Constraint-Only Learning

A no-answers learning method built on constraints, invariants, and constructive stuckness

Working paper

Author: Brandon Kennedy (with ChatGPT)
Date: 10 February 2026
Version: 1.0

Core rule

Never transmit answers. Transmit constraints.

If the learner asks for the answer, give another constraint.

Contents

1. Executive summary

Constraint-Only Learning (COL) is a learning method where the instructor never shows answers, solutions, or final forms. Instead, the instructor supplies constraints, invariants, and contradiction feedback that force the learner to construct the solution internally.

COL treats being stuck as an essential compression phase. The method is not neglect: the instructor actively regulates pressure and direction without collapsing the task into an answer.

Core rule:

Never transmit answers. Transmit constraints. If the learner asks for the answer, respond with another constraint.

COL is suitable for mathematics, physics, creative arts, and intuition training, especially when the goal is to produce original thinkers rather than operators who can mimic procedures.

2. Definition and scope

Definition. Constraint-Only Learning is an instructional method in which learners are never shown answers, solutions, or canonical final forms. Learning occurs exclusively through internal construction under externally supplied constraints, invariants, and contradiction feedback.

Scope. COL is designed for deep understanding, transfer, and originality. It is not optimized for short-term test performance or rapid coverage of large curricula.

What COL is not. COL is not gatekeeping, silence, or a refusal to help. The instructor remains highly active, but only in ways that preserve the learner’s construction process.

- Not: "Figure it out alone."
- Is: "I will shape the search space so you can build it yourself."
- Not: memory testing.
- Is: structural reconstruction.

3. Axioms and non-negotiables

These axioms make the method explicit and defensible.

Axiom 1: Non-Transmission. Understanding cannot be transferred; it must be reconstructed.

Axiom 2: Constructive Pressure. Cognitive restructuring requires unresolved tension (stuckness).

Axiom 3: Constraint Sufficiency. Given sufficient constraints, valid structure can emerge without answers.

Axiom 4: Internal Ownership. A solution constructed internally is categorically different from one received.

Non-negotiables.

- No revealing the final answer, the final derivation, or the canonical trick.
- No confirming correctness with "yes" or "that’s right". Correctness is implied only through survival under constraints.
- Feedback must be structural: contradictions, invariants, boundary failures, scaling failures, or unjustified assumptions.
- If the learner requests the answer, respond with a new constraint or a test they can run.

4. Instructor role and allowed moves

In COL, the instructor is not a source of truth. The instructor is a constraint manager, contradiction detector, and pressure regulator.

Allowed moves (examples).

- Constraint checks: "That violates constraint X."
- Edge cases: "Does that still hold if n = 0 / n is huge / the signal is noisy / the lighting changes?"
- Invariants: "What stays the same if you rotate / scale / translate / change coordinate systems?"
- Assumption audit: "Which step depends on an assumption you have not justified?"
- Dimensional sanity: "Do the units match?" (physics)
- Perceptual sanity: "Where does the eye go first, and is that intentional?" (art)

Forbidden moves.

- Stating the final result or giving a worked example that contains the result.
- Providing the next step in a way that collapses the search ("just do X then Y").
- Confirming correctness with direct praise of the solution itself. (You may praise process: persistence, careful testing, clean reasoning.)

5. The COL session protocol

This protocol works for 1-on-1 tutoring, self-study, or classroom facilitation.

5.1 Phase 1: Problem seeding

- State the goal without giving the form of the answer.
- State the boundaries and what counts as a valid solution.
- Provide any necessary definitions and the constraint list.

5.2 Phase 2: Exploration under constraints

- Learner proposes candidate structures.
- Instructor responds only with constraint feedback, invariant prompts, and assumption audits.
- Learner iterates, tests, and refactors.

5.3 Phase 3: Stuckness (the sacred phase)

When the learner says "I’m stuck", do not unblock forward. Redirect sideways while preserving pressure.

- Back up one layer: "What must be true before this question makes sense?"
- Strip complexity: "What is the smallest version that still hurts?"
- Invert: "What would it look like if the opposite were true?"
- Perturb: "What changes if you remove one variable or relax one constraint?"
- Assumption hunt: "What are you treating as fixed that might not be?"

5.4 Phase 4: Emergence

The learner reaches a self-generated inevitability: "If that’s true, then this has to be true." Instructor response is minimal: "Check it against the constraints."

5.5 Phase 5: Stabilization

- Stress-test against new cases and adversarial examples.
- Generalize: identify the invariant that made the solution possible.
- Translate across representations (algebraic, geometric, verbal, computational, perceptual).

- Only after stabilization, optionally compare to standard formulations (without presenting as 'the answer').

6. Implementation patterns

6.1 Classroom pattern

- Publish constraints and invariants on the board as the single source of truth.
- Students work in groups; each group must present an attempted model and a failure mode.
- Whole-class discussion is about which constraints eliminate which families of solutions.
- No one is allowed to present a final answer; only surviving structures and remaining gaps.

6.2 1-on-1 coaching pattern

- Start each session with a two-minute recap of constraints (learner says them).
- Use a 'move limit': instructor can only speak in questions or constraint statements.
- End with a reflection: learner writes the three invariants they discovered and one assumption they corrected.

6.3 Self-study pattern

- Write the constraints at the top of the page before you start.
- Whenever you feel stuck, you must generate three alternative representations (diagram, analogy, algebra).
- Maintain a 'failure log' describing how each attempt violates a constraint.

7. Domain mappings

7.1 Mathematics

Provide definitions, axioms, invariants, and counterexamples. Withhold worked solutions and canonical tricks.

- Instead of giving a proof, demand properties any valid proof must satisfy (e.g., no continuity assumptions; works for all n; survives a pathological case).
- Use contradiction feedback: "Your step assumes what you’re trying to prove."
- Use edge-case forcing: small n, large n, degenerate geometry, non-generic inputs.

7.2 Physics

Provide conservation laws, symmetries, dimensional constraints, and limiting cases. Withhold formulas and standard derivations.

- Require correct units and limiting behavior (e.g., low-velocity or weak-field limit).
- Require invariance under a change of frame when appropriate.
- Use sanity checks: energy conservation, monotonicity, stability.

7.3 Art and perception training

Provide perceptual constraints and intention checks. Withhold aesthetic 'answers'.

- Constraint examples: eye-path control, value grouping, unintended tangents, depth cues, balance vs tension.
- Feedback is descriptive, not prescriptive: "This pulls attention here" rather than "move it there".
- Learner iterates until the image satisfies the intended constraint set.

7.4 Intuition and abstract reasoning

Provide paradoxes, ill-posed questions, and incomplete systems. Preserve ambiguity until the learner identifies what must be specified.

- Teach 'question hygiene': the learner must state what would count as an answer.
- The learner earns progress by surfacing hidden assumptions and unstated constraints.
- Allow incubation: stuckness is allowed to persist across sessions.

8. Assessment without answers

COL assessment focuses on structure, not recall of a canonical solution.

- Constraint fluency: can the learner state and apply the constraints without prompting?
- Failure-quality: can the learner produce informative counterexamples and explain why attempts fail?
- Transfer: can the learner apply the same invariants to a new problem with different surface features?
- Compression: can the learner summarize the core invariant in one paragraph, then one sentence?

Practical rubrics often use three levels: violates constraints, partially satisfies, survives all known constraints. No category is "correct"; the highest category is "currently undefeated".

9. Troubleshooting: stuckness that never resolves

When stuckness does not produce emergence, it is usually one of the following.

9.1 Missing prerequisite structure

- Symptom: the learner cannot even generate plausible candidates.
- Fix: back up one layer and run COL on the prerequisite.

9.2 Hidden assumption

- Symptom: the learner cycles between similar failed attempts.
- Fix: force an assumption inventory: "List everything you are treating as fixed."

9.3 Ill-posed problem

- Symptom: constraints conflict or do not specify a unique family of solutions.
- Fix: celebrate it. The learning is the discovery that the question needs additional constraints.

> Pressure regulation. If frustration becomes unproductive, reduce complexity (smaller instance), increase visibility (draw it), or loosen one constraint temporarily to explore the landscape—then re-tighten.

Appendix A: Constraint prompt bank

Use these as reusable instructor moves. They preserve the learner’s construction while giving direction.

A.1 Invariant prompts

- What stays the same when you change representation?
- What cannot change without breaking the goal?
- If you scale the input by 10, what should scale and what should not?
- What symmetry do you suspect is present?

A.2 Boundary and edge-case prompts

- What happens at the extremes (0, infinity, degenerate cases)?
- Can you design an input that breaks your current model?
- Does your idea survive noise, rounding, or small perturbations?

A.3 Assumption audits

- Which step is a guess rather than a deduction?
- What are you assuming about smoothness/linearity/independence?
- If that assumption is false, what fails first?

A.4 Reframing prompts

- What is the smallest version of this problem?
- What is the opposite claim and what would it imply?
- Can you restate the problem without the key word you are relying on?
- What question would make this answer inevitable?

Appendix B: One-page cheat sheet

The rule. Never transmit answers. Transmit constraints.

Instructor identity. Constraint manager • Contradiction detector • Pressure regulator

Allowed moves. Invariants • Edge cases • Assumption audits • Dimensional/perceptual sanity • Contradiction feedback

Forbidden moves. Worked solutions • Canonical tricks • Direct confirmation of correctness

When they ask for the answer. Provide a new constraint or a test they can run.

When they are stuck. Redirect sideways: back up a layer, strip complexity, invert, perturb, expose assumptions.

End.