ELEMENTO: "Mecanismo de Conexión Inferida (BridgeCraft v1)"

contexto: {
  propósito: "Permitir al sistema proponer y ejecutar conexiones entre módulos/procesos aunque el usuario no las explicite, manteniendo trazabilidad, límites y validación.",
  cuándo_usarlo: [
    "La entrada no está en forma canónica requerida por un módulo de salida.",
    "Existen 2+ módulos que, combinados, satisfacen la intención del usuario.",
    "Hay hueco operativo (p. ej., falta 'pre-procesar' antes de 'construir estructura')."
  ],
  principios: [
    "Parcimonia (usar el menor número de saltos).",
    "Compatibilidad formal (forma/ritmo/semántica) entre módulos.",
    "No alucinar: toda conexión debe estar justificada con reglas explícitas o invariantes.",
    "Trazabilidad: cada puente deja rastro (provenance) y métrica de confianza."
  ],
  entradas: [
    "intención_usuario (texto/artefacto)",
    "catálogo_modular (capacidades, requisitos, pre/post-condiciones)",
    "estado_actual (forma del insumo, validaciones, errores)"
  ],
  salidas: [
    "plan_conexión (pipeline inferido con pasos y contratos)",
    "justificación (reglas/usos previos que avalan la conexión)",
    "confianza∈[0..1] + criterios de rollback",
    "artefacto_resultante en forma canónica"
  ]
}

nodes: [

  { id: "detectar-brecha",
    rol: "Descubrir incompatibilidad",
    acción: [
      "Comparar requisitos del módulo-objetivo con la forma del insumo.",
      "Si falta una precondición conocida (ej. forma canónica), registrar 'brecha'."
    ],
    salida: "brecha={requisito_faltante, módulo_objetivo}"
  },

  { id: "mapear-puentes-posibles",
    rol: "Búsqueda en catálogo",
    acción: [
      "Consultar catálogo_modular para módulos que 'satisfagan' la brecha.",
      "Filtrar por compatibilidad: forma, ritmo, semántica, costo."
    ],
    salida: "candidatos=[{módulo, pre→post, evidencias}]",
    requiere: ["detectar-brecha"]

```
  },

  { id: "abducir-conexión-minima",
    rol: "Elegir el puente más simple",
    acción: [
      "Ordenar candidatos por parcimonia (nº saltos) y cobertura de requisitos.",
      "Proponer pipeline mínimo (ej.: de-pre-procesar-a-estructura →
constructor-de-estructuras)."
    ],
    salida: "hipótesis_puente={pipeline, contrato, costo_estimado}",
    requiere: ["mapear-puentes-posibles"]
  },

  { id: "validar-invariantes",
    rol: "Prueba de seguridad",
    acción: [
      "Chequear invariantes globales: no romper composición/adyacencia/orden; no perder
provenance.",
      "Simular post-condiciones del módulo objetivo con el resultado del puente."
    ],
    salida: "validez={ok|fail, detalles}",
    requiere: ["abducir-conexión-minima"]
  },

  { id: "estimar-confianza",
    rol: "Medición",
    acción: [
      "Calcular score por evidencia (regla explícita>ejemplo previo>analogía).",
      "Penalizar saltos adicionales/adaptadores frágiles."
    ],
    salida: "confianza∈[0..1], riesgos[]",
    requiere: ["validar-invariantes"]
  },

  { id: "ejecutar-o-escalar",
    rol: "Decisión",
    política: [
      "Si confianza≥0.7 y validez=ok → ejecutar puente y continuar.",
      "Si 0.4≤confianza<0.7 → ejecutar en sandbox y pedir confirmación breve.",
      "Si <0.4 o validez=fail → no ejecutar; solicitar instrucción explícita."
    ],
    salida: "plan_conexión, justificación, criterios_rollback",
    requiere: ["estimar-confianza"]
  },

  { id: "trazabilidad-y-rollback",
    rol: "Provenance y rescate",
    acción: [
```

```
      "Guardar: {intent, brecha, pipeline, reglas, confianza, artefactos}.",
      "Proveer comando de rollback para deshacer cambios si el usuario lo indica."
    ],
    salida: "registro_conexión (persistido)",
    requiere: ["ejecutar-o-escalar"]
  }
]

metáforas: [
  { nombre: "Herrero de Puentes",
    definición: "Forjar el eslabón faltante mínimo que convierte lo que tienes en lo que tu
destino requiere, sin quebrar el puente que ya existe."
  }
]

reglas_operativas: {
  "no_alucinar": "ningún paso sin regla o precedente; si no existe, escalar a confirmación.",
  "parcimonia": "preferir 1 puente a 2; 2 a 3, etc.",
  "forma_canónica": "si output final no es {contexto,nodes,…} → aplicar
constructor-de-estructuras.",
  "provenance": "toda conexión deja log con fuentes/reglas/métricas.",
  "safe_exit": "si cualquier invariante falla → abortar y reportar brecha."
}

ejemplo_instanciado: {
  brecha: "El análisis no está en forma canónica requerida por el constructor.",
  puente_elegido: ["de-pre-procesar-a-estructura", "constructor-de-estructuras"],
  confianza: 0.82,
  justificación: "Reglas de precondición + precedentes previos.",
  resultado: "Cognitive Node canónico listo para integración."
}
```