Software de Coherencia — Especificación del Lenguaje -ASM

Versión 1.0 — 1 de noviembre de 2025

Autor: Genaro Carrasco Ozuna · ORCID: 0009-0005-6358-9910

---

0. Propósito

Este documento define un *lenguaje de máquina de coherencia* (-ASM) minimalista y **virgen**, sin datos de medición ni casos de uso específicos. Su objetivo es ofrecer una semántica ejecutable para procesos cuyo equilibrio se rige por la ley:

$$Q - \phi = \Sigma \leftrightarrow \chi,$$

donde Q (intención), ϕ (fricción), Σ (coherencia) y χ (soporte) determinan el estado válido del sistema. Esta especificación es independiente de cualquier dominio (científico, legal, documental, experimental) y sirve como **núcleo del Software de Coherencia**.

1. Axiomas

1. **Balance**: Una instrucción válida no disminuye Σ por debajo de un umbral de seguridad, y toda transición de estado debe poder rastrearse en χ.
2. **Falsabilidad**: Toda salida debe ser sometible a pruebas nulas (null tests) definidas a nivel de lenguaje.
3. **Trazabilidad**: Cada ciclo produce huellas mínimas (identidad, licencia, hash, tiempo) en el soporte χ.
4. **Parquedad**: El ISA es mínimo; no hay opcodes redundantes.

2. Modelo de ejecución

El sistema actúa como un *microkernel de coherencia*. Cada *proceso coherencial* mantiene cuatro registros lógicos y un estimador:

$$(Q,\ \phi,\ \Sigma,\ \chi;\ \kappa_\Sigma)$$

- Q — registro de intención de control (`Q_ctrl`).
- ϕ — registro de fricción (entropía/ruido/norma).
- Σ — registro de coherencia (estado de sincronía).
- χ — registro de soporte (infraestructura material/digital).
- κ_Σ — tasa de restauración de coherencia (k-rate).

3. Tipos y objetos

- `intent`: cadena o estructura que representa causa/propósito.
- `trace`: tupla de metadatos (licencia, hash, identidad, tiempo).
- `metric`: escalar normalizado $m \in [0, 1]$.
- `substrate`: descriptor abstracto del soporte χ.

4. Métricas de coherencia (sin datos)

El lenguaje define símbolos, no valores:

$$LI \in [0, 1], \quad R(t) \in [0, 1], \quad \mathrm{RMSE}_{SL} \in [0, 1], \quad \kappa_\Sigma \in [0, 1].$$

No se fija fórmula concreta (queda al *runtime*); la especificación sólo impone que κ_Σ mida la recuperación relativa de Σ frente a ϕ en un intervalo Δt:

$$\kappa_\Sigma \equiv \mathcal{K}(\Sigma_{t-1}, \Sigma_t, \phi_t, \Delta t), \quad \text{con } \kappa_\Sigma \in [0, 1].$$

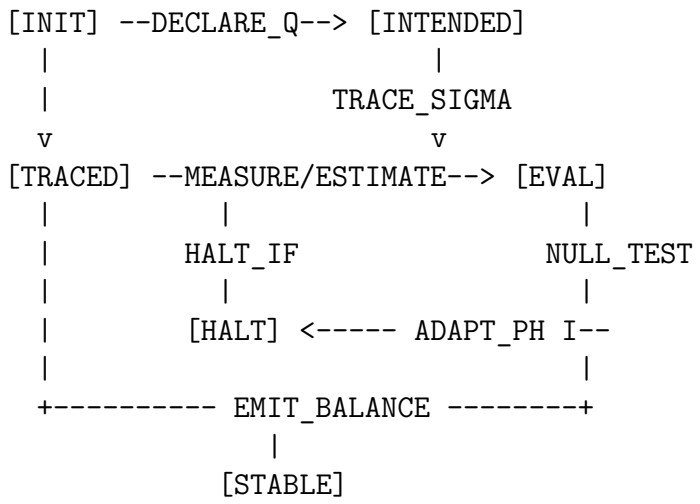
Nota: Esta definición es abstracta y no incorpora datos ni heurísticas.

5. ISA — Instrucciones del Lenguaje -ASM

Sintaxis canónica en estilo ensamblador. Cada instrucción puede *emitir* huellas mínimas en χ (dependiente del runtime).

Instrucción	**Semántica**
`DECLARE_Q <intent>`	Inicializa Q con causa/propósito explícito y lo fija como `Q_ctrl`.
`TRACE_SIGMA <trace>`	Establece traza mínima de coherencia Σ (licencia, identidad, hash, tiempo) sobre χ.
`MEASURE_SIGMA`	Solicita al runtime calcular símbolos $LI, R(t), \mathrm{RMSE}_{SL}$ (sin valores en la especificación).
`ESTIMATE_KAPPA`	Solicita al runtime estimar κ_Σ a partir de los estados de Σ y ϕ (sin fórmula impuesta).
`LIMIT_PHI <policy>`	Aplica política de contención de fricción (norma, licencia, contabilidad, ética).
`ADAPT_PHI <strategy>`	Ajusta dinámicamente ϕ para estabilizar el sistema.
`LOCK_PQ <p:q, params>`	Declara intento de sincronía p:q (no fija implementación).
`NULL_TEST <mode>`	Ejecuta prueba nula (p. ej., inversión, aleatorización, silencio) sobre el ciclo corriente.
`EMIT_BALANCE`	Declara ciclo válido $Q - \phi = \Sigma \leftrightarrow \chi$ y solicita registro en χ.
`HALT_IF <predicate>`	Detiene el ciclo si la condición de coherencia falla (predicado de runtime).

6. Máquina de estados (abstracta)



7. Programa mínimo (sin datos ni dominio)

```
; Programa -ASM virgen (no ejecuta mediciones reales)
DECLARE_Q        "origen-etico:v1"
TRACE_SIGMA      (license="CC-BY-NC-SA-4.0", id="ORCID:...", t="ISO-8601", hash="...")
MEASURE_SIGMA
ESTIMATE_KAPPA
LIMIT_PHI        "policy:coherencia-minima"
LOCK_PQ          "p:q=1:1"
NULL_TEST        "mode:invert"
HALT_IF          "predicate:coherencia_invalida"
EMIT_BALANCE
```

Observación: El programa no contiene ni requiere datos. Los selectores (políticas, predicados) son nombres abstractos.

8. ABI/Runtime (contrato mínimo)

- **Entrada**: secuencia de instrucciones -ASM.
- **Salida**: registro en χ de huellas mínimas y `status` del ciclo.
- **Predicados válidos**: `coherencia_valida`, `coherencia_invalida`, `phi_excesiva`.
- **Null tests**: `invert`, `shuffle`, `mute`, `cross_host`; semántica definida por el runtime.

9. Conformidad y pruebas (sin métricas numéricas)

1. **Conformidad sintáctica**: el parser acepta todas las instrucciones definidas y rechaza opcodes no especificados.
2. **Conformidad semántica**: cada instrucción modifica exclusivamente su registro lógico asociado.

3. **Conformidad de falsabilidad**: `NULL_TEST` debe existir en todo programa que pretenda declarar `EMIT_BALANCE`.

10. Seguridad y ética

- El lenguaje impone **origen ético explícito** vía `DECLARE_Q`.
- **Trazabilidad mínima obligatoria** vía `TRACE_SIGMA`.
- **Falsabilidad estructural** vía `NULL_TEST`.

11. Autocrítica

Este documento es **virgen**: no incluye datos, fórmulas operativas concretas ni dominios de aplicación. La abstracción garantiza portabilidad y evita sesgos, pero traslada al *runtime* la responsabilidad de:

1. Implementar el cálculo de $LI, R(t), \mathrm{RMSE}_{SL}$ y κ_Σ conforme a su dominio.
2. Definir políticas y estrategias de ϕ (`LIMIT_PHI`, `ADAPT_PHI`).
3. Establecer predicados de `HALT_IF` sin comprometer la neutralidad del lenguaje.

La parquedad es deliberada para preservar unicidad y universalidad del *Software de Coherencia*.

Contacto: geozunac3536@gmail.com