(grafo tipado).:* Un DAG tipado $\mathsf{Arch} = (V, E, \tau)$ con (i) nodos V (módulos $\mathsf{V}, \mathsf{L}, \mathsf{M}, \mathsf{R}, \mathsf{G}, \mathsf{U}, \mathsf{S}$); V=Visión/Percepción, L=Lenguaje, M=Memoria, R=Recuperación, G=Goals/Planificador, U=Uso de herramientas/Actuación, y S=Simbólico. La semántica denotacional es el morfismo compuesto (ii) aristas $E \subseteq V \times V$, y (iii) tipado τ que asigna a cada arista un morfismo (lineal o no) compatible con los espacios.

$$\mathsf{Arch}:\ \mathcal{X} \xrightarrow{\mathsf{Pre}} \mathcal{X} \xrightarrow{\text{grafo}} \mathcal{X} \xrightarrow{\mathsf{Post}} \mathcal{Y}.$$

b) Hiperparámetros Θ*.:* Un producto de espacios medibles $\Theta = \prod_{r=1}^{R} \Theta_r$; un modelo M_θ está bien tipado si $(\mathsf{Arch}, \theta) \in \mathcal{C}$ (conjunto de compatibilidad estructural).

c) Operadores $\mathsf{Pre}, \mathsf{Post}$*.:* $\mathsf{Pre} : \mathcal{X} \to \mathcal{X}$ y $\mathsf{Post} : \mathcal{Y} \to \mathcal{Y}$, típicamente composiciones de transformaciones/normalizaciones, conmutando con el tipado de Arch.

d) Permisos de herramientas Tools*.:* Una política como función booleana/medida de cuotas

$$\mathsf{permite}:\ \underbrace{\mathcal{U}}_{\text{herramientas}} \times \underbrace{\mathcal{C}}_{\text{contexto}} \to \{0,1\},$$
$$\mathsf{quota}:\ \mathcal{U} \times \mathcal{C} \to \mathbb{R}_{\geq 0}.$$

que acota invocaciones y uso.

e) Política de resguardo Guard*.:* Colección de invariantes/budgets $\mathcal{I} = \{I_q \leq B_q\}$ (p. ej., privacidad, seguridad, energía), que deben preservarse en todo t.

f) Gates Gates*.:* Conjunto finito de predicados parametrizados Γ_a sobre ventanas de $\mathcal{R}$:

$$\Gamma_a:\ \big(\mathcal{R}_{t-W:t,\cdot}\big)\ \mapsto\ \{\text{true}, \text{false}\},$$

cada uno definido por una 7-tupla

$$\Gamma_a = \big(m_a,\ \mathrm{Agg}_a,\ \mathrm{op}_a,\ (\tau_a^{\uparrow}, \tau_a^{\downarrow}),\ \mathrm{pers}_a,\ \mathrm{stage}_a,\ \mathrm{sev}_a\big),$$

donde m_a es la métrica, Agg_a el agregador (p. ej., $\max$, media, CVaR_α) sobre slices/regiones, $\mathrm{op}_a \in \{\leq, \geq\}$, umbrales de histeresis $(\tau^{\uparrow}, \tau^{\downarrow})$, persistencia $\mathrm{pers}_a = (B, G, \text{cooldown})$, etapa (shadow/canary/prod) y severidad (*hard*/*soft*). El *gate vectorial* por par (j, k) y dataset i es

$$C_{i,j,k} = \bigwedge_{a \in \mathsf{Gates}} \Gamma_a(\mathcal{R}_{t-W:t,\, i,j,k,\cdot}), \qquad \mathcal{V}_{j,k} = \bigwedge_i C_{i,j,k}.$$

g) Consenso Cons*.:* Una política distribuida $\mathsf{Cons} = (q, f, \prec_{\text{lex}}, \mathrm{Agg}_{rg})$ con quórum q, tolerancia a fallas f, orden lexicográfico de prioridades (seguridad $\succ$ privacidad $\succ$ fairness $\succ$ latencia $\succ$ costo $\succ$ calidad) y agregador inter-regiones Agg_{rg} (p. ej., $\max$ para *hard*, media recortada o CVaR para *soft*). Define

$$\mathcal{D}:\ \{v^{(n)}\}_n \xrightarrow{\text{regional}} d^{(rg)} \xrightarrow{\text{global}} d \in \{\text{promote}, \text{hold}, \text{rollback}\}.$$

C. Epigenoma, Transcriptoma y Fenotipo

a) Epigenoma $\mathbf{E}_t$*.:* Estado operativo mutable (knobs) en un politopo $\mathcal{E} \subset \mathbb{R}^p$ con restricciones $\mathbf{E}_t \in \mathcal{K}$ (rangos y velocidades de cambio). Dinámica controlada:

$$\mathbf{E}_{t+1} = \mathrm{proj}_{\mathcal{K}}\big(\mathbf{E}_t + u_t\big),$$
$$u_t = \arg\min_u\ J\big(\widehat{\boldsymbol{r}}_{t+1}(u)\big)$$
$$\text{sujeto a } \mathsf{Gates}\ \&\ \mathsf{Guard}.$$

b) Compilación (Transcriptoma/Build) $\mathbf{T}_t = \mathsf{compile}(\mathbf{G}, \mathbf{E}_t)$*.:* Aplicación determinista que produce un paquete $(\text{imagen}, \text{pesos}, \text{eval\_config})$ con *digests* (CIDs) mediante una función de hashing canónico H:

$$\mathrm{CID}_{\text{eval}} = H(\text{canonical}(\mathsf{Gates}, \mathsf{Guard}, \mathsf{Cons}))$$
$$\mathrm{CID}_{\text{build}} = H(\text{imagen}, \text{pesos}, \mathrm{CID}_{\text{eval}}).$$

c) Fenotipo Φ_t*.:* Semántica operacional inducida por $(\mathbf{T}_t, \mathbf{E}_t)$ y la ventana de métricas: una política

$$\pi_{\mathbf{G}}:\ \big(\mathcal{R}_{t-W:t},\ \text{estado de histeresis}\big)\ \mapsto\ \{\text{promote}, \text{hold}, \text{rollback}\} \times \mathcal{E},$$

obtenida encadenando evaluación de gates con histeresis y consenso Cons, más el controlador seguro sobre $\mathbf{E}_t$.

D. Histeresis como autómata finito

Para cada gate Γ_a definimos un autómata H_a con estados $\{\text{GREEN}, \text{WARN}, \text{RED}\}$ y contadores $(\text{bad}, \text{good})$. Dado $x_a = \mathrm{Agg}_a(\cdot)$:

si $x_a > \tau_a^{\uparrow}$ por B ticks $\Rightarrow$ RED; si $x_a < \tau_a^{\downarrow}$ por G ticks $\Rightarrow$ GREEN; si

con *cooldown* temporal para transiciones descendentes. La decisión local por nodo n es $v^{(n)} = \mathsf{vote}(\{H_a\}_a)$.

E. Semántica declarativa de promoción

La *validez* de (j, k) en ventana W bajo $\mathbf{G}$ es

$$\mathsf{Valid}_{\mathbf{G}}(j, k;\, t) = \mathcal{V}_{j,k}\ \wedge\ \bigwedge_q (I_q \leq B_q)\ \wedge\ \mathsf{Cons}\Big(\{v^{(n)}\}_n\Big),$$

y la acción fenotípica es

$$\mathrm{act}_t = \begin{cases} \text{rollback}, & \neg\mathsf{Valid}_{\mathbf{G}} \\ \text{promote}, & \mathsf{Valid}_{\mathbf{G}} \text{ y todas las regiones promueven} \\ \text{hold}, & \text{en otro caso.} \end{cases}$$

F. Aprendizaje continuo (mutación, cruce, selección)

Sea $\mathcal{M}$ el conjunto de operadores de mutación μ y cruce χ sobre $(\mathsf{Arch}, \Theta, \mathsf{Pre}, \mathsf{Post})$, cerrados bajo tipado. Definimos la *selección*:

$$\mathbf{G}_{t+1} = \arg\max_{\mu,\chi} \ \mathrm{Score}\Big(\widehat{\mathcal{R}}\big(\mathbf{G}_t \xrightarrow{\mu,\chi}\big)\Big)$$
$$\text{sujeto a:} \quad \mathrm{CVaR}_\alpha\Big[\Gamma_a\Big(\widehat{\mathcal{R}}\Big)\Big] = 0, \ \forall a.$$

con prueba *shadow→canary* y validación de Gates + Guard. Toda transición $(\mathbf{G}_t \to \mathbf{G}_{t+1})$ emite manifiesto con CIDs y firmas (anclaje).

G. Propiedades deseables

- **Idempotencia declarativa**: si $\mathbf{G}$ y el stream de métricas no cambian, $\pi_{\mathbf{G}}$ produce decisiones idénticas y $\mathrm{CID}_{\text{eval}}$ permanece constante.
- **Monotonicidad lexicográfica**: si métricas mejoran sin violar prioridades, act_t no empeora (no pasa de promote a hold/rollback) salvo histeresis/cooldown.
- **Composicionalidad**: $\mathsf{Arch}_1 \oplus \mathsf{Arch}_2 = \mathsf{Arch}_1 \circ \mathsf{Arch}_2$ si el tipado lo permite; los gates se preservan por agregación adecuada.

IV. Capa Blockchain: Seguridad y Trazabilidad

Off-chain (pesado, mutable por contenido): datasets, pesos, imagen, eval-config, $\mathcal{R}$ por ventana → objetos *content-addressed* (hash/CID).
On-chain (ligero, inmutable): referencias (CIDs), decisiones, firmas y (opcional) *ZK-proofs* de cumplimiento de gates.

A. Manifiesto por combinación $\chi = (i, j, k, t)$

```
{
  "key": {"t":"2025-08-23T16:20:05Z","i":"ds_A@v3","
    j":"pipe_B@v2","k":"model_C@v7",
          "g":"region=latam&device=mobile","s":"
    canary","v":"A","rg":"us-east-1"},
  "metric_order": ["acc","ece","lat_p99_ms","
    cost_usd","eo_gap","retry_rate"],
  "R": [0.923,0.019,387.0,0.036,0.011,0.004],
  "R_sketch_cid": "cid:QmSketch...",
  "seeds": {"app":1337,"split":2025,"sampler":7,"
    init":123,"hp":99},
  "digests": {
    "dataset":"sha256:...","split_manifest":"sha256
    :...",
    "pipeline":"sha256:...","model_code":"sha256
    :...",
    "weights":"sha256:...","image":"sha256:...","
    eval_config":"sha256:..."
  },
  "signature": {"type":"sigstore","bundle":"...
    base64..."}
}
```

Listing 1. Manifiesto de slice con timestamp y CIDs.

B. Contrato (interfaz lógica)

```
contract AGILedger {
  event RecordSlice(bytes cid_slice, bytes32
    eval_cfg, bytes32 build_id);
  event Promote(bytes32 build_id, bytes reasonHash);
  event Rollback(bytes32 build_id, bytes reasonHash)
    ;
  event MutateGenome(bytes32 parentG, bytes32 childG
    );
  event CrossoverGenome(bytes32 GA, bytes32 GB,
    bytes32 Gchild);

  function registerGenome(bytes32 Gcid) external;
  function recordSlice(bytes cid_slice, bytes32
    eval_cfg, bytes32 build_id, bytes zkProof)
    external;
  function promote(bytes32 build_id, bytes
    reasonHash, bytes zkProof) external;
  function rollback(bytes32 build_id, bytes
    reasonHash) external;
}
```

Listing 2. Interfaz de contrato (pseudo-Solidity).

C. Pruebas ZK opcionales

Se publican pruebas de rango/comparación (e.g., acc $\geq \tau$, lat_p99 $\leq \tau$) sin exponer valores ni slices sensibles; on-chain se verifica $\mathsf{verify}(\pi_{\text{zk}}) = \text{true}$.

V. Gates con Histeresis y Consenso Distribuido

Un gate para métrica m y ventana W:

$$\Gamma_m^{(n)} = \mathrm{Agg}_g\big[\mathcal{R}_{t-W:t,\ldots,m,g,s,rg,n}\big] \ \text{op} \ \tau_m. \tag{2}$$

Histeresis: umbrales dobles $\tau_m^{\uparrow} > \tau_m^{\downarrow}$, persistencia (contadores de *malo/bueno*) y *cooldown* tras cambio.

a) Consenso regional y global.: Cada nodo emite voto $v^{(n)} \in \{\text{promote}, \text{hold}, \text{rollback}\}$. Por región:

$$\text{decisión}(rg) = \begin{cases} \text{rollback}, & \#\{n : v^{(n)} = \text{rollback}\} \geq q \\ \text{promote}, & \#\{n : v^{(n)} = \text{promote}\} \geq q \\ \text{hold}, & \text{otro.} \end{cases}$$

Global: si $\exists rg$ con *gate hard* violado, **rollback**; si todas las regiones promueven, **promote**; en otro caso, **hold**. Las decisiones se emiten como eventos on-chain con razones firmadas.

VI. Control Seguro y Aprendizaje Continuo

A. MPC robusto por región

Sea $w_{rg} \in [0, 1]$ el peso de tráfico del candidato:

$$\min_{w_{rg}} \sum_{rg} \Big(J(\widehat{\boldsymbol{r}}_{t+1}^{(rg)}(w_{rg})) + \lambda_{\text{sw}}(w_{rg} - w_{rg}^{\text{prev}})^2 \Big) \tag{3}$$

$$\text{s.a. } \Gamma(\widehat{\boldsymbol{r}}_{t+1}^{(rg)}) \leq 0, \ \mathrm{CVaR}_\alpha = 0, \ \forall rg. \tag{4}$$

B. Supervisión dual on-line

$$\lambda_m^{(rg)} \leftarrow \big[\lambda_m^{(rg)} + \eta \ \max(0, \ r_{t,m}^{(rg)} - \tau_m)\big]_+.$$

Si $\sum_m \lambda_m^{(rg)} > \Lambda_{\text{crit}}$, voto regional de *rollback*.

C. Aprendizaje continuo seguro

Actualizaciones Θ_{t+1} (fine-tuning/LoRA/adapters) se permiten sólo si:

$$|\Delta\mathcal{R}_t| \le \varepsilon, \quad \|\Delta W\|_F \le B, \quad \text{y gates hard en verde.}$$

Cada actualización loguea $(\Delta\mathcal{R}, \|\Delta W\|_F)$ y se ancla on-chain.

VII. Pseudocódigo de Orquestación con Ledger

```
def map_eval_node(stream, genome, epi):
    R_local = roll_metrics(stream, sketches=["
    tdigest","hdr","ema"])
    votes = []
    for window in windows(stream):
        vote = apply_gates_hysteresis(R_local[window
    ], genome["gates"], epi["state"])
        votes.append(vote)  # "promote" | "hold" | "
    rollback"
    return votes, export_sketches(R_local)
```

Listing 3. Mapa nodal: evaluación, histeresis y export de sketches.

```
def reduce_and_anchor(votes_by_region, policy,
    build_id, rpc):
    q = policy["quorum"]
    per_rg = {}
    for rg, votes in votes_by_region.items():
        c = {k:votes.count(k) for k in set(votes)}
        per_rg[rg] = "rollback" if c.get("rollback"
    ,0) >= q else \
                     "promote"  if c.get("promote"
    ,0)  >= q else "hold"
    if "rollback" in per_rg.values():
        rpc.rollback(build_id, reasonHash=
    hash_reason(per_rg))
        return "rollback", per_rg
    if all(v=="promote" for v in per_rg.values()):
        rpc.promote(build_id, reasonHash=hash_reason
    (per_rg), zkProof=mk_proof())
        return "promote", per_rg
    return "hold", per_rg
```

Listing 4. Reducción por consenso + decisión global + anclaje.

```
{
  "build_id": "cid:QmBuild...",
  "genome_cid": "cid:QmGenome...",
  "epi_id": "2025-08-23T16:44:10Z",
  "digests": {
    "image": "docker://...@sha256:...",
    "weights": "sha256:...",
    "pipeline": "sha256:...",
    "eval_config": "sha256:..."
  },
  "timestamps": { "compiled_at_utc": "2025-08-23T16
    :44:10Z" }
}
```

Listing 5. Manifiesto de build/transcriptoma con timestamp y CIDs.

VIII. Pensamiento como Interacción de Múltiples Arquitecturas

Esta sección aterriza el *pensamiento* de la IA como un ciclo de control discreto que integra varias arquitecturas especializadas sobre un *bus tipado* de mensajes y un contrato de puertos bien definido. Denotamos el conjunto de módulos por

$$\mathcal{A} = \{\mathsf{V}, \mathsf{L}, \mathsf{M}, \mathsf{R}, \mathsf{G}, \mathsf{U}, \mathsf{S}\},$$

donde: V (visión/percepción), L (lenguaje/razonamiento distribuido), M (memoria semántica+episódica), R (recuperación), G (goals/planificador), U (uso de herramientas/acción) y S (simbólico/solver).

A. Puertos tipados e invariantes de interfaz

Cada módulo expone puertos de entrada/salida con tipos y contratos:

- $\mathsf{V}: \mathcal{X} \to \mathcal{Z}_{\mathsf{V}}$ (encoders multimodales); $\|h_t^{(\mathsf{V})}\|_2 \le B_{\mathsf{V}}$.
- $\mathsf{L}: \mathcal{Z}_{\mathsf{L}} \times \mathcal{C} \to \mathcal{Z}_{\mathsf{L}}$ (estado recurrente y contexto); preserva cotas de atención y longitud.
- M : KV-store semántico/episódico con *get/put* tipados y políticas de retención (TTL, PII_mask).
- $\mathsf{R}: \mathrm{Retrieve}(\mathrm{Query}, \mathsf{M}) \to \mathrm{KV}$ con *scores* y *evidences*.
- $\mathsf{G}: \mathrm{Plan}(b_t, \mathsf{S}) \to g_t$ (objetivos, restricciones, pasos, criterios de parada).
- $\mathsf{U}: \mathrm{Act}(g_t, \mathrm{tools}) \to a_t$ bajo Tools/Guard (cuotas, presupuestos, permisos).
- $\mathsf{S}: \mathrm{Solve}(\Phi, \mathrm{facts}) \to$ derivaciones (reglas/invariantes verificables).

Las invariantes de seguridad/privacidad/costo (Guard) aplican a todas las llamadas; *gates* (§IV y secciones previas) gobiernan el flujo.

B. Estado de creencia y ciclo operativo

Sea $x_t \in \mathcal{X}$ la observación en el *tick* t. Definimos un estado de creencia agregado

$$b_t = \left[h_t^{(\mathsf{V})},\ h_t^{(\mathsf{L})},\ \mathrm{KV}_t,\ \mathrm{hechos}_t^{(\mathsf{S})}\right].$$

El ciclo de pensamiento ejecuta:

$$h_t^{(\mathsf{V})} = \mathrm{Enc}_{\mathsf{V}}\big(\mathsf{Pre}(x_t)\big), \tag{5}$$

$$\mathrm{KV}_t = \mathrm{Retrieve}\Big(\mathrm{Query}\big(h_{t-1}^{(\mathsf{L})}\big),\ \mathsf{M}\Big), \tag{6}$$

$$h_t^{(\mathsf{L})} = \mathrm{XAttn}\big(h_{t-1}^{(\mathsf{L})},\ [\,h_t^{(\mathsf{V})},\ \phi(\mathrm{KV}_t)\,],\ \psi(\mathsf{S})\big), \tag{7}$$

$$g_t = \mathrm{Plan}\big(b_t\big), \qquad a_t = \mathrm{Act}\big(h_t^{(\mathsf{L})},\ g_t,\ \mathrm{tools}\big). \tag{8}$$

Los mapeos ϕ, ψ son proyecciones/normalizaciones tipadas que alinean embeddings y hechos simbólicos.

a) Inyección de cápsulas (opcional ver sección XI): Si existe una CGC activa $c > 0$ (§S6), se aplica síntesis:

$$h_t^{(\mathsf{L})} \leftarrow h_t^{(\mathsf{L})} \oplus \mathrm{CGC}_c, \quad \mathrm{KV}_t \leftarrow \mathrm{KV}_t \cup \mathrm{adn}(\mathrm{CGC}_c).$$

C. Métricas operativas y emisión al tensor $\mathcal{R}$

Cada paso produce un vector de métricas

$$\boldsymbol{r}_t \in \mathbb{R}^M, \quad m \in \{\text{calidad}, \text{seguridad}, \text{latencia}, \text{costo}, \text{fairness}, \text{retry}\},$$

que se registra en $\mathcal{R}_{t,i,j,k,m,g,s,rg,n}$ con *sketches* mergenables (t-digest/HDR) y *digests* (seeds, hashes, CIDs). Las ventanas W alimentan Gates con histeresis; la decisión regional se agrega por quórum y la global por Cons (§5–6), quedando *anclada on-chain*.

D. Políticas de herramientas y control seguro en bucle

El subconjunto de llamadas a herramientas en t es $\mathcal{U}_t \subseteq \mathcal{U}$. Se verifica:

$$\sum_{u \in \mathcal{U}_t} \mathsf{quota}(u, \mathcal{C}_t) \;\le\; B_{\text{tools}}, \qquad \mathsf{permite}(u, \mathcal{C}_t) = 1.$$

El controlador seguro ajusta w_{rg} (tráfico) y $\mathbf{E}_t$ (epigenoma) sólo si *gates hard* están en verde y $\mathrm{CVaR}_\alpha = 0$ (ver §6):

$$\big(w_{rg},\ \mathbf{E}_{t+1}\big) \leftarrow \mathrm{MPC}\big(\boldsymbol{r}_t,\ \mathbf{E}_t,\ \mathsf{Gates},\ \mathsf{Guard}\big).$$

E. Degradación controlada y tolerancia a fallos

Definimos un *árbol de degradación*:

$$\mathsf{V} \to \mathsf{L} \to \mathsf{R} \to \mathsf{G} \to \mathsf{U}$$

$$\rightsquigarrow \quad \text{fallbacks} = \{\text{cache local},\ \text{plantillas},\ \text{modo sólo-lectura}\}.$$

Si $\mathrm{KV}_t = \varnothing$ o hay *timeouts*, se conmuta a plantillas calibradas; si *gates hard* violan por B *ticks*, el voto regional es *rollback* y se reduce w_{rg} a cero.

F. Pseudocódigo operativo (por tick)

```
def think_step(x_t, genome, epi, policy):
    zV = Enc_V(Pre(x_t))
    KV = Retrieve(Query(state.L), Memory)
    hL = XAttn(state.L, [zV, project(KV)], lift(
    Symbolic))
    if has_active_CGC():
        hL = inject_capsule(hL, CGC.load())
        KV = KV.union(CGC.index())
    g  = Plan(belief=(zV,hL,KV,Symbolic.facts()))
    a  = Act(hL, g, tools=policy.tools_allowed())
    r  = measure_metrics(x_t, a, KV)          #
    calidad, latencia, costo, seguridad...
    emit_tensor_R(r, sketches=True, digests=True)
    vote = apply_gates_hysteresis(window=W, R_slice=
    recent_R(), gates=genome["gates"])
    epi  = safe_controller_update(epi, r, guards=
    genome["guard"])
    return a, r, vote, epi
```

Listing 6. Bucle de pensamiento multi–arquitectura con control seguro.

G. Semántica de cierre con auditoría

Cada *tick* emite trazas firmadas τ_t (decisiones, $\boldsymbol{r}_t$, cambios en $\mathbf{E}_t$, usos de herramientas) que se empaquetan off-chain (*content-addressed*) y se referencian on-chain. Con ello, el pensamiento multi–arquitectura es:

$$\pi_{\mathbf{G}} : \ (x_{t-W:t},\ \mathcal{R}_{t-W:t},\ \mathbf{E}_t) \ \mapsto \ (a_t,\ w_{rg},\ \mathrm{act}_t),$$

estable por histeresis, *robusto* por consenso/CVaR y *auditado* por anclaje en la cadena.

IX. Seguridad, Privacidad y Cumplimiento

Separación de responsabilidades: el runtime no puede alterar *gates* ni prioridades; sólo acciona w_{rg} o *rollback*.
Privacidad: ZK-proofs de cumplimiento; $\mathcal{R}$ completo off-chain cifrado; on-chain sólo CIDs y hashes.
Reproducibilidad: decisiones vinculan CIDs de build, eval-config y manifiestos $\chi = (i, j, k, t)$.
No-regresión: se exigen cotas $|\Delta\mathcal{R}| \le \varepsilon$ y $\|\Delta W\|_F \le B$ previo a promoción; ambos valores se registran y firman.

X. Solidificación Genómica y Blockchain

Se propone que el *genoma de IA* incorpore un modo de **solidificación** del conocimiento: cuando un tema alcanza estabilidad y utilidad suficientes, se crea una **Cápsula Genómica de Conocimiento** (CGC) que contiene una síntesis verificable (p.ej., ΔW comprimido, reglas simbólicas, plantillas de recuperación y calibraciones). La CGC se *cifra*, se *direcciona por contenido* (hash/CID) y se *ancla on-chain* con metadatos y firmas. Estas CGC quedan como *genes sólidos* que otras arquitecturas consumen mediante interfaces tipadas.

Objetivos: (i) proteger lo aprendido (inmutabilidad auditada), (ii) facilitar la composición entre arquitecturas (síntesis compacta), (iii) habilitar rollback/linaje transparente, y (iv) compartir conocimiento sin exponer artefactos crudos (privacidad y control de acceso).

XI. Criterios de Solidificación

Sea $\mathcal{R}_{t,i,j,k,m,g,s,rg}$ el tensor distribuido de métricas y defínase una *ecuación de solidez*:

$$\begin{aligned}\mathsf{Sol}(T) = \omega_1\,\mathsf{Estab}(T) \;+\; \omega_2\,\mathsf{Reprod}(T) \;+\; \omega_3\,\mathsf{Util}(T) \\ -\; \omega_4\,\mathsf{Riesgo}(T).\end{aligned} \tag{9}$$

para una ventana temporal T.

- Estab: estabilidad de desempeño ($1 - \mathrm{CVaR}_\alpha$ de variaciones, ausencia de *flapping* con histeresis).
- Reprod: reproducibilidad (coincidencia de seeds/hashes/-firmas en corridas independientes).
- Util: utilidad cruzada (tasa de uso exitoso por otras arquitecturas/slices).
- Riesgo: cercanía a *gates* críticos y margen de seguridad (lexicográfico: seguridad $\succ$ privacidad $\succ$ fairness $\succ$ latencia).

Regla: se *solidifica* si $\mathsf{Sol}(T) \ge \tau_{\text{sol}}$ durante B ventanas consecutivas y no existen *hard gates* violados. La decisión queda anclada en cadena con razones firmadas.

XII. Cifrado, Anclaje y Control de Acceso

Cada CGC se empaqueta off-chain como objeto *content-addressed* (CID) y se **cifra** (sobre envuelta) con políticas de acceso. On-chain se registra:

1) **Metadatos inmutables**: CID, tema, versión, huellas (sha256), timestamps.
2) **Firmas y atestaciones**: autores, validador de *gates*, consenso regional/global.
3) **Política de acceso**: referencia a identidades/roles de agentes (DID/VC) y a pruebas opcionales de conocimiento-cero (*ZK*) que certifican *cumplimiento de gates* sin revelar el contenido.

La clave de descifrado se gestiona por *gobernanza* (multifirma/umbral). La revocación/rotación de claves y la *deprecación* de CGC se publican como eventos on-chain.

XIII. Interacción precisa entre Arquitecturas

Para que una CGC sea *útil*, definimos interfaces tipadas:

- **Adapter paramétrico**: ΔW comprimido (p.ej., LoRA/adapter) con especificación de compatibilidad (espacio, capas, rango r, normas $\|\Delta W\|_F$).
- **Cápsula de recuperación**: índices/embeddings específicos $\rightarrow$ respuestas de alta precisión para un tema; incluye umbrales y post-calibración.
- **Reglas simbólicas**: invariantes, restricciones lógicas y *guardrails* derivados del entrenamiento.
- **Resumen experto**: un "précis" formal del tema (definiciones, límites, contraejemplos) usable por planificadores/razonadores.

Estas interfaces permiten que L (lenguaje), V (visión/percepción), R (retrieval), G (planificador) y S (simbólico) *componen* la CGC sin reentrenar desde cero.

XIV. Flujo Operativo de Solidificación

1) **Aprendizaje y evaluación**: el candidato mejora $\mathcal{R}$ en tema q; los *gates* (con CVaR e histeresis) permanecen verdes.
2) **Detección de convergencia**: $\mathsf{Sol}(T)$ supera τ_{sol} y persiste B ventanas; no hay *hard* violados.
3) **Síntesis**: se comprime el conocimiento en una CGC con interfaces tipadas y métricas ($\Delta\mathcal{R}$, $\|\Delta W\|_F$).
4) **Cifrado y anclaje**: se cifra la CGC, se registra CID, firmas y política de acceso en blockchain.
5) **Disponibilización**: otras arquitecturas consultan el *catálogo genómico* y referencian la CGC con políticas de uso (sujeto a gates).
6) **Evolución**: nuevas evidencias producen CGC_{v+1} (mutación/cruce), que hereda y puede *depreciar* versiones previas con evento on-chain.

XV. Formalización Tensorial con Cápsulas y Timestamp

Ampliamos el índice del tensor para registrar cápsulas c:

$$\mathcal{R}_{t,i,j,k,c,m,g,s,rg} \in \mathbb{R}, \qquad c \in \{0\ (\text{sin CGC}), 1, \ldots, C\}. \tag{10}$$

La *activación* de una CGC se modela como operador de síntesis $\oplus$:

$$\hat{Y}^{(i,j,k,c)} = \left(M_k \oplus \mathrm{CGC}_c\right)\Big(P_j(X^{(i)})\Big), \tag{11}$$

donde $\oplus$ denota inyección de ΔW o sustitución de reglas/recuperación. El gate sobre CVaR/consenso se aplica tanto a $c{=}0$ como a $c{>}0$. Cada combinación (i, j, k, c, t) se registra con seeds/hashes/firmas y CID del manifiesto.

XVI. Gobernanza, Políticas y Auditoría

Gobernanza: cambios de *gates*, publicación de CGC y accesos sensibles requieren *multifirma* o quórum. **Política de acceso**: basada en roles de agentes y necesidad (p.ej., sólo planificadores y módulos de recuperación pueden descifrar cierta CGC). **Auditoría**: cada uso de CGC emite huellas (respetando privacidad), enlazadas a la decisión de despliegue y a $\mathcal{R}$ local/regional/global. **Revocación**: si aparece evidencia adversa, se marca la CGC como *deprecada* y se publica evento on-chain; la inferencia consulta el estado vigente antes de cargar.

XVII. Ventajas, Limitaciones y Mitigaciones

Ventajas: composicionalidad entre arquitecturas, protección de propiedad intelectual, reproducibilidad, menor latencia (al reutilizar síntesis), y gobernanza fuerte. **Limitaciones**: gestión de claves y políticas, riesgo de *fragmentación* del conocimiento en cápsulas pequeñas, sobrecosto de verificación. **Mitigaciones**: catálogos con *scores* de cobertura/solapamiento, consolidación periódica (fusión de CGC), rotación de claves, y *hints* de planificador para priorizar cápsulas más útiles/recientes.

XVIII. Relación con CVaR y Consenso

La solidificación se dispara sólo si las *colas* (CVaR) están controladas y la *decisión* pasa por *consenso* regional y global con histeresis. Así, una CGC no se "graba en piedra" por un buen promedio local sino por *robustez sostenida* en $\mathcal{R}$ y validación distribuida. El *ledger* conserva el linaje: quién/qué/por qué se solidificó o se dejo libre de la cadena.

XIX. Conclusión Específica

El **genoma de IA solidificado** convierte aprendizajes estables en *cápsulas cifradas* con identidad y políticas, ancladas en blockchain. Esto permite que múltiples arquitecturas *razonen y actúen* con piezas de conocimiento sintetizadas, confiables y auditables, manteniendo la agilidad del aprendizaje continuo sin sacrificar seguridad ni trazabilidad.

References

[1] J. A. Menéndez Silva, "MLOps Tensorial: Formalización Algebraica Unificada y Concreción Algorítmica con un Caso de Aplicación a AGI mediante Aprendizaje por Refuerzo," Barion Technologies, 2025.