```
{
  "cognitive_node": "meta-metafora@1.0.0",

  [reason-for-creating-definition: "Elevar la METÁFORA desde comparación retórica a artefacto EJECUTABLE que preserve estructura entre dominios vecinos y pueda integrarse a otros nodos."]
  [ideas-involved: [
    {ideaName: "vecindad", ideaConcept: "vocabulario mínimo común para transferir meta-información"},
    {ideaName: "anchors/edges", ideaConcept: "anclas observables y relaciones (composición, vecindad, orden)"},
    {ideaName: "structure-mapping", ideaConcept: "alineación de subestructuras homólogas"},
    {ideaName: "adapter", ideaConcept: "reglas ejecutables from→to que preservan constraints"},
    {ideaName: "fitness predictivo", ideaConcept: "validación empírica del mapeo sobre datos reales"},
    {ideaName: "provenance", ideaConcept: "trazabilidad de decisiones para auditoría"}
  ]]

  "orchestrator": {
    "name": "meta-metaphor.orchestrator",
    "describe_how_the_nodes_interact": [
      "1) Intake: recibir (a) metaphor_spec o (b) par de esquemas origen/destino + sus anchors/edges.",
      "2) Vecindad: verificar vocabulario mínimo común; si falta, sugerir alias_map.",
      "3) Alineación: proponer candidate_mappings vía structure-mapping sobre anchors/edges.",
      "4) Constraints: filtrar mappings que NO preserven composición, vecindad y orden.",
      "5) Síntesis: compilar metaphor_map (adapter_spec) con reglas from→to y alias_map.",
      "6) Evaluación: medir fitness_score del adapter en casos reales (predicciones vs. verdad).",
      "7) Emisión: publicar artifacts (metaphor_map.json, diagnostics, provenance) y delta para integration_profile.",
      "8) Bucle: si fitness<umbral, ajustar alias_map/anchors/constraints y re-intentar (hasta convergencia)."
    ],
    "policy": {
      "preserve": ["composition", "adjacency", "ordering", "type-role"],
      "evidence_priority": ["perception>textual", "estructura>palabras", "hechos>asociaciones"],
      "fitness_threshold": 0.75,
      "provenance": true
    }
  },

  "nodes": [
    {
```

```
    "id": "input_hub",
    "what": "Recolecta entradas: metaphor_spec O {schema_origen, schema_destino,
anchors, edges}.",
    "in": ["metaphor_spec | schemas+anchors+edges"],
    "out": ["workset"]
  },
  {
    "id": "shared_vocabulary_checker",
    "what": "Comprueba vecindad y propone alias_map cuando hay desalineación léxica.",
    "in": ["workset"],
    "out": ["alias_map", "vecindad_ok:boolean"]
  },
  {
    "id": "alignment_engine",
    "what": "Alinea subestructuras (structure-mapping) usando anchors/edges y alias_map.",
    "in": ["workset", "alias_map"],
    "out": ["candidate_mappings"]
  },
  {
    "id": "mapping_constraints",
    "what": "Valida que los mapeos preserven composición, vecindad y orden.",
    "in": ["candidate_mappings"],
    "out": ["validated_mappings"]
  },
  {
    "id": "adapter_synthesizer",
    "what": "Compila adapter ejecutable: metaphor_map.json (reglas from→to, alias_map,
constraints).",
    "in": ["validated_mappings", "alias_map"],
    "out": ["metaphor_map.json"]
  },
  {
    "id": "predictive_evaluator",
    "what": "Evalúa poder predictivo del adapter contra casos reales del dominio destino.",
    "in": ["metaphor_map.json", "workset"],
    "out": ["fitness_score", "diagnostics"]
  },
  {
    "id": "provenance_tracker",
    "what": "Anota fuentes, decisiones y versiones para auditoría.",
    "in": ["validated_mappings", "metaphor_map.json", "diagnostics"],
    "out": ["provenance.log"]
  },
  {
    "id": "integration_profile_emitter",
    "what": "Genera delta para export_surface/adapters en el contenedor que consuma el
adapter.",
    "in": ["metaphor_map.json", "provenance.log"],
```

```
      "out": ["integration_profile.delta.json"]
    }
  ]

  // (opcional) export_surface conciso si lo necesitas junto a otros nodos
  "export_surface?": {
    "PROVIDES": [
      {"name": "metaphor_adapter", "kind": "mapper", "version": "1.0.0", "types_used":
["Anchor","Edge","MetaphorMap"]}
    ],
    "REQUIRES": [
      {"name": "schemas+anchors+edges", "kind": "analyzer", "version_range": "^0.3.0"}
    ],
    "CONSTRAINTS":
["deterministic","preserve:composition","preserve:adjacency","preserve:ordering"],
    "TYPES": ["MetaphorSpec","MetaphorMap","AliasMap"]
  }
}
```