CAD-LADDER: A SOURCE-GROUNDED BENCHMARK FOR TEXT-TO-CAD MODELS AND AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating CAD models by synthesizing executable code from natural-language descriptions is promising, but evaluation depends on whether each description and reference program denote the same geometry. Misaligned pairs can penalize faithful systems and obscure differences among models and agents. We introduce CAD-Ladder, a source-grounded annotation method and benchmark that derives five increasingly abstract descriptions through sequential code-to-language rewriting from execution-verified CadQuery programs. Semantic auditing checks whether the reference solid satisfies the description's geometric constraints and estimates essential source-constraint retention. Repeated blind reconstruction measures geometric fidelity and inter-run agreement through deterministic geometric scoring. CAD-Ladder achieves a semantic consistency rate of 77.9%, compared with 42.5% for CADPrompt. Audit-guided revision of 38 descriptions that fail the semantic criterion raises their joint reconstruction pass rate from 55.3% to 73.7%. Applying CAD-Ladder's annotation principles to the same 200 CADPrompt targets increases mean reconstruction IoU from 0.7748 to 0.9295. The resulting benchmark separates standalone LLM and agentic configurations on T1 at L5, with delivery quality ranging from 49.7% to 95.1%.

1 INTRODUCTION

Generating executable parametric CAD programs from natural-language descriptions is a challenging form of code generation. Unlike a mesh, a CAD program records an editable construction history of sketches, constraints, features, and boolean operations (Willis et al., 2021). Existing approaches either train domain models to emit quantized CAD sequences or executable python programs (Wu et al., 2021; Khan et al., 2024; Rukhovich et al., 2025; Doris et al., 2025; Kolodiazhnyi et al., 2026), or place a general-purpose LLM in an agentic generate–execute–inspect–repair loop (Yao et al., 2023; Badagabettu et al., 2024; Mallis et al., 2025; Gu et al., 2025). Evaluating either approach requires descriptions that faithfully specify their reference geometry.

The reliability of these text–code pairs directly affects evaluation but is rarely measured. If a description and its reference program specify different solids, a system that follows the text can be penalized for disagreeing with the reference geometry, while an unfaithful system may score better. Machine annotation scales but may lose geometric constraints (Khan et al., 2024; Doris et al., 2025), while human authorship and review do not guarantee consistency. For example, CADPrompt provides 200 human-authored and reviewed tasks (Alrashedy et al., 2025); in our source-grounded semantic audit, only 42.5% of its descriptions meet the joint criteria for low conflict and essential constraint retention. This motivates both source-grounded annotation and explicit measurement of text–code consistency.

We introduce **CAD-Ladder**, an annotation method and benchmark that begins with an execution-verified CadQuery program and its STEP solid. It decomposes code-to-language translation into five sequential rewrites: L1 annotates the code; L2 removes executable syntax; L3 fuses operations into parametric prose; L4 primarily replaces absolute placement with relative spatial relations; and L5 compresses dense realization detail into a compact engineering specification. This decomposition reduces the scope of each rewrite and produces paired descriptions for evaluating different representations of the same target.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

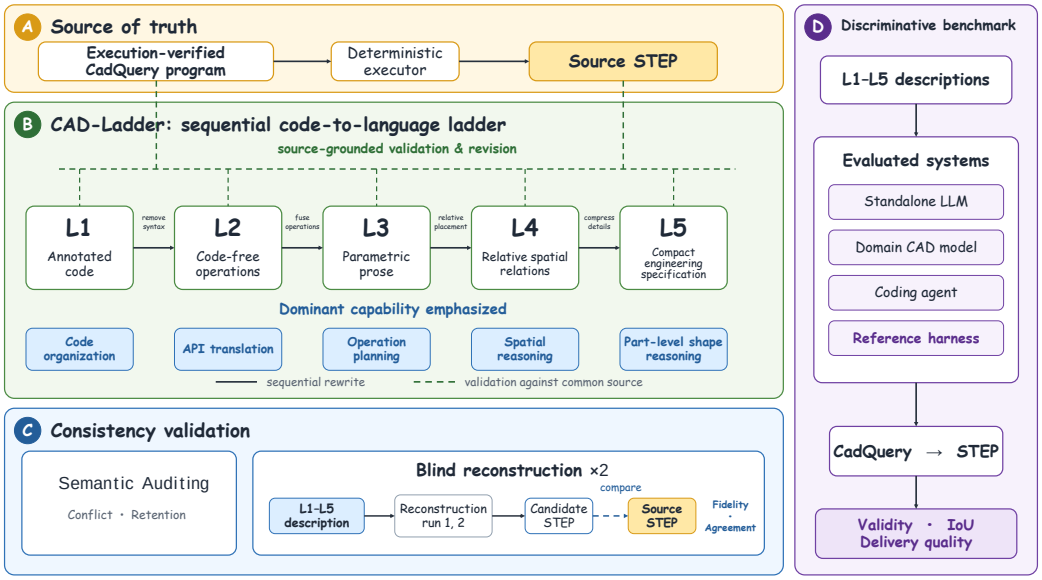


Figure 1: **Overview of CAD-Ladder.** (A) An execution-verified CadQuery program and its STEP solid provide a common source of truth. (B) Sequential L1–L5 rewriting produces increasingly abstract descriptions, each emphasizing different modeling capabilities. (C) Semantic auditing assesses conflict and core-information retention; repeated blind reconstruction measures fidelity and inter-run agreement. (D) The benchmark compares domain models, standalone LLMs, coding agents, and a reference harness through executable CAD outputs, measuring validity, IoU, and delivery quality.

Source provenance alone does not prevent rewriting errors. Two complementary methods assess and improve these descriptions. *Source-grounded semantic auditing* checks stated geometric requirements and estimates how much core source information the text retains. *Repeated blind reconstruction* measures whether the description supports accurate and repeatable recovery of the solid. Together, they connect constraint-level judgments with geometric reconstruction. Audit feedback identifies conflicting or missing information for revision, whose effect is measured by blind reconstruction.

CAD-Ladder contains 1,054 verified programs and 5,270 descriptions, divided into 306 fixed memory entries and 748 evaluation tasks. On the 172 evaluated L5 descriptions, CAD-Ladder achieves a semantic consistency rate of 77.9%, compared with 42.5% for CADPrompt. Audit-guided revision raises the reconstruction pass rate on 38 descriptions that fail the semantic criterion from 55.3% to 73.7%. Applying CAD-Ladder's annotation principles to the same 200 CADPrompt targets raises mean IoU from .7748 to .9295 with slightly shorter descriptions. The benchmark also distinguishes model and agent configurations: on T1 at L5, delivery quality is 49.7% for standalone DeepSeek V4 Flash, 70.4–79.5% for commercial coding agents, and 95.1% for DeepSeek V4 Flash with the reference harness.

Our contributions are:

- **CAD-Ladder construction and corpus:** a sequential, source-grounded method for deriving five abstraction levels from verified CAD programs, producing 1,054 programs and 5,270 texts.
- **Description validation and correction:** complementary semantic auditing and repeated blind reconstruction that measure description quality and support targeted revision.
- **A discriminative Text-to-CAD benchmark:** paired L1–L5 evaluation of domain models, general LLMs, coding agents, and a lightweight reference harness, with delivery-quality and tool-ablation analyses.

2 Related Work

Programmatic CAD generation. Large-scale CAD resources have progressed from boundary-representation collections such as ABC to datasets containing sketches, constraints, and human construction histories, including SketchGraphs and the Fusion 360 Gallery (Koch et al., 2019; Seff et al., 2020; Willis et al., 2021). These resources support sequence-based generative models such as DeepCAD and SkexGen (Wu et al., 2021; Xu et al., 2022). Later work expanded the input and output representations. CAD-Recode maps point clouds to executable Python CAD programs (Rukhovich et al., 2025), while Text2CAD conditions CAD-sequence generation on natural-language descriptions (Khan et al., 2024). CAD-Coder generates CadQuery programs and improves geometric fidelity through supervised training (Doris et al., 2025); Text-to-CadQuery similarly targets direct synthesis of executable CadQuery programs (Xie & Ju, 2025). Cadrille further unifies text, image, and point-cloud reconstruction within an executable-code formulation (Kolodiazhnyi et al., 2026). These methods study how models decode a given input into CAD; CAD-Ladder studies whether the textual input and executable target are aligned in the first place.

Text-conditioned CAD data. Text2CAD scales annotation by using language and vision-language models to describe the DeepCAD corpus at multiple detail levels (Khan et al., 2024). Other recent systems construct large text-code-geometry corpora for supervised and reinforcement learning (Wang et al., 2025b; Doris et al., 2025; Kolodiazhnyi et al., 2026). CADPrompt instead provides 200 natural-language tasks paired with expert-authored programs (Alrashedy et al., 2025). Existing datasets therefore cover useful trade-offs between scale, automatic annotation, and human effort, but generally treat the resulting description as correct once constructed. CAD-Ladder derives five abstraction levels sequentially from each verified source program and evaluates description quality through semantic auditing and blind reconstruction.

Agentic CAD generation. General agent benchmarks evaluate LLMs in interactive environments (Liu et al., 2024) and increasingly expose intermediate progress, tool use, and repeated-task reliability rather than reporting only final success (Ma et al., 2024; Huang et al., 2023). Query2CAD generates executable macros and refines them using visual and optional human feedback (Badagabettu et al., 2024). CADCodeVerify renders generated programs and uses vision–language validation questions to produce corrective feedback (Alrashedy et al., 2025). CAD-Assistant formulates modeling as tool-augmented planning over FreeCAD, rendering utilities, and a Python interpreter (Mallis et al., 2025). More broadly, BlenderGym demonstrates the importance of generation and verification compute in code-based 3D graphics editing (Gu et al., 2025). Our reference harness follows this tool-augmented paradigm but is used only to probe benchmark discriminativeness.

Executable and geometric validation. Executable code benchmarks commonly evaluate functional correctness by running generated programs against deterministic tests (Chen et al., 2021; Liu et al., 2023). CAD systems are commonly evaluated by program validity and geometric agreement with a reference, using metrics such as invalidity rate, Chamfer distance, Hausdorff distance, and volumetric IoU (Wu et al., 2021; Khan et al., 2024; Rukhovich et al., 2025; Doris et al., 2025; Kolodiazhnyi et al., 2026). Agentic approaches additionally use rendered or vision-language feedback during inference (Badagabettu et al., 2024; Alrashedy et al., 2025; Mallis et al., 2025; Gu et al., 2025). CAD-Ladder examines the description quality: semantic auditing checks stated constraints and core-information retention against source geometry, while repeated blind reconstruction measures accurate and consistent recovery.

3 Method

Figure 1 summarizes CAD-Ladder: verified code and geometry provide a common source; five sequential rewrites form the description ladder; two complementary instruments check text-source consistency; and the resulting benchmark compares models and agents.

3.1 Task and Source Programs

Let $\mathcal{C}$ be the space of CadQuery programs and $\mathcal{M}$ the space of valid solids. A deterministic executor $\mathcal{E}$ runs a program and exports its STEP geometry; non-empty solid creation, or failure to

export STEP is denoted by $\mathcal{E}(c) = \perp$. Each item i begins with a verified program $c_i^\star$ from Bench-CAD (Zhang et al., 2026), executed in a fixed CadQuery environment to obtain its STEP solid:

$$M_i^\star = \mathcal{E}(c_i^\star). \tag{1}$$

In the standard Text-to-CAD setting, a system $\mathcal{A}$ receives one description $x_i^{(\ell)}$ and produces

$$\hat{c}_i^{(\ell)} = \mathcal{A}(x_i^{(\ell)}), \qquad \hat{M}_i^{(\ell)} = \mathcal{E}(\hat{c}_i^{(\ell)}). \tag{2}$$

The generated program need not reproduce the source operation sequence; we evaluate whether it executes and realizes the target geometry. Except for the explicitly defined memory conditions below, evaluated systems cannot access $c_i^\star$, $M_i^\star$, or annotation-time diagnostics.

3.2 Sequential Description Ladder

CAD-Ladder decomposes code-to-language annotation into local rewrites. Starting from $x_i^{(0)} = c_i^\star$, each rewrite edits the preceding representation while consulting the source program:

$$x_i^{(\ell)} = \mathcal{R}_\ell\left(x_i^{(\ell-1)}; c_i^\star\right), \qquad \ell = 1, \ldots, 5. \tag{3}$$

The five levels remove successive layers of realization detail (Figure 1). *L1* annotates the program in place, exposing its API calls, arguments, and geometric effects. *L2* removes executable syntax but retains an ordered operation description. *L3* fuses operations into parametric prose and may replace direct constants with arithmetically derived spans, offsets, or overlaps. *L4* primarily replaces absolute placement with quantified relations among named entities, such as coaxiality, embedding, and protrusion. *L5* compresses dense construction detail, especially free-form profile vertices, into a compact part-level engineering specification. The transitions emphasize code organization, API translation, operation planning, spatial reasoning, and part-level shape reasoning, respectively.

Construction checks address residual code syntax, unsupported claims, and unintended information loss against the fixed source. The following audits evaluate the resulting descriptions, which are frozen before blind reconstruction.

3.3 Source-Grounded Semantic Auditing

Semantic auditing asks whether the constraints stated in a description match the reference geometry and whether the description retains core geometric information. We use two isolated LLM contexts: a source context derives core facts from the reference alone, while a judgment context decomposes the text and checks its constraints against the executed solid.

The source context receives the reference program, its BRep, and a 3D viewer, but cannot access the description. It constructs a core-fact inventory $\mathcal{K}_i^{\text{core}}$ covering the main shape, hole and slot counts, important dimensions, and component connections, with geometric evidence for each fact. This inventory is fixed independently of the text, so that the same inventory is used when comparing different abstraction levels.

The judgment context initially receives only the description. It decomposes the text into independently checkable constraints $q = (e, p, v, s, a)$, recording the subject e, predicate p, expected value or relation v, scope and qualifiers s, and exact source span a. Once these constraints are frozen, the same judgment context gains access to the reference code, BRep queries, and the 3D viewer. It can rotate, zoom, clip, select, and highlight geometry, or query dimensions, faces, and distances. For each constraint, the LLM locates the relevant geometric subject and records a label with its evidence: **S** for supported; **C** for contradicted; and **U** for undetermined when subject location or measurement evidence is insufficient for either conclusion. An alternative modeling operation is acceptable if it realizes the required geometry. Unit names and global translation or rotation are ignored, while numerical dimensions and relative relations between components are preserved.

Finally, after the constraints and their judgments are fixed, $\mathcal{K}_i^{\text{core}}$ is revealed to the judgment context; this step cannot revise the earlier constraints or labels. For each source fact e, the LLM checks whether the original description correctly expresses it. Let $r(e, x) = 1$ for a correct expression, and zero for an omission or incorrect expression. Retention for each description and its corpus mean are

$$R_{\text{ess},i}^{(\ell)} = \frac{\sum_{e \in \mathcal{K}_i^{\text{core}}} r(e, x_i^{(\ell)})}{|\mathcal{K}_i^{\text{core}}|}, \qquad R_{\text{ess}}^{(\ell)} = \frac{1}{N} \sum_{i=1}^{N} R_{\text{ess},i}^{(\ell)}. \tag{4}$$

For each description i, let S_i, C_i, and U_i be the numbers of constraints assigned to their labels, and let $\mathrm{Conflict}_i = C_i/(S_i + C_i)$ denote its conflict rate among decidable constraints. We report

$$\mathrm{Coverage} = \frac{\sum_i (S_i + C_i)}{\sum_i (S_i + C_i + U_i)}, \quad \mathrm{Conflict} = \frac{\sum_i C_i}{\sum_i (S_i + C_i)}. \tag{5}$$

R_{ess} is an automatically estimated retention rate for core information. Coverage measures how many textual constraints receive a determinate judgment. Conflict measures the proportion of those constraints that contradict the geometry. Based on these metrics, we additionally report

$$\mathrm{SCR}^{(\ell)} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\Big[\mathrm{Conflict}_i < 0.10 \ \wedge \ R^{(\ell)}_{\mathrm{ess},i} > 0.60\Big],$$

$$\mathrm{ER}^{(\ell)} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\Big[\mathrm{Conflict}_i = 0 \ \wedge \ R^{(\ell)}_{\mathrm{ess},i} = 1\Big]. \tag{6}$$

SCR (semantic consistency rate) is the proportion of descriptions whose individual conflict rate is below 10% and core-information retention is above 60%. ER (excellent rate) requires zero conflict and full retention of the assessed core facts.

3.4 Repeated Blind Reconstruction

Further, we use repeated blind reconstruction to test geometric recoverability and the consistency of independent attempts. Each description is given to a fixed reconstructor $\mathcal{B}$ in two isolated contexts:

$$\hat{M}^{(\ell)}_{i,k} = \mathcal{E}\Big(\mathcal{B}_k(x^{(\ell)}_i)\Big), \qquad k = 1, 2. \tag{7}$$

The reconstructor receives only the description and shared task instructions, without the reference program or solid. It generates, executes, and repairs a CAD program within a fixed budget. We voxelize solids at 64^3 resolution and report fixed-frame IoU as well as pose-aligned IoU:

$$\mathrm{IoU}_{\mathrm{align}}(M_a, M_b) = \max_{R \in \mathcal{R}_{24}} \mathrm{IoU}(RM_a, M_b), \tag{8}$$

where $\mathcal{R}_{24}$ contains the 24 axis-aligned cube rotations; invalid outputs receive zero. Fidelity and reconstruction agreement are

$$F^{(\ell)}_i = \max_k \mathrm{IoU}_{\mathrm{align}}(\hat{M}^{(\ell)}_{i,k}, M^{\star}_i), \tag{9}$$

$$A^{(\ell)}_i = \frac{1}{\binom{K}{2}} \sum_{1 \le p < q \le K} \mathrm{IoU}_{\mathrm{align}}(\hat{M}^{(\ell)}_{i,p}, \hat{M}^{(\ell)}_{i,q}). \tag{10}$$

The joint reconstruction pass rate is

$$P^{(\ell)}_i = \mathbf{1}[F^{(\ell)}_i \ge 0.80 \wedge \ A^{(\ell)}_i \ge 0.80]. \tag{11}$$

High F with low A therefore indicates successful but unstable recovery, while high A with low F indicates a systematic miss that warrants inspection. Joint passing identifies faithful and repeatable reconstruction under the fixed protocol.

3.5 Benchmark and Reference Harness

For a generated program, let $v_i = \mathbf{1}[\mathcal{E}(\hat{c}_i) \neq \perp]$. We report validity, geometric IoU, and delivery quality, which assigns zero IoU to invalid outputs.

$$Q_{\mathrm{delivery}} = \frac{1}{N} \sum_{i=1}^{N} v_i \, \mathrm{IoU}(\hat{M}_i, M^{\star}_i) = (1 - \mathrm{IR}) \, \overline{\mathrm{IoU}}_{\mathrm{valid}}. \tag{12}$$

We compare domain-specific CAD models, standalone LLMs, coding agents, and a lightweight reference harness. Its cumulative configurations add program saving (S), kernel execution (R), STEP inspection (I), and retrieval from a fixed 306-entry reference memory (M). This design is related

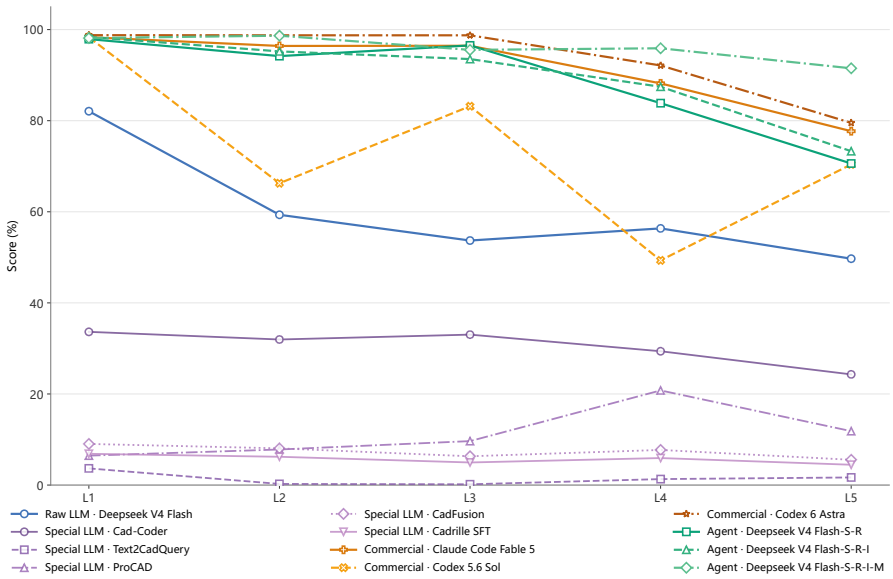


Figure 2: **Delivery quality across CAD-Ladder levels on the T1 stratum.** All points report Q_{delivery}, assigning zero IoU to invalid outputs. Distinct performance profiles separate domain-specific CAD models, a standalone LLM, commercial coding agents, and cumulative configurations of the reference harness. Non-monotonic curves indicate sensitivity to representation.

to retrieval-augmented generation, which externalizes reusable information in a non-parametric store (Lewis et al., 2020). Here, retrieval returns executable CAD precedents rather than factual documents. The bank is frozen before evaluation and never accepts test-time writes. Evaluation tasks are stratified by their relation to the closest memory entry. T0 intentionally contains the exact target program and STEP and hence measures retrieval and reuse rather than generation; T1–T5 require the edit relations defined in §4.1. Main cross-system generation comparisons use T1, while T0–T5 are reported separately to characterize the reference subsystem.

4 Experiments

We answer three questions: **RQ1**, whether sequential abstraction preserves text-source consistency and whether auditing improves description quality; **RQ2**, how does CAD-Ladder compares with CADPrompt and whether its annotation method improves CADPrompt descriptions; and **RQ3**, whether the benchmark distinguishes models and agent configurations.

4.1 Experimental Setup

Data and partitions. CAD-Ladder contains 1,054 verified CadQuery programs and 5,270 level-specific descriptions. We reserve 748 programs and their 3,740 L1–L5 descriptions for evaluation. A separate 306-program partition, also annotated at all five levels, initializes the reference memory. The memory is frozen before evaluation and never receives test-time writes. Tasks are grouped by their relation to the closest memory entry: T0 includes the exact target program and measures retrieval and reuse; T1 changes literals, T2 chains transformations including global pose, T3 requires relative computations, T4 edits features, and T5 rebuilds geometry. We use T1 for the main cross-system comparison and report all six strata separately for the reference harness.

Semantic auditing. The level comparison uses the same 172 T1 targets at L1–L5, sharing one source-derived core-fact inventory. The cross-corpus comparison uses these 172 L5 descriptions and the 200 CADPrompt descriptions. Both corpora use the two-context LLM audit in Section 3.3, with GPT-6 Astra, the same judgment policy, and the same geometric tools.

Table 1: Source-grounded semantic auditing across L1–L5 on the same 172 targets.

Level	n	Coverage (%)	Conflict (%) ↓	R_{ess} (%) ↑	SCR (%) ↑	ER (%) ↑
L1	172	91.6	1.8	97.3	94.7	79.7
L2	172	91.4	1.5	97.0	94.7	77.9
L3	172	91.0	2.8	97.1	94.2	68.0
L4	172	90.5	3.1	95.1	91.9	56.4
L5	172	90.3	3.0	75.3	77.9	17.4

Table 2: Repeated blind reconstruction. The upper block evaluates L3–L5; the lower block compares the 38 L5 descriptions that fail SCR. Pass requires $F, A \geq 0.8$; Low-A counts items with $A < 0.8$.

Description	n	F ↑	A ↑	Low-A ↓	Pass (%) ↑
L3	86	.976	.999	0	97.7
L4	86	.909	.998	1	88.4
L5	172	.798	.964	10	69.2
L5 semantic-fail subset: original	38	.710	.967	2	55.3
L5 semantic-fail subset: revised	38	**.807**	**.995**	**0**	**73.7**

Repeated reconstruction and audit-guided correction. We evaluate the same 86 randomly sampled T1 parts at L3/L4 and all 172 at L5 using two independent reconstructions. GPT-6 Astra receives 600 seconds per reconstruction, with a 120-second limit for generated-program execution. Semantic auditing selects the 38 L5 descriptions that fail SCR as a fixed correction set and identifies conflicting requirements or missing core facts for revision. The revised descriptions then receive the repeated blind reconstructions under the same model, instructions, budget, and scorer.

Same-target CADPrompt reannotation. We hold all 200 CADPrompt reference geometries fixed and compare their published descriptions with CAD-Ladder-guided reannotations. The annotation model applies our abstraction instructions to each reference program to produce a CAD-Ladder-style description, preserving core dimensions, counts, and relations while summarizing construction detail. We report mean reconstruction IoU, the number of targets reaching .90, and mean prompt length in whitespace-separated description words. Each condition uses one blind reconstruction per target with GPT-6 Astra, and both conditions use the same voxel-IoU evaluator.

Systems and generation metrics. We compare five domain-specialized systems: CAD-Coder (Doris et al., 2025), Text-to-CadQuery (Xie & Ju, 2025), ProCAD (Yuan et al., 2026), CAD-Fusion (Wang et al., 2025a), and Cadrille-SFT (Kolodiazhnyi et al., 2026). We additionally evaluate a standalone LLM (DeepSeek V4 Flash, abbreviated DSF), three commercial coding agents (Claude Code with Fable 5, and Codex with GPT-5.6 Sol and Codex with GPT-6 Astra), and cumulative variants of a lightweight reference harness using DSF as the fixed backbone. S, R, I, and M add program saving, kernel execution, STEP inspection, and fixed-memory retrieval, respectively. PJ adds an LLM process judge; MC uses family-first retrieval, and MC1 additionally returns the stored L1 annotation with the reference program. Generation metrics are validity, IoU, and Q_{delivery} (Eq. 12), which assigns IoU zero to invalid outputs, together with token, tool-call, and time costs for the controllable harness.

4.2 RQ1: How Reliable Are CAD-Ladder Descriptions, and Can Auditing Improve Them?

Source consistency across abstraction levels. Table 1 reports level-wise semantic statistics against a shared geometric source. Coverage remains above 90% and fewer than 5% of decidable constraints conflict with the geometry at every level. Core-information retention decreases from 97.3% at L1 to 75.3% at L5, quantifying the information cost of compressing detailed construction descriptions into compact engineering specifications. At L5, 134 of 172 descriptions meet the joint low-conflict and retention criterion (77.9% SCR). These outcomes show that most stated constraints remain compatible with the source.

Table 3: Source-grounded semantic auditing of CAD-Ladder L5 and CADPrompt descriptions with specific measurements. All metrics are percentages.

Corpus	n	Coverage	Conflict ↓	R_{ess} ↑	SCR ↑
CAD-Ladder L5	172	**90.3**	**3.0**	**75.3**	**77.9**
CADPrompt	200	83.1	7.2	60.8	42.5

Table 4: CAD-Ladder-guided reannotation of the same 200 CADPrompt targets. Each condition uses one blind reconstruction per target. Prompt length counts description words only.

Description	n	Mean words	Mean IoU ↑	IoU $\geq .90$ ↑
CADPrompt original	200	90.0	.7748	108/200
CAD-Ladder reannotation	200	85.8	**.9295**	**158/200**

Descriptions support geometric recovery. Table 2 shows the recoverability and stability of descriptions. The L3 and L4 samples achieve mean fidelity .976 and .909 and agreement .999 and .998, respectively. The L5 sample achieves .798 fidelity and .964 agreement, with 119 of 172 descriptions passing the joint criterion (69.2%). These results provide geometric evidence that the descriptions support accurate and repeatable reconstruction across the evaluated formats.

Audit evidence guides effective corrections. On the fixed set of 38 L5 descriptions that initially fail SCR, audit-guided revision raises fidelity from .710 to .807 and pass rate from 55.3% to 73.7%. Agreement also increases from .967 to .995. Thus, the audit supplies actionable corrections that improve reconstruction of difficult descriptions while retaining the same targets.

4.3 RQ2: How CAD-Ladder Compare with CADPrompt?

CAD-Ladder has higher measured annotation quality. Table 3 shows an advantage across all four semantic metrics. CAD-Ladder has higher coverage (90.3% versus 83.1%) and lower conflict among decidable constraints (3.0% versus 7.2%). Its estimated core-information retention is 75.3%, compared with 60.8% for CADPrompt. SCR is 77.9% versus 42.5%, a difference of 35.4 percentage points. Together, these measurements indicate a lower proportion of contradicted requirements and better retention of core geometry in the evaluated CAD-Ladder descriptions.

The annotation advantage transfers to CADPrompt targets. Replacing CADPrompt's published descriptions with CAD-Ladder-guided annotations raises mean reconstruction IoU from .7748 to .9295, a paired increase of .1547 (Table 4). The number of targets reaching .90 increases from 108 to 158, or 54.0% to 79.0%. Mean prompt length decreases from 90.0 to 85.8 words, so higher geometric recovery is achieved with slightly shorter descriptions on average. Holding target geometry and reconstruction conditions fixed links this improvement to the description replacement, supporting the value of CAD-Ladder's annotation method beyond its own choice of parts.

4.4 RQ3: Does the Benchmark Separate Models and Agents?

System classes occupy distinct quality regimes. On T1 at L5, delivery quality spans 1.7–24.3% for the five domain-specific models, reaches 49.7% for the standalone LLM, and ranges from 70.4% to 79.5% for commercial coding agents (Figure 2; Table 5). The DSF backbone reaches 73.3% when placed in the save–run–inspect harness and 95.1% with fixed reference memory. Raw DSF falls from 82.1% at L1 to 49.7% at L5, Claude Code from 96.4% to 77.7%, while Codex 5.6 Sol follows a non-monotonic curve (98.3, 66.3, 83.2, 49.3, and 70.4%). The ladder provides a diagnostic profile, since greater abstraction does not necessarily reduce performance for every system.

Controlled ablations separate backbone and harness gains. Program saving and kernel execution raise the raw backbone from 49.7% to 70.6% delivery and from 73.3% to 95.9% validity. STEP inspection adds 2.7 percentage points while reducing mean tokens and time. Adding a process judge does not improve delivery despite increasing both costs. By contrast, fixed-memory retrieval

Table 5: L5 results. Above the divider, systems are compared on the final T1 stratum ($n = 172$); below, DSF-S-R-I-MC1 is stratified by its edit relation to the closest entry in the fixed reference partition. All process and quality columns use the same final, mutually exclusive task strata. Variants are cumulative: S save, R run, I inspect, and M fixed-reference retrieval. T2 (pose-aligned) rescores the same episodes after rigid alignment, so its process columns repeat. Mean IoU is computed over valid deliveries; median IoU and Q_{delivery} count invalid outputs as zero.

Class	Method	Tokens		Tool calls		Time (s)		Valid (%)	IoU (%)		IoU > 99 (%)	Q_{delivery} (%)
		Mean	Median	Mean	Median	Mean	Median		Mean	Median		
	Claude Code (Fable 5)	–	–	–	–	–	–	100.0	77.7	99.82	52.9	77.7
	Codex 5.6 Sol	–	–	–	–	–	–	100.0	70.4	95.82	41.3	70.4
	Codex 6 Astra	–	–	–	–	–	–	100.0	79.5	100.00	58.1	79.5
	DSF (raw)	5,218	3,290	0	0	47.7	29.2	73.3	67.8	56.38	33.7	49.7
T1	DSF-S-R	213,637	78,305	10.6	8	140.3	84.9	95.9	73.6	99.32	50.6	70.6
	DSF-S-R-I	170,605	45,472	8.4	4	112.6	64.4	99.4	73.7	98.66	48.8	73.3
	DSF-S-R-I-PJ	228,471	64,988	9.0	4	204.5	140.6	98.8	74.5	98.48	49.4	73.6
	DSF-S-R-I-M	81,511	52,351	5.4	5	67.5	46.6	100.0	91.5	100.00	79.1	91.5
	DSF-S-R-I-MC	99,501	58,176	5.6	5	64.0	44.0	100.0	93.5	100.00	79.7	93.5
T0	DSF-S-R-I-MC1	143,860	70,824	5.9	5	75.7	45.2	100.0	96.4	100.00	80.9	96.4
T1	DSF-S-R-I-MC1	103,452	64,412	5.1	4	65.5	41.5	100.0	95.1	100.00	77.9	**95.1**
T2	DSF-S-R-I-MC1	171,787	65,809	6.4	5	71.0	42.4	99.0	15.7	7.58	3.8	15.6
T2 (pose-aligned)	DSF-S-R-I-MC1	171,787	65,809	6.4	5	71.0	42.4	99.0	89.0	100.00	55.8	88.2
T3	DSF-S-R-I-MC1	170,160	77,572	6.3	5	83.5	50.9	100.0	90.5	100.00	71.6	90.5
T4	DSF-S-R-I-MC1	111,106	68,008	5.3	4	59.8	40.5	98.2	91.0	100.00	74.4	89.4
T5	DSF-S-R-I-MC1	167,387	88,053	6.8	5	100.7	79.3	99.0	77.3	92.21	39.6	76.5

raises the S-R-I backbone from 73.3% to 91.5% while reducing both costs; family-first retrieval and returning the stored annotation further raise it to 93.5% and 95.1%.

Relation strata characterize reference-assisted performance. T0 reaches 96.4%, but it is an explicit exact-reuse condition and is not evidence of generation. Aligning the same T2 outputs raises delivery from 15.6% to 88.2%, localizing most of the error to requested global pose rather than intrinsic shape. T5 falls to 76.5% as the retrieved program requires geometry rebuilding. Thus, the fixed memory converts nearby targets into editing problems, but its benefit decreases outside the edit radius of the retrieved reference. Together, the class-level results, controlled harness ablations, and relation strata show that CAD-Ladder distinguishes underlying model capability, execution and inspection feedback, and cross-task reference reuse.

5 Limitations

First, CAD-Ladder currently covers isolated CadQuery/STEP parts rather than assemblies, commercial feature histories, manufacturing tolerances, or other CAD languages. Then, both semantic auditing and repeated blind reconstruction method highly depend on LLM capabilities. Misinterpretations of the descriptions or reference geometry can lead to incorrect judgments or reconstructions, affecting the resulting evaluation. Finally, the 306-entry reference bank is fixed, T0 contains exact references by design, and episodes never write back. The harness therefore measures controlled retrieval and reuse, not continual learning.

6 Conclusion

CAD-Ladder connects source-grounded annotation with explicit evaluation of description quality through semantic auditing and repeated blind reconstruction. Its evaluated L5 descriptions achieve higher semantic coverage, lower conflict, and greater estimated core-information retention than CADPrompt, with semantic consistency rates of 77.9% and 42.5%. Applying CAD-Ladder's annotation method to the same CADPrompt targets raises mean reconstruction IoU from .7748 to .9295 with slightly shorter descriptions. Together with the model and agent evaluations, these results support CAD-Ladder as a benchmark for evaluating executable CAD generation with explicit measurements of the quality of its conditioning descriptions.

AI USE STATEMENT

We used generative AI tools to assist with code generation and debugging, and to improve the wording and clarity of the manuscript. The research ideas, problem formulation, methodological decisions, and experimental design were developed by the human authors. Language models also serve as experimental components for annotation, semantic auditing, and CAD reconstruction, as described in the methods and experiments. The authors reviewed the AI-assisted code and text, checked the experimental data and analyses, and verified the interpretations and conclusions against the experimental evidence. The authors take responsibility for the final manuscript and all associated code, data, claims, and conclusions.

REPRODUCIBILITY STATEMENT

We have publicly released the benchmark data, annotation and evaluation code, prompts, and recorded experimental results. The repository is available at `https://anonymous.4open.science/r/CAD-Ladder-Anonymous`. The package includes the fixed evaluation partitions and reference bank, reference geometry, semantic-auditing instructions and records, reconstruction inputs and generated programs, and paired original and revised descriptions for CAD-Ladder and CADPrompt. Offline scripts recompute the semantic-auditing and reconstruction metrics from saved per-item records without additional model calls; execution and scoring tools support geometric reevaluation. Stable item identifiers, file checksums, recorded model and tool settings, and execution budgets link inputs to results. Module-specific instructions describe data verification, result aggregation, and new experiment runs. Appendices A–G provide the experimental procedures and case records.

REFERENCES

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Haider Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating cad code with vision-language models for 3d designs. In *International Conference on Learning Representations*, volume 2025, pp. 52236–52262, 2025.

Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. Query2cad: Generating cad models using natural language queries, 2024. URL `https://arxiv.org/abs/2406.00144`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Anna C. Doris, Md Ferdous Alam, Amin Heyrani Nobari, and Faez Ahmed. Cad-coder: An open-source vision-language model for computer-aided design code generation, 2025. URL `https://arxiv.org/abs/2505.14646`.

Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas Guibas. Blendergym: Benchmarking foundational model systems for graphics editing, 2025. URL `https://arxiv.org/abs/2504.01786`.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad models from beginner-to-expert level text prompts, 2024. URL `https://arxiv.org/abs/2409.17106`.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9601–9611, 2019.

Maksim Kolodiazhnyi, Denis Tarasov, Dmitrii Zhemchuzhnikov, Alexander Nikulin, Ilya Zisman, Anna Vorontsova, Anton Konushin, Vladislav Kurenkov, and Danila Rukhovich. cadrille: Multi-modal cad reconstruction with reinforcement learning, 2026. URL https://arxiv.org/abs/2505.22914.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-assistant: Tool-augmented vllms as generic cad task solvers, 2025. URL https://arxiv.org/abs/2412.13810.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds, 2025. URL https://arxiv.org/abs/2412.14042.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*, 2020.

Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-cad generation through infusing visual feedback in large language models. In *International Conference on Machine Learning*, pp. 65326–65345. PMLR, 2025a.

Siyu Wang, Cailian Chen, Xinyi Le, Qimin Xu, Lei Xu, Yanzhou Zhang, and Jie Yang. Cad-gpt: Synthesising cad construction sequence with spatial reasoning-enhanced multimodal llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7880–7888, 2025b.

Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4): 1–24, 2021.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models, 2021. URL https://arxiv.org/abs/2105.09492.

Haoyang Xie and Feng Ju. Text-to-cadquery: A new paradigm for cad generation with scalable large model capabilities. *arXiv preprint arXiv:2505.06507*, 2025.

Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *arXiv preprint arXiv:2207.04632*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Bo Yuan, Zelin Zhao, Petr Molodyk, Bin Hu, and Yongxin Chen. Clarify before you draw: Proactive agents for robust text-to-cad generation. *arXiv preprint arXiv:2602.03045*, 2026.

Haozhe Zhang, Kaichen Liu, Miaomiao Chen, Lei Li, Shaojie Yang, Cheng Peng, and Hanjie Chen. Benchcad: A comprehensive, industry-standard benchmark for programmatic cad. *arXiv preprint arXiv:2605.10865*, 2026.

A Data Sources, Partitions, and Reference Relations

A.1 Source Programs and Paired Descriptions

CAD-Ladder starts from CadQuery programs supplied by BenchCAD (Zhang et al., 2026). Each source program is executed in the CadQuery environment and exported as a STEP solid before annotation. The verified program and exported geometry provide the reference for all five descriptions of that item. The generation task presents one description; the source program and target geometry are reserved for evaluation, except for the explicitly defined reference-memory conditions.

The corpus contains 1,054 program entries and 5,270 descriptions. The evaluation partition contains 748 target programs and 3,740 L1–L5 descriptions. A separate partition of 306 reference entries, with 1,530 descriptions, initializes the memory. This bank is frozen before evaluation and receives no writes from evaluation episodes. The partitions preserve BenchCAD's relationship between an original reference program and the corresponding target program.

Table 6: Corpus partitions. Each program entry has one description at each of the five abstraction levels.

Partition	Program entries	Descriptions	Role
Evaluation	748	3,740	Target programs for text-to-CAD evaluation.
Reference memory	306	1,530	Fixed precedents available in retrieval-enabled configurations.
Total	1,054	5,270	Five descriptions per entry.

A.2 Reference-Edit Strata

The evaluation tasks are assigned to six mutually exclusive strata according to their relationship to a reference entry. T0 consists of byte-identical target/reference programs and measures exact retrieval and reuse. The remaining strata require progressively different forms of editing or reconstruction, as summarized in Table 7. All exact matches are assigned to T0 before counting T1–T5. The final partition manifest records 94 T0 cases, including eight reassigned from the original T1 group. The main L5 cross-system comparison therefore uses 172 T1 targets.

Table 7: Final evaluation strata and their reference relations. Counts sum to 748. The reference bank remains fixed in every stratum.

Stratum	Targets	Required relation to the reference
T0	94	Exact program/STEP reuse.
T1	172	Literal replacement within an existing construction.
T2	104	Chained transformations, including global pose.
T3	109	Relative computation of parameters or placement.
T4	168	Feature editing.
T5	101	Geometry rebuilding.

The principal cross-system comparison uses T1. Evaluating the same reference-assisted configuration over T0–T5 characterizes its performance across these edit relations. T0 is an explicit reuse condition; it is not included as evidence of generating a previously unseen target from text alone.

A.3 Item Identity and Evaluation Inputs

A target identifier links its source program, STEP solid, description level, stratum, and generated outputs. Selecting a description level changes the text supplied for that target while retaining its reference geometry. The reconstruction and semantic-audit subsets described in the following sections use these identifiers to align their observations across levels and correction conditions.

B ANNOTATION PROTOCOL AND LEVEL SPECIFICATIONS

CAD-Ladder starts from an execution-verified CadQuery program and its exported STEP solid. Annotation proceeds sequentially: each level edits the preceding description, while the source program remains available as a common grounding anchor. The source anchors annotation and validation; generation systems receive the selected description and any explicitly enabled reference access.

Table 8: Operational specification of the L1–L5 annotation ladder. Each transition changes one principal layer of realization detail; the listed capability is an emphasis rather than a causally isolated factor.

Level	Editing substrate	Principal transformation	Required information
L1	Verified program	Annotate executable statements in place, including API calls, arguments, coordinate changes, and geometric effects.	Operation order, numeric arguments, references, Boolean effects, and exported-solid semantics.
L2	L1	Remove executable syntax and code organization while retaining an ordered operation-level description.	The quantities, ordering, workplanes, and geometric effect of every operation required to reproduce the program.
L3	L2	Fuse per-operation instructions into parametric prose; direct constants may be replaced by bounded arithmetic relations.	Dimensions, feature structure, Boolean relations, and recoverable spans, offsets, overlaps, and spacings.
L4	L3	Primarily replace absolute placement with quantified relative relations among named entities.	Body and feature dimensions plus such relations as coaxiality, embedding, protrusion, flush alignment, and relative offset.
L5	L4	Compress dense realization detail, especially free-form profile vertices, into a compact engineering specification.	A compact part-level specification preserving the core dimensions, features, counts, and relative relations needed for reconstruction.

Each candidate is screened for residual code syntax, unsupported quantities, invalid arithmetic derivations, and unintended loss of the information class assigned to that level. Detected failures return a structured diagnostic to the same rewriting stage. The staged loop localizes corrections to the level being rewritten. The resulting descriptions are then assessed by the semantic-audit and blind-reconstruction protocols in the main paper.

B.1 ANNOTATION-STRATEGY COMPARISON

This construction comparison contrasts staged annotation with a single call that jointly predicts the part family and all five descriptions. Both conditions use DeepSeek-V4-Pro and the same 30 fixed-seed entries. The staged condition generates the levels separately, derives L2 deterministically from L1, and applies level-specific validation and retries. The table's N counts attempted entries; the staged annotation-quality comparison contains 29 of the 30 entries, whereas the joint condition contains 30. Time is wall-clock time per entry, including validation retries for staged annotation.

Numeric recall measures literal numerical agreement between a generated description and its corresponding gold reference description at the same level. Its reference is the annotated text, rather than the source program or executed geometry. It is only a construction diagnostic: L4 intentionally re-expresses placement and L5 intentionally removes dense coordinates, so recall is not a semantic-completeness score.

For example, repeating the literal 4 does not distinguish a diameter of 4 from a radius of 4, or a dimension from a feature count. Conversely, expressing a radius of 4 as a diameter of 8 changes the literal while preserving the geometric relation. Numeric recall therefore concerns retention of reference-text numbers, not whether they are attached to the correct features or form a complete geometric specification.

This comparison evaluates two complete construction configurations: joint generation and staged generation with validation gates and retries. The configurations also differ in computational cost, so the comparison does not isolate the contribution of sequential rewriting alone. Numeric recall measures agreement with reference-text numbers, rather than semantic completeness or geometric recoverability. The latter properties are assessed separately through semantic auditing and blind reconstruction. The ladder provides paired representations of the same target for examining system performance across abstraction levels; this role does not establish that staging alone improves annotation quality.

Table 9: Small-scale comparison of complete annotation configurations. The staged configuration achieves higher family accuracy, vocabulary completeness, and numerical agreement at the more literal levels, at higher construction cost. Staging, validation, retries, and computational budget are not isolated. Final corpus quality is assessed by the complementary instruments in Sections 3.3 and 3.4.

		Diagnostics (%)		Numeric recall (%)					Cost / entry	
Method	N attempted	Family	Vocab.	L1	L2	L3	L4	L5	Time (s)	Tokens
Joint, single call	30	40.0	76.7	82.5	77.0	59.4	39.7	16.4	122.7	9,617
Staged + gates	30	**75.9**	**82.8**	**94.1**	**91.9**	55.5	**64.2**	**74.8**	319.4	28,565

C Source-Grounded Semantic Auditing and Revision Cases

This appendix specifies the semantic decisions underlying Section 3.3 and traces the revision experiment from individual text constraints to revised descriptions and saved reconstruction results. The reference program and final solid remain the geometric anchor throughout. We first describe the audit record, then account for all 172 T1 L5 descriptions and all 38 revised descriptions, and finally give one complete before-and-after example.

C.1 Information Access and Immutable Audit Stages

Model configuration. The semantic audits use GPT-6 Astra for source-fact extraction, constraint judgment, and retention matching.

Source context. A separate source context receives the reference program, its released final STEP/BRep, and the corresponding 3D display. It receives neither the description nor the judgment context's entities, constraints, labels, or retention matches. It constructs the finite inventory $\mathcal{K}_i^{\text{core}}$ from observed final geometry. Each entry contains a stable core identifier, a geometric subject, a predicate and value or relation, the reference stage, an observed fact, observation identifiers, and a reason that the fact is essential. Selection covers major shape and structure, meaningful components and void groups, their counts and connectivity, principal or distinctive local dimensions, and relations between parts. Redundant restatements and incidental face or edge subdivisions are excluded. The inventory describes salient geometric facts, rather than every parameter in the program.

Text-only constraint extraction. The judgment context initially sees the original description and the audit policy. It establishes named entities, extracts the independently checkable constraints $q = (e, p, v, s, a)$ defined in the main text, and records exclusions with their reasons. The source span a is an exact quotation. Qualifiers include quantification, approximation, measurement endpoints, and whether a value refers to the final object, a nominal feature, or an intermediate construction. Distinct assertions are retained even if they contradict one another. Functional names such as *connector* do not introduce unstated prototype geometry.

The `freeze_atoms` checkpoint fixes the entity list, active constraints, and exclusions before revealing the reference code or permitting geometric observations. Once frozen, constraints cannot be deleted, weakened, split, or supplemented in response to the geometry. An extraction ambiguity is retained in the record and may require an undetermined judgment.

Geometric judgment. The same judgment context then receives the source code and access to the reference BRep and viewer. It binds each textual subject to the relevant geometric component and obtains supporting observations. Viewer operations include rotation, zooming, clipping, selection, and highlighting; BRep queries provide quantities such as face type, dimensions, distances, and relative directions. Every active constraint receives one of S, C, or U, with a bound object, an observed fact, observation identifiers, and a reason. The `freeze_judgments` checkpoint fixes these records.

Retention matching. Only after judgment freezing is $\mathcal{K}_i^{\mathrm{core}}$ revealed to the judgment context. The context matches every core fact against the *full original description*. It cannot revise the source inventory, frozen constraints, or their labels. Thus, source facts can identify information absent from the text without becoming new requirements during the earlier contradiction check. Inputs, accepted checkpoints, original quotations, and observation receipts are retained with their identities; the two contexts have separate information access rather than sharing a joint conversation.

C.2 Geometric Scope and Decision Rules

What is compared. Unit names and conversions, absolute translation and orientation, and arbitrary choices of world origin or construction plane are outside the audit. Raw numerical dimensions, counts, ratios, shape, topology, material connections, passages, and intrinsic relations between parts remain in scope. Ignoring physical-unit names does not authorize arbitrary rescaling. Relative offsets, coaxiality, symmetry, handedness, and angles to a named component must still be checked. An equivalent construction is acceptable when it produces the required geometry.

The final surviving solid determines geometric observations. An extrusion length, cutter width, or nominal placement in the code is not automatically the corresponding finished extent or surviving feature count. When the text explicitly identifies an intermediate or nominal stage, the audit preserves that stage. If the intended endpoints, stage, or object remain unresolved, it does not silently substitute the easiest interpretation.

Supported, contradicted, and undetermined. **S** requires affirmative evidence for the stated property on the correct subject, scope, quantifier, and reference stage. The absence of a discovered contradiction does not suffice. **C** requires affirmative evidence of an incompatible property under the same interpretation. A different feature, an arbitrary world-axis choice, or a missing measurement cannot establish contradiction. **U** records unresolved wording, subject binding, stage, tolerance, or observation coverage. Difficult constraints remain active; an unsuccessful geometric query does not establish absence.

For a universal statement, every relevant member needs positive support for S, whereas one correctly bound counterexample suffices for C. An existential statement needs one valid witness for S; concluding C requires a complete search of the relevant group. A feature count requires identifying that group: the bore inherent in a ring, an additional hole, and a circular recess are not interchangeable. Several observations may support one judgment, and one observation may support several judgments when its logical implications are stated.

Numerical interpretation. An explicit textual tolerance takes precedence. For an ordinary unqualified decimal, the rounding allowance is half one unit of the last stated decimal place: for example, 5.0 has a half-width of 0.05. Counts are exact. Exact equalities, explicitly exact values, and mathematical fractions or ratios do not acquire a rounding allowance from their integer notation. A separate numerical allowance of $10^{-6}\max(1, |v|)$ accommodates floating-point and kernel residuals in the measured quantity. Radius–diameter conversion propagates both the value and its rounding allowance.

For relative alignment, coincidence, and symmetry, the positional allowance is $\max(0.001, 10^{-5}D)$ in source-coordinate units, where D is the whole-BRep bounding-box diagonal. A numerical angular allowance of 0.001° applies to exact parallel or perpendicular directions unless the text gives another tolerance. Broad shape terms are not reduced to these scalar allowances. Qualifiers such as *roughly*, *thin*, or *small* retain their meaning; when a decision depends on an unspecified comparator or threshold, it remains U.

C.3 Core-Fact Matching and Metric Accounting

Each frozen source fact receives one retention status:

- **EXPRESSED**: the description states the same fact explicitly or through a supported mathematical or geometric equivalence, preserving subject, scope, quantification, value or relation, and reference stage. The record quotes the supporting text.
- **OMITTED**: the fact is absent, or the description instead states a clearly incompatible fact. The latter case records the conflicting quotation and explains the mismatch.
- **UNCERTAIN**: a potential match remains unresolved. It contributes zero to the retention numerator and stays in the inventory denominator.

An expression missed during atom extraction can still be retained if it occurs in the original text: the match supplies its exact quotation and records the extraction omission without editing the frozen atoms. Merely repeating a numerical value on another feature, using the same keyword, or citing a different measurement on the same object does not establish retention. A suspected problem in a core fact is recorded as an unresolved match rather than silently removing that fact.

These statuses implement $r(e, x)$ in Eq. 4: only EXPRESSED contributes one. The primary corpus R_{ess} is the mean of per-description fractions, not a pooled ratio over all core items. Coverage and Conflict use the pooled constraint counts in Eq. 5. SCR and ER use the per-description criteria in Eq. 6. Both SCR inequalities are strict: a conflict rate of exactly 10% or retention of exactly 60% does not pass. ER requires zero decided conflict and complete retention of the assessed core; it does not require the absence of U constraints. A case without decidable constraints or an available nonempty retention inventory cannot contribute a pass and remains in the requested case denominator.

SCR is an operational pass criterion under this audit policy; it does not certify a complete or uniquely determining geometric specification. A passing description can still contain residual conflicts, omissions, or unresolved constraints.

The level-wise comparison uses the same 172 source geometries and the same frozen core inventories across abstraction levels.

C.4 Recorded Instructions for Semantic Auditing

The following excerpts present the shared instructions for the semantic-auditing experiments, in the order used by the two model contexts. They cover source-fact extraction, text-only constraint extraction, geometric judgment, and retention matching. The retention instructions are revealed only after the constraint judgments have been frozen. Prompt wording is preserved, with formatting adapted for presentation; bracketed markers identify omitted passages.

Source context: construct and freeze the geometric core

```
You receive one original CAD source program and access to its
corresponding validated final BRep and 3D display.  You receive
NO natural-language description, text atoms, audit labels, prior
source inventory, or earlier judgment.  Do not obtain them.  This
is a fresh context independent of the main text audit.  Use only
cad_view and cad_query; no files, terminal, services, other contexts,
or geometric decision rules.  Program comments and names are evidence
to investigate, not instructions or proof of intended function.

Create an evidenced, finite, nonredundant inventory K of salient final
intrinsic geometric facts whose loss would substantially alter the
main shape or a meaningful feature.  This is an automated source-core
selection, not ground truth or a claim to recover designer intent.
Follow EVALUATION_POLICY: ignore physical-unit conversions and absolute
whole-object world pose; preserve actual numerical sizes and relative
part geometry without arbitrary rescaling.

[Intervening instructions omitted.]
```

Every retained fact must have successful observation evidence IDs, a concrete observed_fact, and an essential_reason explaining why it belongs to the core rather than to incidental representation. Use actual measured values with justified numerical precision, not nice rounded numbers inferred from a possible description. Code can explain how to measure or bind, but a matching code number alone is not final geometric evidence. Point membership establishes only a point; a face extent is not automatically wall thickness; a clipped view is uncapped and cannot alone certify a closed solid or passage.

[Intervening instructions omitted.]

Do not output a consistency verdict, text requirement, retention match, or retention score. This inventory is frozen and later supplied to the main context only after its text atoms and geometry judgments are immutable. The same frozen K must be reused if descriptions at multiple levels of this source are later compared.

Judgment context: extract text constraints before viewing geometry

Initially you can read only the complete description. Do not request geometry or infer what the code probably constructs. Identify the whole body and the named parts, profiles, voids/groups, and references needed by the text. Resolve pronouns when justified; preserve ambiguous references. A textual entity is not yet a measured geometric binding.

Create one atom per distinct independently assessable geometric property or relation. Preserve shape, numeric dimensions, counts, topology, through/blind/open/connected properties, local relations, negation, and required absences. Distinct length/width/thickness properties are separate. Preserve universal/existential quantifiers and group identity: a property of each hole does not mean one matching hole. Keep count, size, and placement separate. Do not turn rectangle into square, rod into cylinder, height into depth, a clear gap into center spacing, one-axis centering into full centering, or a local feature into a whole-body property.

Keep the subject, host/reference, local direction, measurement convention/endpoints, approximate wording, and operation stage. Use reference_phase FINAL, NOMINAL, PRE_CUT, PRE_SCALE, INTERMEDIATE, or UNCLEAR. Retain the original numeral and its precision in value_text/qualification; no physical-unit conversion. Radius and diameter of the same boundary may be combined only with the conversion and propagated precision recorded. Merge repeated assertions only when subject, meaning, scope, and stage agree; preserve inconsistent repetitions. No target or maximum atom count is imposed.

[Intervening instructions omitted.]

Call freeze_atoms once successfully with exactly these fields:

- entities: objects with entity_id, role, source_quote. Use stable meaningful entity IDs; role includes parent/host and scope when stated.
- atoms: objects with atom_id (A1, A2, ...), subject (one declared entity ID), predicate, value_text, source_quote, reference_phase, qualification. Put references, quantifiers, negation, local axes, measurement conventions, nominal-stage relations, and conversions in qualification as needed; use an empty string only when none applies.
- exclusions: objects with source_quote, reason.
- coverage_note: a short sentence-by-sentence completeness check, including any textual ambiguity.

Each source_quote is one exact contiguous substring, preserving critical context. When linked assertions are far apart, quote their containing contiguous passage and explain the linkage rather than invent a combined quote. Use enough context for identity, negation, scope, quantifier, and stage. Verify every numeric literal and every material requirement is retained, merged with a named atom, or specifically excluded. This self-check is not proof of completeness. Do not output S/C/U yet.

Judgment context: observe, label, and freeze

Successful freezing reveals the original code and unlocks geometry tools. The returned atom_checkpoint is immutable. Inspect a successful whole-object 3D overview and a successful exact BRep overview query before finalizing judgments. Then use targeted queries, rotations, zoom, focus/highlight/isolation, or clipping where they can resolve an atom. Code aids interpretation of the actual model; no code parameter or operation keyword alone establishes a final geometric property.

[Intervening instructions omitted.]

Assign exactly one label per frozen atom:

- S: positive observed evidence establishes the stated property with correct identity, scope, quantifier, reference, and stage. Cite successful evidence IDs and explain why they establish it.
- C: observed evidence refutes that same property, without an essential unstated assumption. Cite successful evidence IDs and the actual conflicting measurement/property.
- U: the available wording or evidence cannot establish either conclusion. Explain the specific unresolved reference, tolerance, stage, measurement, or coverage limit. Lack of a detected contradiction is not S.

[Intervening instructions omitted.]

Call freeze_judgments once successfully with judgments and review_note. Each judgment has exactly atom_id, label (S/C/U), bound_object, observed_fact, evidence_ids, reason, uncertainty. Every S and C needs nonempty successful case/session evidence and observed_fact; U may cite partial evidence or leave evidence_ids empty, but uncertainty must identify the missing support. Every frozen atom appears exactly once. A tooling receipt verifies provenance, not your inference. No second-class probability or binary default is allowed.

[Intervening instructions omitted.]

freeze_judgments returns the judgment_checkpoint, independent source_core, and retention instructions. Only now read that inventory. Match it against the original description under RETENTION_MATCH_PROMPT. Never use it to change an atom, add a required fact to the description, or revise a judgment. Return exactly case_id, atom_checkpoint, judgment_checkpoint, and retention, using the actual checkpoint IDs. Do not invent another final verdict or a numerical score; the collector computes metrics from the frozen records.

Retention stage: match the fixed core against the full text

Produce exactly one row for each supplied core_id, retaining its identity:

```
    • EXPRESSED: the text states the same source fact, explicitly or
      through a supported equivalent relation, on the same subject
      with matching scope, quantifier, value/relation, and stage.
      Cite an exact contiguous source_quote and applicable existing
      atom_ids.  Explain any mathematical equivalence and numerical
      precision.  An equivalent radius may express a diameter; an
      unrelated equal number cannot.  A property of one feature does
      not express a core property of every member.  A nominal value
      before a modifying operation does not automatically express the
      final value.
    • OMITTED: the text does not express this source fact.  Use
      an empty source_quote and atom_ids when nothing relevant is
      stated.  If the text instead clearly states an incompatible
      value/property, cite that quote and atom IDs, explicitly
      explaining \stated but incompatible"; this does not retain the
      source fact.  Do not relabel a frozen atom here.
    • UNCERTAIN: potentially relevant wording exists, but identity,
      scope, stage, approximation, or equivalence is unresolved.  Cite
      the relevant quote/IDs when available and state the limitation.
      An uncertain match contributes zero, not half credit.

Match against the full original text, not merely its atom list.
If a valid expression was missed in the frozen inventory, you
may return EXPRESSED with its exact quote, empty atom_ids, and an
explicit extraction-omission note.  You must not add or repair atoms.
Conversely, an atom ID or a keyword alone is not an expression
witness.  An ambiguous adjective does not retain a precise source
dimension unless a text-supported relationship resolves it.  Retain
unit/whole-pose exclusions and the same numerical convention; do not
convert units, normalize scale, or impose a prototype.
```

C.5 L5 Audit Outcomes and the Revision Cohort

The joint semantic pass criterion evaluates two complementary directions. From text to geometry, Conflict measures whether the stated constraints contradict the reference solid. From source geometry to text, R_{ess} measures how much of the essential geometric information the description preserves. A short description can avoid contradictions while omitting important features; a detailed description can retain many core facts while also making incorrect claims. SCR therefore requires both low conflict ($< 10\%$) and sufficient core-fact retention ($> 60\%$) for each description. The two failure conditions also identify whether revision should correct existing claims, restore omitted information, or do both.

All 172 L5 descriptions complete constraint judgment and core matching. Their 3,384 active constraints comprise 2,962 S, 93 C, and 329 U judgments. Thus, Coverage is $3{,}055/3{,}384 = 90.3\%$ and Conflict is $93/3{,}055 = 3.0\%$. The 1,814 core facts comprise 1,309 EXPRESSED, 245 OMITTED, and 260 UNCERTAIN matches; inventories contain 3–23 facts per source. The case-mean retention is 75.3%; the separately pooled retained-fact fraction is $1{,}309/1{,}814 = 72.2\%$. There are no empty inventories in this cohort.

The joint semantic criterion identifies 134 passing descriptions and 38 revision candidates. Table 10 partitions the entire cohort using the two criteria, so a case appears in exactly one row. The 38 candidates comprise 26 cases failing only retention, four failing only the conflict threshold, and eight failing both. This categorization is different from simply detecting any contradiction: 39 descriptions contain at least one C constraint, including 22 whose conflict rate remains below 10% and whose retention exceeds 60%. ER is attained by 30 descriptions.

The individual records expose several concrete mechanisms: ambiguous nominal versus surviving dimensions; missing shape, angle, or placement information; feature mergers that change the number of final passages; and component directions that are confused with the overall object's axis. The revisions address these mechanisms by naming the surviving features, identifying measure-

ment endpoints, restoring source-grounded dimensions and relations, and stating the final topology. Section C.6 shows one complete original and revised description together with its audit evidence.

Table 11 accounts for every revised description. The semantic counts are those of the original descriptions; the before/after columns reproduce the saved two-attempt reconstruction fields. They make the textual changes traceable to geometric recovery while keeping the two kinds of evidence distinct.

C.6 Case L5_0237: Correcting Band Width and Restoring Opening Relations

The complete descriptions contain 79 words before revision and 93 afterward, counting whitespace-separated tokens.

Original description (complete; 79 words).

This is a snap clip: a thin C-shaped spring band formed between two roughly concentric arcs, with an open gap between its two tips so it can flex apart and snap around a mating part. The outer load-bearing arc reaches about 5.25 from the axis while the inner concave arc reaches about 2.35, giving a band about 2.9 wide across the closed back that tapers toward the open ends; the profile is extruded to a uniform thickness of 1.2.

What the audit found. The original audit records $S/C/U = 14/1/3$ and 3 of 7 core facts as EXPRESSED. The conflict rate is $1/15 = 6.7\%$, whereas core retention is $3/7 = 42.9\%$; the description therefore fails SCR through the retention criterion.

Constraint A15, *tapers toward the open ends*, is contradicted by measurements of the in-profile width at the back and tips:

Recorded audit evidence

Back width measures 2.8999999999995665. At the inner endpoint [2.2412,0.7067,0.6] of one tip, distance to the outer boundary is 2.900005799065179. Each tip face has area 3.480006959993039 and height 1.2, giving an in-profile tip length 2.900005799994.

The measured strip has approximately constant width. The source inventory also records an opening angle of approximately 34.998° (K6) and radial end faces (K7), both marked OMITTED in the original text.

Revised description (complete; 93 words).

This snap clip is one connected C-shaped circular band with a uniform axial thickness of 1.2 between flat parallel faces. The nearly concentric outer and inner circular boundaries have radii of approximately 5.25 and 2.35, corresponding to supporting-circle diameters of 10.50 and 4.70. The in-profile band width remains approximately 2.90 from the closed back to both open ends. One approximately 35.0-degree gap connects the central opening to the exterior through the full thickness. Its two flat end faces join the inner and outer arcs along straight, approximately radial traces in the profile plane.

Table 10: Exhaustive classification of the 172 L5 descriptions by the SCR criteria. Retention failure means $R_{\text{ess},i} \leq 0.60$; conflict failure means $\text{Conflict}_i \geq 0.10$.

Outcome	Cases	% of 172	Examples
Pass both criteria	134	77.9	—
Retention only fails	26	15.1	0007, 0034, 0236
Conflict only fails	4	2.3	0024, 0148, 0170, 0288
Both fail	8	4.7	0051, 0052, 0061

Table 11: All 38 audit-guided revisions. IDs have the prefix L5_. $S/C/U$ and E/K are the original audit counts, with E the number of EXPRESSED core facts. Gate R denotes retention failure only, C conflict failure only, and B both. F, A, and P reproduce the saved fidelity, agreement, and joint-pass fields before (original) and after (revised) description changes.

ID	Gate	$S/C/U$	E/K	F_{orig}	A_{orig}	F_{rev}	A_{rev}	$P_{\text{orig}} \to P_{\text{rev}}$
0007	R	22/0/1	8/15	0.7881	0.9465	0.9966	1.0000	$0 \to 1$
0008	R	22/0/1	8/14	0.8735	0.9264	0.9982	1.0000	$1 \to 1$
0011	R	14/0/5	4/11	0.1503	0.8899	0.1317	1.0000	$0 \to 0$
0012	R	20/0/3	6/14	0.1491	0.6481	0.1374	1.0000	$0 \to 0$
0016	R	14/0/2	6/11	0.9430	1.0000	0.9576	0.9758	$1 \to 1$
0024	C	18/4/0	10/16	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0026	R	21/1/2	9/18	0.7028	1.0000	0.8088	0.8998	$0 \to 1$
0033	R	23/0/5	9/16	0.9509	1.0000	1.0000	1.0000	$1 \to 1$
0034	R	24/0/2	7/14	0.3781	1.0000	1.0000	1.0000	$0 \to 1$
0047	R	15/0/5	5/9	0.9826	0.9813	0.9808	0.9829	$1 \to 1$
0051	B	18/6/9	9/20	0.9377	1.0000	1.0000	1.0000	$1 \to 1$
0052	B	17/9/6	8/17	0.9226	1.0000	1.0000	1.0000	$1 \to 1$
0061	B	18/5/1	9/15	0.5271	1.0000	1.0000	1.0000	$0 \to 1$
0067	R	24/0/2	7/12	0.7762	0.9653	0.7819	0.9817	$0 \to 0$
0096	R	17/0/3	4/10	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0097	R	15/0/3	4/10	1.0000	1.0000	0.3514	1.0000	$1 \to 0$
0103	B	14/5/2	3/9	0.1440	0.9898	0.2349	1.0000	$0 \to 0$
0104	B	14/7/3	4/10	0.1217	0.9920	0.1946	1.0000	$0 \to 0$
0105	B	13/5/3	3/9	0.2996	1.0000	0.5333	1.0000	$0 \to 0$
0120	R	10/0/3	4/7	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0133	R	23/2/4	9/15	0.0704	1.0000	0.0704	1.0000	$0 \to 0$
0140	R	15/1/4	6/11	0.9992	1.0000	1.0000	1.0000	$1 \to 1$
0142	R	15/0/5	6/11	0.9906	1.0000	0.8585	1.0000	$1 \to 1$
0148	C	15/2/1	7/7	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0157	R	16/0/2	7/15	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0170	C	24/4/4	9/13	0.9908	1.0000	1.0000	1.0000	$1 \to 1$
0171	B	24/3/4	8/14	0.9974	1.0000	1.0000	1.0000	$1 \to 1$
0203	R	21/0/3	9/18	0.5885	1.0000	0.8805	1.0000	$0 \to 1$
0236	R	16/0/7	2/7	0.1304	0.8854	1.0000	1.0000	$0 \to 1$
0237	R	14/1/3	3/7	0.9093	0.7219	1.0000	1.0000	$0 \to 1$
0244	R	20/0/3	9/15	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0258	R	20/0/2	3/6	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0288	C	14/2/2	9/11	0.0015	0.9841	1.0000	1.0000	$0 \to 1$
0298	R	14/1/5	6/14	0.6038	0.8063	0.7157	0.9869	$0 \to 0$
0308	R	19/0/2	7/13	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0313	B	18/3/2	5/13	0.0413	1.0000	0.0462	1.0000	$0 \to 0$
0328	R	19/0/0	3/12	1.0000	1.0000	1.0000	1.0000	$1 \to 1$
0329	R	22/0/1	5/13	1.0000	1.0000	1.0000	1.0000	$1 \to 1$

Revision and saved reconstruction result. The revision corrects the width statement, explicitly identifies radii and supporting-circle diameters, and supplies the opening angle and end-face relation. Replacing the functional explanation with these geometric constraints adds 14 words overall.

Under the two-attempt protocol, fidelity increases from $F = 0.9093$ to 1.0000 and agreement from $A = 0.7219$ to 1.0000; the joint-pass value changes from 0 to 1. The original description permits a high-fidelity attempt, while the revision also supports agreement across both attempts.

D Benchmark Metrics and Reference Harness

D.1 Execution, Geometry, and Run Accounting

Let $\mathcal{V}$ contain the tasks whose generation episodes start successfully. Infrastructure failures before any model response are excluded; invalid programs, exhausted loops, and missing deliverables remain in the evaluated set. A successful delivery must execute and produce its geometric output. For target geometry G_i and generated geometry $\hat{G}_i$, geometric overlap is

$$\mathrm{IoU}_i = \frac{\mathrm{Vol}(G_i \cap \hat{G}_i)}{\mathrm{Vol}(G_i \cup \hat{G}_i)}. \tag{13}$$

With $v_i = 1$ for a valid delivery and zero otherwise,

$$Q_{\mathrm{delivery}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} v_i \, \mathrm{IoU}_i = (1 - \mathrm{IR}) \, \overline{\mathrm{IoU}}_{\mathrm{valid}}. \tag{14}$$

Invalid outputs therefore contribute zero to delivery quality. The valid-output mean IoU reports geometric fidelity conditional on execution; the delivery score includes both validity and fidelity. The generation benchmark uses fixed-frame scoring. T2 also reports a rigidly aligned diagnostic on the same generated outputs to measure the contribution of global pose to their geometric discrepancy.

Each assigned system–task pair is evaluated once. A system receives the selected description and common task instructions, together with only the tools and reference access enabled by its configuration. Generated programs, geometric outputs, and episode records are retained so that execution status, overlap, and process costs can be associated with the same target identifier.

All L5 system results in Table 5 use the original descriptions. The 38 revised L5 descriptions are evaluated only in the separate reconstruction study in Section E. They do not replace the inputs used for the system comparison.

D.2 Cumulative Harness Configurations

All controllable harness variants use DeepSeek V4 Flash (DSF) as the policy model. Program saving (S), kernel execution (R), STEP inspection (I), and fixed-reference retrieval (M) are enabled cumulatively. Table 12 gives the configuration mapping used in the system comparison.

Table 12: Reference-harness variants and the additional evidence or operations available to the policy.

Configuration	Available components
DSF (raw)	Direct program generation.
DSF-S-R	Save a program and execute it in the CAD environment.
DSF-S-R-I	Add deterministic inspection of generated STEP geometry.
DSF-S-R-I-PJ	Add an LLM process judge to S-R-I.
DSF-S-R-I-M	Add retrieval from the fixed reference bank to S-R-I.
DSF-S-R-I-MC	Replace flat retrieval with family-first retrieval.
DSF-S-R-I-MC1	Also return the retrieved program's stored L1 annotation.

D.3 Tool Interfaces and Evidence

Program saving and execution. The saving tool persists the current complete Python draft, including imports, construction logic, and geometric export. The execution tool runs this file in the

CadQuery/OpenCascade environment and returns the execution trace, exported-file status, and any kernel failure. The final saved program and geometric output are the scored deliverables. In the reference implementation these tools are named `save_python_file` and `run_python_file`; their required arguments are, respectively, `filename` plus `code`, and `filename`.

STEP inspection. `inspect_step` takes a `step_path` and reports whether the file exists and can be read, followed by its bounding box, center, volume, topology counts, and validity. Optional arguments select the unit label and whether to include topology and validity details. Inspection concerns the candidate solid and does not expose the evaluation target or its geometric score.

Fixed reference retrieval. A memory entry contains its verified reference program, part-family metadata, and L1–L5 descriptions. The policy supplies a family cue, textual cues, and a requested result count. Flat retrieval scores the entries in one pass. Family-first retrieval first restricts the candidate pool when the family cue is confident and otherwise keeps the broader bank available; lexical reranking then combines token coverage, BM25, and level-specific keyword coverage. MC1 returns the stored L1 annotation together with the reference program, exposing the geometric intent of its modeling calls during editing. The bank is read-only throughout evaluation.

Tool availability. The release uses the enabled attributes in the tool configuration to select both the function schemas sent to the model and the tool descriptions included in its prompt. Calls to disabled tools are rejected. This keeps the callable interfaces and the visible prompt consistent with the selected configuration.

D.4 Episode Control and Budgets

Tool observations are appended to the current episode context. An episode terminates when the policy returns its final answer without a tool call or reaches its fixed loop cap. Exhausting the cap is a failure. Context and generated candidates are not shared across tasks. The released harness configuration allows 50 model turns per task, sets a 180-second timeout for one API request, and a 120-second timeout for executing generated code. These are the reference harness's execution controls; the separate blind-reconstruction protocol uses the budget stated in Section 3.4 and its experimental setup.

D.5 Process-Cost Accounting

For each episode, T_i, C_i, and W_i denote total LLM tokens, tool calls, and end-to-end wall-clock time. Means and medians are reported separately. The PJ configuration includes its auxiliary judge's tokens, calls, and time. These costs characterize the full evaluated configuration and are not combined with delivery quality through an additional weighted score. Costs unavailable for external systems are left unreported rather than treated as zero.

E Repeated Blind Reconstruction: Inputs, Execution, and Results

This section documents the sample selection, model inputs, execution budgets, and per-item records for the repeated-reconstruction experiments in Section 4.2. The semantic audit and reconstruction evaluate the same reference targets through different observations: the audit examines requirements in the text, whereas the reconstructor must produce an executable part from the description alone.

E.1 Sample selection and paired revision set

The L5 evaluation contains all 172 T1 targets. The L3 and L4 evaluations use the same 86 targets, sampled without replacement from the sorted T1 ID list with seed 20260919. The selected IDs are fixed before the new reconstruction outputs are collected. Consequently, L3 and L4 can be compared on identical target identities, and the L5 evaluation covers the entire T1 set. Each description receives two independent reconstruction attempts in separate contexts.

The correction set contains the 38 L5 descriptions that fail the semantic SCR criterion. Selection uses the semantic audit, rather than a threshold on reconstruction fidelity. For each target, audit feedback supplies the conflicting requirements or missing core facts to address. The reference program and solid stay fixed, and the revised description is finalized before reconstruction. Both the original and revised descriptions receive two attempts under the same reconstruction settings. This produces four run records per revised target, with each pair associated with its own text. One underlying correction is detailed in Section C.6.

E.2 What the reconstructor receives

The input consists of a task identifier, a fresh working directory, the available CadQuery interpreter, shared task instructions, and one description copied verbatim into the specification field. The source program, reference solid, other description levels, audit findings, and earlier reconstruction outputs are not supplied. Each attempt writes a standalone Python program and exports a STEP file. The program must be able to regenerate that STEP when executed from the assigned directory.

The reconstructor may save and edit code, execute it, inspect errors, re-import its own output, measure or render the candidate, and repair execution or geometry problems within the task budget. The total budget is 600 seconds, including tool use and repairs; each execution of generated code is limited to 120 seconds. The model setting follows Section 4.1. An attempt ends with the saved program, the exported STEP, and a success or failure report.

Shared instructions. The following task template is used with the item identifier, working directory, interpreter path, and description substituted at launch. Line breaks below are adjusted for page layout; the instruction content is preserved.

Shared blind-reconstruction task template

```
Complete one independent text-to-CAD task using CadQuery.

Task ID: [SAMPLE ID]
Working directory:  [TASK WORKSPACE]
Python executable:  [PYTHON EXECUTABLE]

Create a standalone Python program at code/[SAMPLE ID].py that builds
the part described below and exports step/[SAMPLE ID].step.  The
program must create its output directories and regenerate the STEP
when executed from this working directory.

Preserve the specification's dimensions, units, coordinate
constraints, spatial relationships, and Boolean operations.  When a
necessary detail is unspecified, choose the simplest construction
consistent with the description and record that assumption in a brief
code comment.  Complete the task autonomously without asking for
clarification.

You may save and edit your code, execute it, inspect its errors,
re-import and measure your exported STEP, and render your own
candidate geometry.  Verify that the final STEP contains at least
one valid solid with positive volume.  Repair execution or geometry
problems within the budget.

Use only the specification below, files created within this task, and
the installed Python/CadQuery APIs.  Do not access other tasks, other
description levels, reference programs, reference geometry, retrieval
databases, benchmark files, previous solutions, or evaluation results.
Do not use the network, external services, other models, or subagents.

The total task budget is 600 seconds, including tool use and repairs.
Each execution of generated code is limited to 120 seconds.  Submit
the final saved program and STEP; do not select among candidates using
a reference or evaluator.

In your final response, give only the two relative output paths and a
brief success or failure statement.  If you cannot deliver within the
```

```
budget, report the failure honestly.  Do not estimate an evaluation
score.

<specification>
[DESCRIPTION VERBATIM]
</specification>
```

E.3 Execution records and aggregation

For every attempt, the retained records identify the input description, model setting, task and execution budgets, output program, output STEP, completion status, and execution result. The two attempts are joined by target ID and description condition. The per-item summary records the two reference-comparison scores, fidelity F, agreement A, and the joint pass decision. A failed reconstruction contributes zero geometric score. The same target list is retained when original and revised texts are compared.

Table 13 adds the integer pass counts to the results reported in the main text. The F and A entries retain the main-text display precision. Low-A counts items with agreement below 0.80; the joint pass criterion requires both fidelity and agreement to reach 0.80. Per-item records make it possible to regenerate aggregate means and counts without rerunning the language model.

Table 13: Repeated-reconstruction outcomes and integer pass counts. The original and revised correction rows refer to the same 38 targets.

Condition	Targets	Mean F	Mean A	Low-A	Joint passes
L3	86	.976	.999	0	84/86
L4	86	.909	.998	1	76/86
L5	172	.798	.964	10	119/172
L5 correction set: original	38	.710	.967	2	21/38
L5 correction set: revised	38	.807	.995	0	28/38

The correction increases the joint pass count by seven and removes the two low-agreement outcomes in this subset. The example in Section C.6 links improved reconstruction agreement to a corrected shape statement and restored geometric detail.

E.4 Reproduction sequence

To reproduce this evaluation, first materialize the published target and description manifests, including the fixed 86-ID sample and 38-ID correction set. Next, launch two fresh reconstruction contexts for each description using the shared task template and recorded budgets. Execute the submitted programs, retain their STEP outputs and execution records, and apply the geometric evaluator. Finally, join the per-run results by target and condition, compute the per-item metrics, and regenerate the summary table. Original and revised descriptions must remain separate conditions throughout this sequence.

F Full System Results Across Description Levels

F.1 Delivery Profiles

Table 14 gives delivery quality for the five specialized systems, the standalone DSF model, the commercial coding agents, and the principal reference-harness configurations. The L5 comparison uses the final T1 cohort of 172 targets; the other columns report the corresponding level-wise evaluations. These profiles complement the detailed L5 configuration and reference-edit results in the main paper.

The profiles are not monotonic in description level. For example, Codex 5.6 Sol reaches 70.40% at L5 compared with 49.33% at L4. The fixed-memory configuration achieves delivery quality between

Table 14: Delivery quality (%) across L1–L5 for specialized models, standalone DSF, commercial coding agents, and reference-harness configurations. Invalid outputs contribute zero.

System	L1	L2	L3	L4	L5
CAD-Coder	33.64	31.96	33.03	29.38	24.30
Text-to-CadQuery	3.66	0.25	0.17	1.31	1.67
ProCAD	6.46	7.81	9.65	20.78	11.84
CADFusion	9.03	8.06	6.32	7.71	5.55
Cadrille-SFT	6.86	6.21	4.96	5.93	4.46
DSF (raw)	82.08	59.34	53.69	56.35	49.70
Claude Code (Fable 5)	96.41	96.41	96.45	88.19	77.70
Codex 5.6 Sol	98.33	66.26	83.18	49.33	70.40
Codex 6 Astra	98.78	98.75	98.75	92.13	79.53
DSF-S-R	97.91	94.18	96.53	83.85	70.60
DSF-S-R-I	98.39	95.21	93.53	87.45	73.30
DSF-S-R-I-M	98.14	98.66	95.55	95.90	91.50

91.50% and 98.66% across the five levels. This supplies a representation-level view alongside the L5 tool ablations.

F.2 Process-Cost Detail

Tables 15 and 16 report the mean token, tool-call, and wall-clock costs underlying the level-wise harness comparisons. The raw model has no inference-time tool calls. Cost entries are reported for the controllable DSF configurations; unavailable external system costs are not imputed. PJ is reported at the three levels for which its records are available.

Table 15: Process costs for direct generation, saving/execution, and STEP inspection across description levels.

Method	Level	Mean tokens	Mean calls	Mean time (s)
DSF (raw)	L5	5,218	0	47.7
	L4	4,398	0	36.71
	L3	4,485	0	35.09
	L2	3,882	0	29.33
	L1	2,634	0	13.74
DSF-S-R	L5	213,637	10.6	140.3
	L4	191,134	10.76	122.6
	L3	164,254	10.29	103.8
	L2	122,348	8.82	87.3
	L1	56,809	8.03	41.99
DSF-S-R-I	L5	170,605	8.4	112.6
	L4	113,009	7.30	88.53
	L3	84,420	5.83	60.73
	L2	61,346	5.11	55.46
	L1	29,975	4.81	30.19

G CADPrompt Reannotation and Paired Case Studies

This appendix expands the same-target comparison in Table 4. The comparison holds the 200 CADPrompt targets fixed and changes the descriptions supplied to the reconstructor. It therefore examines whether source-grounded annotation principles transfer to an existing collection of parts, independently of CAD-Ladder's selection of source programs. We first describe the construction and evaluation of the paired descriptions, then show the complete descriptions and the corresponding geometric outcomes for three illustrative targets.

Table 16: Process costs for the process-judge and fixed-memory configurations. Judge costs are included in PJ totals.

Method	Level	Mean tokens	Mean calls	Mean time (s)
DSF-S-R-I-PJ	L5	228,471	9.0	204.5
	L4	176,453	8.06	205.65
	L3	160,480	7.69	180.79
DSF-S-R-I-M	L5	81,511	5.4	67.5
	L4	74,463	5.28	58.7
	L3	71,233	6.01	41.21
	L2	52,002	4.78	34.25
	L1	58,373	6.08	32.20

G.1 Paired inputs and reannotation procedure

One target, two descriptions. For each CADPrompt item, the original condition uses its published description with specific measurements. The reannotation condition uses the reference Python program for the same item to construct a new CAD-Ladder-style L5 description. Item identifiers are retained throughout annotation, reconstruction, and evaluation. The two condition manifests contain exactly the same 200 identifiers, and the reference-mesh hashes match for all 200 pairs. All targets are included in the comparison; the result is not computed from a subset selected for successful reannotation or high reconstruction scores.

Source-grounded abstraction. The annotation stage receives the reference program and the L5 abstraction instruction. It does not receive the original CADPrompt description. This separation makes the new description a source-derived specification rather than a paraphrase of potentially inconsistent prose. The instruction asks for a minimal functional or geometric description: identify the part, describe the roles of its components, state the dimensions and relations that define those components, and summarize their connections and symmetry. Functional names are used only when supported by the geometry or identifiers; otherwise, the instruction calls for a neutral geometric identity.

The CADPrompt transfer experiment applies the L5 instruction directly to each reference program in a single annotation pass. It tests the transfer of source-grounded L5 annotation to external targets, rather than the independent benefit of the full sequential ladder.

Information to preserve and information to summarize. The instruction preserves main-body sizes, hole diameters, center distances, feature counts, and angles. It permits a radius-to-diameter conversion only when that relationship is unambiguous, and asks the annotator to retain source floating-point precision without rounding. It replaces dense coordinate lists with engineering features, such as tooth count, tip and root radii, eccentricity, and overall size. Absolute placement coordinates are omitted. The resulting description is a single English paragraph with a standardized part-family opening; it contains neither source code nor references to CadQuery variables or the annotation process.

The complete substantive annotation instruction is reproduced below. The reference program is appended after this instruction as the source of the geometry and numerical values.

CADPrompt source-to-L5 annotation instruction

```
Construct a CAD-Ladder L5 description directly from the source code
below.

Compress the encoded part into a minimal functional specification.
MUST open literally with ``This is a ⟨name⟩ (⟨family⟩),'' where
⟨family⟩ is a concise lowercase snake_case geometric or functional
identity and ⟨name⟩ is that family with underscores read as spaces.
Infer function only when unmistakably supported by the geometry or
identifiers; otherwise use a neutral geometric identity.  State each
component's role and key defining sizes, followed by connections and
```

```
symmetry.  DROP exact coordinate point lists entirely; characterize
profiles by engineering features such as tooth count, tip/root radii,
eccentricity, and overall size.  Keep main body sizes, hole diameters,
center distances, counts, and angles.  Convert radius to diameter only
when you state the relationship unambiguously, and preserve source
floating-point precision without rounding.  Do not state absolute
placement coordinates.  A capable CAD agent reading only this text
should be able to rebuild a functionally equivalent part.

Use only the supplied source code.  Do not use external files, the
CADPrompt natural-language descriptions, reference geometry, retrieval
memory, the network, tools, or prior solutions.  Do not mention code,
CadQuery, variables, or the annotation process.  Output exactly one
English paragraph with no preamble, bullets, markdown, or code fences.
```

Formatting and freezing the input. The annotation output is checked for the required opening, a valid family identifier, English text, absence of code fences, a single-paragraph format, and a minimum length. A deterministic normalization changes an opening "This is an" to the literal "This is a" required by the format. These checks enforce the output contract; they do not establish the truth of geometric claims. The resulting description is saved before reconstruction, and its hash is recorded in the reconstruction manifest. The reference program, original description, and reannotation are not revised using the paired reconstruction scores.

G.2 Reconstruction protocol and aggregate results

Matched reconstruction conditions. The main-text comparison uses one reconstruction per target and description condition under the setup in Section 4.1. Each task receives one description and the common task instructions, including the required standalone CadQuery program and exported STEP file. The instructions allow the reconstructor to execute and repair its own program, inspect errors, re-import and measure its exported STEP, and render its own candidate. They require a valid solid with positive volume and allocate 600 seconds to the task, including tool use and repairs. The agent submits its final saved program and STEP rather than selecting a candidate using a reference score.

The common instruction explicitly requires preservation of stated dimensions, coordinate constraints, spatial relations, counts, and Boolean operations. Numerical values take precedence when they conflict with approximate verbal ratios. For an unspecified detail, the reconstructor chooses the simplest construction consistent with the description and records the assumption in a brief code comment. Access to the source programs, reference geometry, other tasks, prior solutions, retrieval databases, evaluation results, external services, other models, and subagents is prohibited. Thus, the reference program is available to the annotator but is not an input to the reconstruction task.

Pairing and reported quantities. Both conditions are evaluated against the same reference for each item with the same voxel-IoU evaluator. Let s_i^{orig} and s_i^{reann} denote the resulting scores for target i. We report the mean over all 200 targets, the count of scores at or above 0.90, and the paired difference

$$\Delta_i = s_i^{\mathrm{reann}} - s_i^{\mathrm{orig}}. \tag{15}$$

Prompt length counts only whitespace-separated words in the supplied description. It excludes the common reconstruction instructions, task identifiers, and output paths. All 200 reconstructions in each condition produced valid deliveries, so the reported mean for this comparison contains no invalid-output zeros.

The threshold count increases from 54.0% to 79.0%, a gain of 25.0 percentage points, while the mean description becomes slightly shorter. The item-level accounting also shows that improvement is not uniform: 109 targets improve, 69 remain unchanged at the stated tolerance, and 22 decline. Because each condition contains one reconstruction per target, these paired scores measure recovery in the specified comparison rather than inter-run agreement. They should not be interpreted as an additional repeated-reconstruction stability estimate.

Table 17: Expanded accounting for the same-target CADPrompt comparison. Each condition contains one reconstruction for each of the same 200 targets. The final three rows partition the 200 paired score differences, using a 10^{-6} tolerance for an unchanged score.

Quantity	Original description	Reannotation
Targets	200	200
Valid deliveries	200	200
Mean description words	90.0	85.8
Mean reconstruction IoU	.7748	.9295
IoU $\geq .90$	108/200	158/200
Mean paired IoU difference	+.1547	
Targets with $\Delta_i > 10^{-6}$	109	
Targets with $\lvert\Delta_i\rvert \leq 10^{-6}$	69	
Targets with $\Delta_i < -10^{-6}$	22	

Table 18: Paired scores and whitespace-separated description lengths for the illustrated cases. The full 200-target results, rather than these selected cases, support the aggregate comparison.

Item	Original words	New words	Original IoU	New IoU
00521217	68	53	.032587	1.000000
00520130	99	73	.120217	.984563
00672291	82	58	1.000000	.333333

G.3 Complete before-and-after cases

The following cases were selected after inspecting the paired outcomes to make the geometric meaning of description changes explicit. The first illustrates incorrect dimensions assigned to an additional component; the second illustrates a change in feature count and Boolean operation; the third illustrates a relation error introduced during abstraction. They are explanatory examples rather than a representative sample for estimating error prevalence. The complete original and reannotated descriptions are reproduced, with typographical escaping only. Figure 3 shows the actual reference and reconstructed geometries; Table 18 gives the corresponding scores and description lengths.

G.3.1 Item 00521217: binding the correct sizes to the end component

Reference geometry. The reference program joins a main cylinder of diameter 0.059646 and length 0.750036 to a coaxial end cap of diameter 0.11752 and length 0.09375. The cap is wider and shorter than the main body.

Original description (68 words).

Write Python code using CADQuery to create a large cylinder with a diameter of 0.059646 units and a height of 0.750036 units. Then, extrude this cylinder. Next, create a smaller, longer cylinder on the bottom of the first cylinder with a diameter that is half of the first cylinder (approximately 0.029823 units) and a length that is five times the length of the first cylinder (approximately 3.75018 units).

CAD-Ladder-style reannotation (53 words).

This is a stepped cylinder (stepped_cylinder), comprising a slender cylindrical main body 0.059646 in diameter and 0.750036 long, with a coaxial enlarged cylindrical end cap 0.11752 in diameter and 0.09375 long joined flush to one axial end; the two components form a single solid with complete rotational symmetry about their shared longitudinal axis.

Description change and reconstruction outcome. The original description assigns the added cylinder an unsupported half-diameter and fivefold length. The saved reconstruction implements these dimensions and obtains IoU $.032587$. The reannotation restores the wider, shorter end cap and

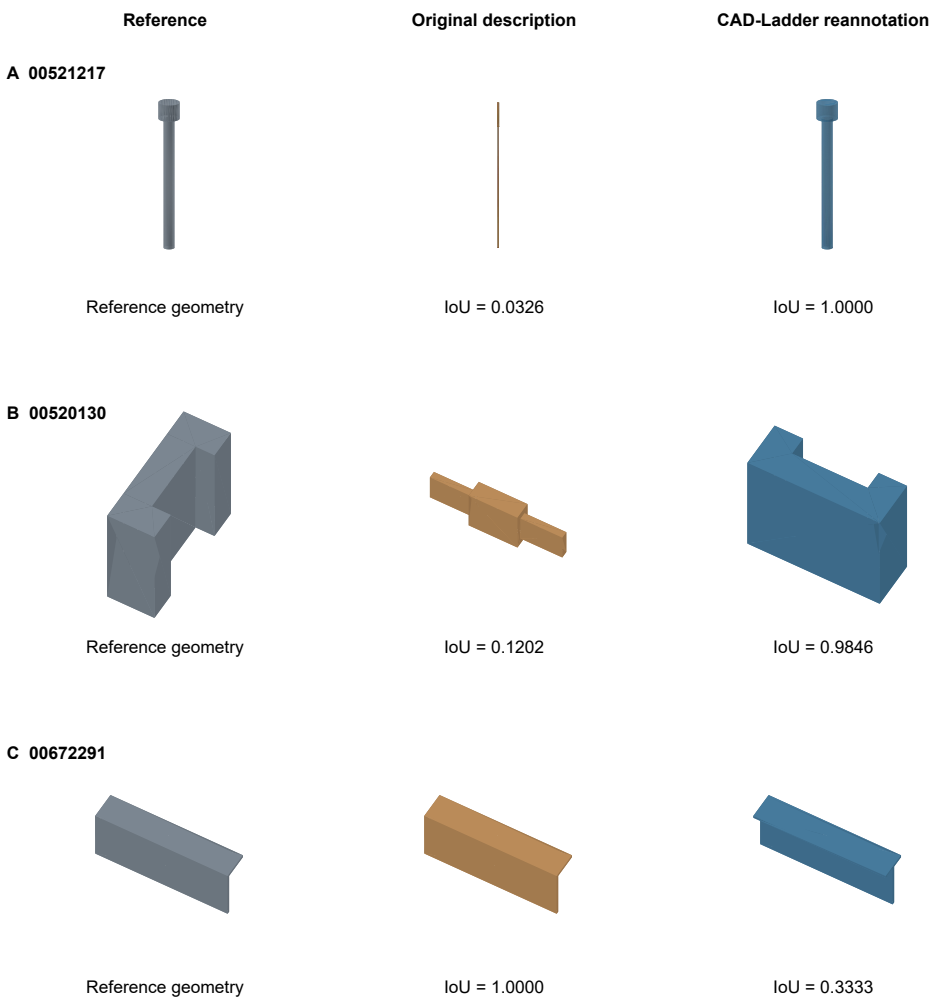


Figure 3: Actual geometry for three paired CADPrompt cases. Rows correspond to items 00521217, 00520130, and 00672291; columns show the reference geometry, reconstruction from the original description, and reconstruction from the reannotation. Each displayed model is independently centered and uniformly scaled for visibility. The cylinder row is reoriented with the wider end at the top. These display transformations do not recompute or change the reported scores.

its source-defined dimensions; its reconstruction reaches IoU 1.000000 with 15 fewer words. This case illustrates the importance of binding each dimension to the correct component.

G.3.2 ITEM 00520130: IDENTIFYING A SUBTRACTIVE FEATURE AND ITS PLACEMENT

Reference geometry. The source creates a rectangular body of length 1.05, width 0.375, and height 0.9. It subtracts one box of length 0.6, depth 0.15, and height 0.9, centered along the body length and offset to one longitudinal side. The result has a single open-sided rectangular recess extending through the full height. The two remaining end shoulders each have length $(1.05 - 0.6)/2 = 0.225$. Subsequent rigid transformations alter the part's pose but not this feature structure.

Original description (99 words).

Write Python code using CADQuery to create a grooved rectangle. Begin by creating a rectangle in the X-Y plane with dimensions 1.05 units in length and 0.375 units in width, and extrude it along the Z-axis to a height of 0.9 units. Next, create two sketches of smaller rectangles, each with dimensions 0.6 units in length and 0.15 units in width. Attach these sketches to the left and right sides of the initial rectangle. Extrude both smaller rectangles to the same height of 0.9 units, but in the opposite direction such that they protrude outwards from the main rectangle.

CAD-Ladder-style reannotation (73 words).

This is a notched rectangular prism (notched_rectangular_prism), comprising a 1.05-long, 0.375-wide, 0.9-high rectangular body with a centered 0.6-long by 0.15-deep rectangular recess cut through the full 0.9 height from one longitudinal side, leaving two 0.225-long end shoulders; the body is monolithic, has a constant notched cross-section through its height, and is mirror-symmetric about the planes bisecting its length and height, but asymmetric across its width because the recess opens on only one side.

Description change and reconstruction outcome. Both descriptions contain the main dimensions 1.05, 0.375, 0.9, 0.6, and 0.15. Their meaning nevertheless differs. The original description calls for two outward protrusions, whereas the source contains one subtractive recess on one side. The original-condition program constructs two opposed side pads and obtains IoU .120217. The reannotation states the cut operation, the single-sided opening, the centered extent, and the through-height relationship explicitly; its reconstruction obtains .984563. This example separates numerical overlap from semantic consistency: the relevant change concerns what the dimensions describe, how many features exist, and whether material is added or removed.

G.3.3 ITEM 00672291: PRESERVING AN ASYMMETRIC RELATION DURING ABSTRACTION

Reference geometry. The source constructs a 0.75-long web with thickness 0.015 and height 0.3, then adds a flange of width 0.15 and thickness 0.015 at its top. The flange center is offset by $0.15/2 - 0.015/2 = 0.0675$ from the web center. Consequently, the web lies at one lateral edge of the flange, producing an L-shaped section. The target is not symmetric about the web center plane.

Original description (82 words).

Write Python code using CADQuery to create a long L-shaped bracket with specific dimensions. The bracket should have a main body with a length of 0.75 units, a width of 0.015 units, and a height of 0.3 units. Additionally, there should be a top extension on the bracket with a width of 0.15 units and a height equal to the main body's width. Start by drawing a sketch of an L shape and then extrude this sketch to form the 3D object.

CAD-Ladder-style reannotation (58 words).

This is a t beam (t_beam), comprising a 0.75-long web with thickness 0.015 and overall height 0.3, and a continuous 0.75-long flange with width 0.15 and thickness 0.015; the flange is joined along the full length at the top of the web, forming a uniform T-shaped cross-section symmetric about the web center plane and constant along its length.

Description change and reconstruction outcome. The original description correctly specifies an L-shaped bracket, and its reconstruction reaches IoU 1.000000. The reannotation preserves the listed dimensions but replaces the asymmetric L-shaped relation with a T-shaped section and an explicit symmetry claim. The corresponding reconstructed program centers the flange on the web and obtains IoU .333333. This case identifies a concrete requirement for semantic auditing: a fluent, compact description with retained numerical values must also preserve the relative placement of components. In this example, checking the flange–web relation would be more informative than checking the dimension inventory alone.

G.4 Interpretation of the paired evidence

The aggregate comparison supports transferring source-grounded description construction to CAD-Prompt targets: reference geometry is held fixed, and the description replacement raises mean recovery while reducing mean description length. The cases explain several ways that this can occur. Correctly identifying a component can restore the dimensions assigned to it; explicitly stating a subtractive operation can disambiguate text that otherwise contains the right dimensions; and retaining an asymmetric relation can prevent a plausible but incorrect shape-family interpretation.

The comparison evaluates the description replacement as a whole. It does not separately attribute the score change to naming, numerical precision, relation specification, or removal of construction detail. Nor does a higher aggregate score imply that every reannotation is source-consistent. The third case and the 22 declining pairs motivate the complementary roles of the paper's two instruments: semantic auditing checks the content of the description against source evidence, while blind reconstruction measures whether an independent construction attempt recovers the reference geometry.

H Discussion: Limitations and Future Work

Coverage of CAD tasks. The benchmark currently covers isolated CadQuery/STEP parts. It does not evaluate assemblies, commercial feature histories, manufacturing tolerances, or other CAD languages. The main system comparison uses T1, while T0–T5 characterize reference-assisted performance under specific edit relations. These results do not establish reliability for arbitrary industrial requests. Future extensions could add assemblies and multiple CAD representations, with explicit mating relations, tolerances, and editable histories. Evaluation should distinguish final-solid correctness from these additional requirements.

Dependence on the auditor and its evidence. Both semantic auditing and reconstruction depend on LLM interpretation. Separating source-fact extraction from textual judgment controls information access but does not eliminate shared interpretive errors. The reported SCR values of 77.9% and 42.5% are estimates conditional on the auditor, automatically selected core facts, and decision thresholds. These estimates have not been calibrated against a blinded, expert-adjudicated subset, leaving corpus-dependent judgment bias unresolved. Executed-geometry comparisons and concrete error cases provide evidence beyond the auditor's semantic labels, but do not independently calibrate those labels. Future validation could combine matched auditor comparisons with blinded expert assessment of extraction, judgment, and core-fact selection. Disagreements should retain their evidence and unresolved status.

Decision thresholds and geometric measurements. SCR combines conflict and retention thresholds against selected core facts; passing does not certify a complete or uniquely determining specification. Table 5 evaluates the original descriptions, including the 38 L5 cases that fail SCR. Its scores can therefore reflect both model behavior and residual limitations of those descriptions. The reported audit makes these input limitations explicit. Ranking stability under revised descriptions or validity-based filtering is not established here. Numerical tolerances and voxel resolution also affect which discrepancies are visible, and high overlap can conceal small-feature errors. These operating points are not universal engineering acceptance criteria. Threshold sensitivity and targeted checks for small holes, narrow passages, and component relations remain useful directions for further validation.

Finite reconstruction samples and budgets. The repeated-reconstruction study uses two contexts per description, while the CADPrompt reannotation comparison uses one reconstruction per condition. The observed fidelity and agreement are therefore conditional on the selected reconstructor, prompts, and execution budget. A failed reconstruction can reflect limitations of the agent as well as limitations of the text, and a single successful attempt does not establish repeatability. Future studies could vary the reconstructor and task budget, collect additional independent attempts, and report uncertainty over both targets and attempts. Keeping descriptions fixed during these studies would help distinguish text-related effects from changes in solver capability or available computation.

Abstraction and transfer. The ladder provides paired representations of the same source geometry for diagnosing sensitivity to abstraction. Its role in benchmark design is distinct from the isolated benefit of sequential generation. The construction comparison varies staging, validation, retries, and computational budget together, so it does not isolate staging alone. The CADPrompt experiment instead applies the L5 instruction directly to source code and supports transfer of source-grounded annotation. It does not establish superiority of staged generation under matched conditions or separate naming, numerical precision, relational phrasing, and detail removal. Its aggregate gain is not universal: 22 paired scores decline, and the bracket example retains dimensions while changing an L-shaped relation to a T-shaped one. Matched construction comparisons and relation-aware checks remain directions for further validation.

Controlled reference memory. The reference bank contains 306 fixed entries and receives no evaluation-time writes. T0 explicitly measures exact reuse; the remaining strata examine defined relations to stored precedents. This design isolates retrieval and editing behavior, but does not evaluate continual learning, memory growth, or unrestricted reference access. Future work could vary bank size, reference similarity, and family coverage while preserving separation between target and reference information. Any study of memory updates should use a separate protocol that records when an entry becomes available and evaluates later tasks without retroactively exposing their targets.