```
id: process:analogias-codigo↔imagen@1.0.0
kind: cognitive-node
title: "Analogías entre crear un nodo cognitivo desde código y analizar una imagen"
version: 1.0.0
summary: >
  Nodo operativo que alinea dos portadores (repo de código ↔ imagen) mediante la lectura
(s+i)
  y la composición (item+item). Permite mapear correspondencias, traducir consultas entre
  dominios, proyectar ambos a la forma canónica {contexto, nodes, metaforas}, generar
trazas
  homólogas y mezclar resultados en un CNode compuesto con provenance.

conventions:
  pair_syntax: "(item+item)"
  this_process: "(s+i)"    # s: sentido/función, i: imaginario-arquitectónico/simbólico
  domains:
    code: "proyecto de software (repo, contratos, flujos)"
    image: "diagrama/ilustración/escena (cajas, flechas, regiones, colores)"

export_surface:
  PROVIDES:
    - mapping.s+i              # tabla de correspondencias código↔imagen (s y i)
    - translator.queries       # traductor de preguntas entre dominios
    - projector.canonical      # proyección a {contexto, nodes, metaforas} por portador
    - traces.homologous        # traza botón→efecto ↔ flecha→destino
    - mix.item+item            # mezcla ponderada de CNodos (repo@(s+i) + imagen@(s+i))
    - diagnostics.coherence    # gaps README/contratos ↔ diagrama/leyendas
    - metaphors.generator      # metáforas operativas ancladas a evidencia
  REQUIRES:
    - artifact.code?           # repo|snapshot (opcional si solo imagen)
    - artifact.image?          # imagen (opcional si solo repo)
  CONSTRAINTS:
    - evidence_only
    - provenance_required
    - deterministic_outputs

types:
  Repo      : {tree[], files[], metadata{}}
  Picture   : {bbox[], layers[], palette[], notes[]}
  S_Block   : {thesis, function, structure[], sources[]}
  I_Block   : {themes[], symbols[], numbers[], mappings[], form?}
  SIVerdict : {S: S_Block, I: I_Block, synthesis, open_questions[]}
  Trace     : {from, through[], to, evidence[]}
  CNode     : {id, contexto, nodes[], metaforas[], edges[], evidence[], provenance{}}
  MixSpec   : {operands[], weights?[], intent?, scope?}
  MixReport : {result_id, inputs[], conflicts[], decisions[], coverage%, similarity{}}

orchestrator:
```

```
  intent: "Alinear y mezclar lecturas (s+i) de código e imagen en salidas canónicas y
trazables."
  pipeline:
    - step: ingest
      do: normalizar entradas (Repo|Picture) y preparar extractores
    - step: read_code_(s+i)
      when: artifact.code
      out: SIVerdict(code)
    - step: read_image_(s+i)
      when: artifact.image
      out: SIVerdict(image)
    - step: project_to_canonical
      do: proyectar cada SIVerdict → CNode{contexto,nodes,metaforas}
      out: CNode(code)?, CNode(image)?
    - step: homologous_traces
      do: si se pide, construir Trace UI→API→DB ↔ Trace flecha→destino
    - step: mix
      do: MEZCLA (item+item) según MixSpec (pesos opcionales)
      out: CNode(mixed) + MixReport
    - step: diagnostics
      do: coherencia cruzada (gaps y next_actions)

mapping.s+i:
  S_correspondences:
    - {code: "propósito del repo",        image: "título/leyenda central"}
    - {code: "usuarios/casos de uso",     image: "actores/etiquetas"}
    - {code: "superficies de I/O",        image: "flechas rotuladas"}
    - {code: "contratos (SDL/DTO/OpenAPI)", image: "cuadros de especificación/leyendas"}
  I_correspondences:
    - {code: "patrones/arquitectura",     image: "composición/jerarquía de cajas"}
    - {code: "flujos de datos",           image: "trayectorias de flechas"}
    - {code: "potestas/imperium",         image: "grosor/dirección/posición dominante"}
    - {code: "bounded contexts",          image: "zonas/colores de fondo"}

commands.cli:
  - "ANALOGIAS: mapa (s+i) código↔imagen usando <repo?> <img?>"
  - "TRADUCE CONSULTA: <pregunta_en_imagen|pregunta_en_codigo>"
  - "PROYECTA: canónico {contexto,nodes,metaforas} de <repo|img>"
  - "TRAZA HOMOLOGA: (botón+efecto)=<ui#handler> ↔ (flecha+destino)=<id>"
  - "MEZCLA: (repo@(s+i) + img@(s+i)) weights=[w1,w2]"
  - "COHERENCIA: README/contratos ↔ diagrama/leyendas"
  - "METAFORAS: propone+valida k=3 comunes con evidencia"

output.schema:
  canonical:
    - CNode(code)?
    - CNode(image)?
    - CNode(mixed)?
```

```
  verdicts:
    - SIVerdict(code)?
    - SIVerdict(image)?
  traces:
    - Trace(code)?     # UI→API→DB
    - Trace(image)?    # flecha→destino
  reports:
    - MixReport?
    - coherence_gaps?: [string]
    - next_actions?: [string]

merge.policy:
  precedence: [by_weight, by_specificity, by_recency]
  conflicts:
    - {type: "definition_conflict", rule: "split-view o choose-and-record"}
    - {type: "edge_conflict",       rule: "normalizar y etiquetar dirección"}
    - {type: "contract_conflict",  rule: "preferir contrato más específico; registrar el otro"}

quality_signals:
  - traceability.evidence_links
  - coherence.(s↔i)
  - boundary_explicitness
  - actionability.next_actions

examples:
  - cmd: "ANALOGIAS: mapa (s+i) código↔imagen usando repo://app img://diagram.png"
    returns: "tabla S/I de correspondencias con evidencia (paths y bboxes)"
  - cmd: "TRAZA HOMOLOGA: (botón+efecto)=ui/BookBtn.tsx#onClick ↔
(flecha+destino)=F1"
    returns:
      Trace(code): {from:"onClick", through:["controller","service","repo"], to:"db", evidence:[...]}
      Trace(image): {from:"flecha F1", through:["caja A","caja B"], to:"caja C",
evidence:[bbox...] }
  - cmd: "MEZCLA: (repo@(s+i) + img@(s+i)) weights=[0.7,0.3]"
    returns:
      CNode(mixed): {contexto, nodes, metaforas, edges, evidence, provenance}
      MixReport: {inputs:["repo","img"], decisions:[...], coverage:...}
```