Constraint-Driven Abduction (CDA) — Technical Spec

Status: MVP — implementation-ready

Audience: Internal (Metis Core + Daedalus partner profile)

Container: Conversation (Canvas-first)

Transports: WebSocket ToolBus (control) · **SSE** (streams). **No REST anywhere.**

Contracts: ToolBus **JSON schemas** (in/out) + **SSE event names** only.

> **Terminology refresh: Actions** replace Work Units (WUs). **Abilities** are action types. CDA is invoked primarily as an **Action** with `ability:"cda"`. There is **no public** `cda.*` **ToolBus namespace** for partners (internal-only debug exists behind flags).

---

0) Purpose & Scope

CDA is a bounded, auditable reasoning engine for forming the **best explanatory hypotheses** under explicit **constraints** given observations and goals. It orchestrates iterative **hypothesis generation → constraint propagation → verification → plan refinement**. CDA prioritizes **safety, determinism, and boundary-only disclosure** (no raw chain-of-thought to clients).

Use cases (examples): root-cause analysis, multi-step verification of claims, plan synthesis under conflicting evidence, policy compliance checks, red-team scenario analysis.

Non-goals (MVP): unconstrained creative brainstorming (use Compose/Generative), unrestricted web crawling (use Research with approvals), public REST.

---

1) Mental Model

- **Inputs:** observations `O`, constraints `C` (hard/soft), goals `G` (prove/disprove, best explanation, feasible plan), background knowledge via **Ken**.
- **Outputs:** a **bounded hypothesis set** `H*` with scores/confidence, **boundary summaries**, **verifier justifications**, and **next-best tests**.

- **Search:** seeded, deterministic exploration of hypothesis space with pruning by constraints and verification feedback.
- **Disclosure:** only **boundary summaries** and artifact IDs are streamed to clients; raw traces stay internal behind Reflection (gated).

---

2) Interfaces

2.1 Action Inputs

```
CDAInputs {
  problem: string,                     // natural-language statement of the abductive task
  observations: SpanRef[] | ArtifactRef[],
  goals: { kind: "best_explanation|prove|disprove|plan", success_criteria?: string[] },
  constraints: {
    hard?: string[],                   // DSL clauses (see §3)
    soft?: {clause: string, weight?: number}[]
  },
  policies?: {pii_posture?: "on|off|consent", egress?: "local|partner|external"},
  budgets?: {tokens?: number, wall?: number},
  risk_class?: "low|medium|high|hard",   // hard ⇒ escalate immediately
  seed?: {purpose?: string}
}
```

2.2 Action Outputs

```
CDAOutputs {
  summary_artifact_id: string,         // boundary summary document
  plan_bundle_id?: string,             // cda_plan (bundle) with machine-readable plan graph/constraints
  verdicts: Array<{hypothesis_id:string, status:"supported|refuted|inconclusive", confidence:number}>,
  next_tests?: Array<{question:string, suggested_ability?:"research|sandbox|default", expected_discriminative_power?:number}>,
  evidence_bundle_id?: string          // citations/SpanRefs for verifier
decisions
}
```

2.3 DisplayPlan (client hints)

```
DisplayPlan {
  primary: {kind:"summary", id_ref:"summary_artifact_id"},
  chips: ["receipts", "context_pack"],
  drawers: {integrity:true}
}
```

---

3) Constraint DSL (MVP)

A compact, auditable DSL for **hard** and **soft** constraints. Parsed and validated server-side; logged in receipts.

Forms (allowed):

- **Boolean predicates:** `holds(pred(args))`, `not(pred(args))`.
- **Relational over Ken:** `exists path(entity:A ->related_to-> entity:B within 2 hops)`.
- **Temporal ordering:** `before(event:X, event:Y)`, `within(event:X, 14d of event:Y)`.
- **Mutual exclusivity:** `xor(hyp:A, hyp:B)`.
- **Numeric bounds:** `value(metric:Cost) <= 5000`.
- **Selection policy:** `use(provider:ModelX) only_if risk <= medium` (interpreted by Router caps).
- **Soft clause:** `prefer(pred(args)) weight 0.4`.

Safety: only the **safe query set** to Ken is permitted (`search|neighborhood|path|related_by`); any external egress triggered by a clause requires approvals-as-input. PII posture gates predicates over personal data.

---

4) Algorithm (Deterministic, Budget-Aware)

1) **Initialize** frontier from observations `O` using seeded generators (deterministic enumeration → beam of candidates).
2) **Propagate constraints**: apply hard constraints; compute penalties for soft constraints.
3) **Verify**: run verifiers (LLM fact-checkers, symbolic checkers, typed functions/tests) & **Ken** queries; attach **citations (SpanRefs)**; emit `action_checkpoint(kind:"verifier_update")`.
4) **Refine plan**: expand or prune hypotheses; emit `action_checkpoint(kind:"plan_delta")` with boundary-safe rationale.
5) **Context updates**: when adopting new evidence, emit `action_checkpoint(kind:"context_delta")` and update the active **context pack**.
6) **Converge**: stop when success criteria met, budget exhausted, or frontier stable; produce **boundary summary** + **next-best tests**.

Search control: beam width, depth caps, and reranker settings are deterministically derived from **hmac-v1** seeds and budgets.
Escalation: may spawn sub-Actions (`research`, `sandbox`, or `default`) via edges `dependsOn|waitFor|bind|notify`; outputs are bound as locked dependencies.

---

5) Verification Suite (MVP)

- **LLM verifiers**: cross-model agreement, self-consistency checks, claim-level citation enforcement.
- **Symbolic verifiers**: constraint satisfiability, unit checks, temporal logic evaluation.
- **Function evaluators**: deterministic code tests in **Sandbox** (egress denied unless approved).
- **Ken evidencing**: `search|neighborhood|path|related_by` with receipts; Weave supplies span pointers.
- **Scoring:** `score = salience + posterior_support - penalties(violations) - uncertainty`; confidence bands derived from calibration tables.

---

6) Privacy, Egress & Reflection

- **Boundary-only**: clients receive fixed-size boundary summaries; raw traces accessible only via **Reflection** (read-only, approval-gated).
- **PII posture:** `on|off|consent`; when **on**, external provider calls are redacted; concept-only

embeddings allowed with approval.

- **Egress classes:** `local|partner|external`; Router enforces caps and approvals.
- **Audit/Receipts:** every verifier and Ken call logs a **receipt** with seeds, caps, and citations; WORM-logged.

---

7) Public Contracts

7.1 ToolBus (WS)

- **No public** `cda.*` for partners. Invoke via `actions.start{ ability:"cda", inputs:CDAInputs }`.
- **Sub-abilities**: CDA may call `research.*`, `sandbox.*`, and `ken.*` through the orchestrator.
- **Reflection**: `reflect.*` (read-only) for internal reviewers (approval-gated).

7.2 SSE Events (Streams)

- **Actions:** `action_started, action_progress, action_notice, action_waiting_user, action_linked, action_checkpoint(kind:"boundary_summary|plan_delta|verifier_update|context_delta"), action_completed, action_partial, action_failed`.

Cursor resume: by `Last-Event-ID`; ≥24h retention per topic.

---

8) Artifacts & Display

- **Boundary summary (primary)**: short document with decisions, key evidence citations, and residual uncertainty.
- **Plan bundle (**`cda_plan`**)**: **bundle** artifact containing `{hypotheses.json, constraints.json, plan_graph.json, verifiers.json, receipts/}` (no raw CoT).
- **Evidence bundle**: citations with SpanRefs and excerpts; missing evidence yields **receipt stubs**.
- **DisplayPlan** emphasizes the boundary summary; Plan bundle & receipts are available via chips/drawers.

9) CUP → CDA Escalation & Budgets

- **Triggers:** escalate to CDA when **any 2 of 5** uncertainty/risk signals fire (e.g., contradictions in evidence, high doubt, low confidence band, policy risk, critical decision), or **immediately** for `risk_class:"hard"`.
- **Budget reservation:** CDA should reserve ~**20%** of wall time (≥ **20 s**) upon escalation; if less remains, emit **partial** with explicit caveats.
- **Grace → freeze → partials** applies on budget breach; **Incomplete** badge shown in Canvas.

---

10) Observability & SLOs

- **Metrics:** hypothesis frontier size, constraints satisfied/violated, verifier pass rate, citation coverage, Ken query p95, TTFS (time-to-first summary), partial ratio, approvals latency.
- **Targets (MVP):** TTFS ≤ **3.0 s** for boundary summary draft; citation coverage ≥ **95%** for supported claims; Ken p95 ≤ **700 ms** @ top-50.
- **Degradation ladder:** shrink frontier/beam → simplify constraints → reduce external calls → return conservative boundary summary with next tests.

---

11) Acceptance Criteria (MVP)

1) CDA runs as an **Action** (`ability:"cda"`), streaming **action_checkpoint** events with kinds `boundary_summary|plan_delta|verifier_update|context_delta`.
2) **No public** `cda.*` for partners; all invocations go through `actions.start`.
3) **Boundary-only** client output; raw traces accessible via **Reflection** (approval-gated).
4) **Ken** integration uses only the safe set; every query and verifier call emits **receipts** with seeds and citations; Weave span pointers present or stubs recorded.
5) **Bind ⇒ lock** when consuming sub-Action outputs; **update-available** bump produces a **structure revision** on the consumer.
6) **CUP escalation** rules enforced; ~20% wall reserved; partials correctly labeled on breach.
7) **PII posture** and **egress caps** honored; approvals-as-input audited.
8) **ToolBus + SSE only**; cursor resume works across deploys.

9) SLOs in §10 met on staging.

---

12) QA Scenarios & Runbooks

1) **Ambiguous claim**: CDA receives conflicting observations; emits verifier_update → plan_delta cycles; produces boundary summary with confidence bands and next tests.
2) **Bind bump**: CDA spawns Research Action; later revision of evidence triggers **update available**; accepting bump yields structure revision; receipts updated.
3) **Denied egress**: constraint requires external check; approval denied; CDA returns conservative boundary summary and marks missing evidence with receipt stubs.
4) **Budget exhaustion**: CDA reserves 20%; if consumed, emits terminal **partial** with explicit caveats and next tests.
5) **Cursor loss**: SSE reconnect with `Last-Event-ID`; timeline consistent.

---

13) Compatibility & Migration (WU → Actions)

- Replace all mentions of **CDA WU** with **CDA Ability** running as an **Action**.
- Legacy `wu_*` event consumers must switch to `action_*`; optional aliases may exist for one CalVer window (off by default for partners).
- Artifacts referencing `work_unit_id` map to `action_id` and are preserved under `legacy.aliases[]` in receipts.

---

14) Open Questions (track to backlog)

- Constraint DSL evolution: add quantifiers/aggregation safely or keep pattern-based subset?
- Learned scoring vs rule-based calibration for confidence bands.
- Canonical schema for **next-best test** utility estimates across abilities.
- Partitioned caching of partial hypothesis frontiers for warm resumes.

#ideas/metis