Le Bézier du calcul

A Constraint-First Framework for Computation as Geometry

Version 4.0 — Extended Theory, Discrete Mathematics, and Formal Logic

Jared Lewis
parcRI Research / Ada Computing Company
Houston, Texas
jlew@parcri.net

February 2026

Abstract

We present **Version 4.0** of the constraint-first execution model in which *verification is the computation.* Let X denote a system state space (discrete, continuous, or mixed) and let $\Omega \subseteq X$ be the lawful region induced by a family of predicates (laws). Execution is a trajectory $\gamma : [0, 1] \to X$ that remains in Ω for all t. Candidate trajectories are restricted to Bézier curves, yielding previewable, locally controllable, and bounded proposals. A computation commits (`fin`) only when the entire curve is admissible; otherwise it rejects (`finfr`) and returns a witness identifying the first violation.

This extended edition deepens the mathematical foundations from high-school algebra through graduate-level analysis, introduces *discrete mathematics* (posets, lattices, Boolean algebras), *formal logic* (propositional calculus, predicate logic, sequent calculus, and typing judgments), and provides a comprehensive **notation guide**. We develop six domain use cases, unify them in a single **master worked example** (safe autonomous-agent navigation), and present worked exercises with full solutions throughout—structured as an academic monograph rather than a problem set.

Contents

1 Motivation: Programs as Paths, Safety as Admissibility

Modern systems fail because they can *reach* states that should never exist. Bugs, vulnerabilities, inconsistent interfaces, and unreliable agents share a common root: the execution model permits illegal intermediate states and attempts recovery afterward.

We invert the paradigm.

Computation should be defined such that invalid states cannot be expressed.

In the constraint-first model, **verification is the computation**. Rather than "compute then check," we "propose a path, verify it entirely, then commit." The key insight is that restricting candidate paths to Bézier curves converts this from an open-ended verification problem into a bounded geometric one: the convex-hull property, subdivision stability, and parametric smoothness give us deterministic, previewable, auditable execution.

What is new in V4.0. This version:

(i) Develops the full mathematical stack from high-school algebra to graduate analysis.
(ii) Introduces discrete mathematics (lattices, partial orders, Boolean algebras) as a native formalism for law composition.
(iii) Adds formal logic (propositional, predicate, and sequent calculus) with typing judgments for proposals.
(iv) Provides a complete notation and symbol guide.
(v) Presents six use-case domains unified in a single master worked example.
(vi) Includes worked exercises with solutions integrated into the exposition.

2 Notation and Symbol Guide

We collect every symbol used in this paper. The reader is encouraged to return to this section as a reference.

Symbol	Meaning
X	State space (set of all possible system states).
x, y, z	Elements of X (individual states).
Ω	Lawful region $\subseteq X$; the set of all admissible states.
$L_i : X \to \{\top, \bot\}$	A *law*: a predicate that classifies states as lawful ($\top$) or unlawful ($\bot$).
$\mathcal{L} = \{L_1, \ldots, L_m\}$	The law family; a finite collection of laws.
$\top, \bot$	Logical true and false (also written `true`, `false`).
$\wedge, \vee, \neg, \Rightarrow$	Logical and, or, not, implies.
$\forall, \exists$	Universal ("for all") and existential ("there exists") quantifiers.
$\gamma : [0,1] \to X$	Execution trajectory (a proposed path through state space).
P_0, P_1, P_2, P_3	Control points of a cubic Bézier curve.
$B_n(t)$	General Bézier curve of degree n.
$b_{i,n}(t)$	Bernstein basis polynomial $\binom{n}{i} t^i (1-t)^{n-i}$.
$\mathrm{conv}(S)$	Convex hull of a point set S.
`fin`	Commit: the entire trajectory is admissible.
`finfr`	Reject: the trajectory violates at least one law.

$t^\star$	First-violation time: $\inf\{t \in [0,1] \mid \gamma(t) \notin \Omega\}$.
$W = (i^\star, t^\star, \gamma(t^\star), \Delta)$	First-violation witness tuple.
Δ	Suggested repair direction in control-point space.
$\mathcal{E}$	Ledger (append-only sequence of entries).
$(P, \leq)$	Partially ordered set (poset).
$\sqcap, \sqcup$	Meet (greatest lower bound) and join (least upper bound) in a lattice.
$\mathbf{0}, \mathbf{1}$	Bottom and top elements of a bounded lattice.
$\overline{a}$	Complement of element a in a complemented lattice.
$\vdash$	Entailment / derivability (sequent calculus turnstile).
Γ	A context: a set of assumptions or premises.
$\Gamma \vdash \varphi$	"From assumptions Γ, we can derive φ."
$\cdot$	Denotational semantics brackets (interpretation function).
$\mathbb{Z}, \mathbb{R}, \mathbb{R}^n$	Integers, reals, n-dimensional real space.
$\| \cdot \|$	Norm (Euclidean unless stated otherwise).
∇	Gradient operator.
∂	Partial derivative.

3 Mathematical Preliminaries I: Algebra through Calculus

We build the mathematical machinery from first principles, assuming only high-school algebra and progressing to the calculus of parametric curves.

3.1 Sets, Tuples, and Functions (HS Level)

A *set* is an unordered collection of distinct objects. We write $A = \{1, 2, 3\}$. Set-builder notation: $A = \{x \in \mathbb{Z} \mid 0 < x < 4\}$.

An *ordered pair* is (a, b) where order matters: $(1, 2) \neq (2, 1)$. An *n-tuple* is $(x_1, x_2, \ldots, x_n) \in X_1 \times X_2 \times \cdots \times X_n$.

A *function* $f : A \to B$ assigns to each $a \in A$ a unique $b \in B$. A *predicate* is a function $P : A \to \{\top, \bot\}$ (returns true or false).

Example: State as a Tuple

A traffic-light controller has state $x = (color, timer, pedestrian\_request) \in \{\text{R,Y,G}\} \times \mathbb{Z}_{\geq 0} \times \{\top, \bot\}$. The state space is $X = \{\text{R,Y,G}\} \times \mathbb{Z}_{\geq 0} \times \{\top, \bot\}$.

3.2 Predicates and the Lawful Region (HS/Intro College)

Definition 3.1 (Law)**.** A *law* is a predicate $L_i : X \to \{\top, \bot\}$. We say state x *satisfies* L_i when $L_i(x) = \top$.

Definition 3.2 (Lawful Region)**.** Given a law family $\mathcal{L} = \{L_1, \ldots, L_m\}$, the *lawful region* is

$$\Omega \;=\; \{x \in X \mid \forall\, i \in \{1, \ldots, m\},\; L_i(x) = \top\} \;=\; \bigcap_{i=1}^{m} L_i^{-1}(\top).$$

The lawful region is the *intersection* of the preimages of true. This is the key geometric fact: adding laws can only shrink Ω, never enlarge it.

Example: Pythagorean Constraint Space

Let $X = \mathbb{Z}^3$ with laws:

$$L_1(a,b,c) : a^2 + b^2 = c^2, \tag{1}$$
$$L_2(a,b,c) : a > 0 \ \wedge \ b > 0 \ \wedge \ c > 0, \tag{2}$$
$$L_3(a,b,c) : \gcd(a,b,c) = 1 \quad \text{(primitive triple)}. \tag{3}$$

Then $\Omega = \{(3,4,5),\ (5,12,13),\ (8,15,17),\ \dots\}$ is the set of all primitive Pythagorean triples.

3.3 Parametric Curves and the Bézier Family (Precalculus/Calculus)

A *parametric curve* in $\mathbb{R}^n$ is a function $\gamma : [0,1] \to \mathbb{R}^n$. The parameter t traces the curve from $\gamma(0)$ (start) to $\gamma(1)$ (end).

Definition 3.3 (Bernstein Basis Polynomials)**.** For integers $0 \le i \le n$,

$$b_{i,n}(t) \ = \ \binom{n}{i} t^i \, (1-t)^{n-i}, \qquad t \in [0,1].$$

Key properties:

(a) **Non-negativity**: $b_{i,n}(t) \ge 0$ for $t \in [0,1]$.
(b) **Partition of unity**: $\sum_{i=0}^{n} b_{i,n}(t) = 1$ for all t.
(c) **Symmetry**: $b_{i,n}(t) = b_{n-i,n}(1-t)$.

Definition 3.4 (Bézier Curve)**.** Given control points $P_0, P_1, \dots, P_n \in \mathbb{R}^d$, the Bézier curve of degree n is

$$B_n(t) \ = \ \sum_{i=0}^{n} b_{i,n}(t)\, P_i \ = \ \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \, P_i.$$

For the **cubic** case ($n = 3$), which is our primary execution primitive:

$$\gamma(t) = (1-t)^3 P_0 + 3(1-t)^2 t \, P_1 + 3(1-t) t^2 \, P_2 + t^3 P_3. \tag{4}$$

Worked: Expanding the Cubic Bézier

Setting $n = 3$ in the Bernstein basis:

$$b_{0,3}(t) = (1-t)^3, \qquad b_{1,3}(t) = 3t(1-t)^2,$$
$$b_{2,3}(t) = 3t^2(1-t), \qquad b_{3,3}(t) = t^3.$$

Verification of partition of unity:

$$(1-t)^3 + 3t(1-t)^2 + 3t^2(1-t) + t^3 = \big((1-t)+t\big)^3 = 1^3 = 1. \ \checkmark$$

This confirms that $\gamma(t)$ is always an *affine combination* of the control points, which is the algebraic root of the convex-hull property.

3.4 Derivatives of Bézier Curves (Calculus I–II)

The derivative of a degree-n Bézier curve is a degree-$(n-1)$ Bézier curve whose control points are the *forward differences* of the original:

Theorem 3.1 (Bézier Derivative)**.**

$$B_n'(t) = n \sum_{i=0}^{n-1} b_{i,n-1}(t)\,(P_{i+1} - P_i).$$

Proof. Differentiate the Bernstein form using the product rule and the identity $b_{i,n}'(t) = n\big(b_{i-1,n-1}(t) - b_{i,n-1}(t)\big)$, then re-index. The result follows by telescoping. □

For the cubic case:

$$\gamma'(t) = 3\big[(1-t)^2(P_1 - P_0) + 2(1-t)t(P_2 - P_1) + t^2(P_3 - P_2)\big]. \tag{5}$$

This means the tangent vector at $t = 0$ is $\gamma'(0) = 3(P_1 - P_0)$ and at $t = 1$ is $\gamma'(1) = 3(P_3 - P_2)$. The control handles literally *are* the velocity of the trajectory at its endpoints—a fact that makes "steering" a computation geometrically intuitive.

4 Core Framework: Trajectories, Proposals, Commitment

We now formalize the execution model.

Definition 4.1 (State Space)**.** Let X denote the set of all possible system states. X may be discrete (database records), continuous (physical configurations), or a product of both.

Definition 4.2 (Execution Trajectory)**.** An *execution trajectory* is a continuous map $\gamma : [0,1] \to X$ satisfying

$$\forall\, t \in [0,1], \quad \gamma(t) \in \Omega.$$

We say γ is *admissible* when this holds.

Definition 4.3 (Proposal)**.** A *proposal* is a candidate cubic Bézier trajectory γ specified by control points (P_0, P_1, P_2, P_3), where P_0 is the current state and P_3 is the intended target. The proposer is called **Ada**.

Definition 4.4 (`fin` and `finfr`)**.**

- If $\forall\, t \in [0,1]$, $\gamma(t) \in \Omega$, the proposal *commits*: the system transitions to P_3. This is `fin`.
- If $\exists\, t$ such that $\gamma(t) \notin \Omega$, the proposal is *rejected* without state mutation. This is `finfr`.

4.1 The Convex Hull Theorem

Theorem 4.1 (Convex Hull Containment)**.** *For a Bézier curve with control points $\{P_0, \ldots, P_n\}$,*

$$\gamma(t) \in \mathrm{conv}(\{P_0, \ldots, P_n\}) \quad \textit{for all } t \in [0,1].$$

Proof. Since $b_{i,n}(t) \geq 0$ and $\sum_i b_{i,n}(t) = 1$, each point $\gamma(t) = \sum_i b_{i,n}(t)\, P_i$ is a convex combination of $\{P_i\}$. By definition, every convex combination lies in the convex hull. □

Corollary 4.2 (Quick Reject)**.** *If $\mathrm{conv}(\{P_0, \ldots, P_n\}) \cap \Omega = \varnothing$, then γ is inadmissible and we may immediately return `finfr` without sampling the curve.*

Remark 4.1. The converse does *not* hold: $\mathrm{conv}(\{P_i\}) \subseteq \Omega$ does not guarantee admissibility unless Ω is itself convex. When Ω is non-convex, we must verify pointwise (via subdivision).

4.2 Witnesses: Where the Path Breaks

Definition 4.5 (First-Violation Witness)**.** Let γ be an inadmissible proposal. Define

$$t^\star = \inf\{t \in [0,1] \mid \gamma(t) \notin \Omega\}.$$

A *witness* is the tuple

$$W = (i^\star,\ t^\star,\ \gamma(t^\star),\ \Delta),$$

where $i^\star$ is the index of a violated law at $t^\star$ and $\Delta \in \mathbb{R}^d$ is a suggested repair direction in control-point space.

The repair direction Δ can be computed as the negative gradient of the constraint violation at the responsible control point:

$$\Delta_k = -\eta \cdot \nabla_{P_k}\big[\max(0,\ -L_{i^\star}(\gamma(t^\star)))\big], \tag{6}$$

where $\eta > 0$ is a step size and the gradient is taken with respect to the control point P_k that has the largest Bernstein weight $b_{k,n}(t^\star)$ at the violation time.

4.3 De Casteljau Subdivision

The *de Casteljau algorithm* splits a Bézier curve at parameter $t = s$ into two sub-curves, each of which is itself a Bézier curve of the same degree. This is the backbone of the verification algorithm.

For a cubic curve with control points (P_0, P_1, P_2, P_3) split at s:

$$\begin{aligned}
Q_0 &= P_0,\\
Q_1 &= (1-s)P_0 + sP_1,\\
Q_2 &= (1-s)^2P_0 + 2s(1-s)P_1 + s^2P_2,\\
Q_3 &= \gamma(s) \quad \text{(the point on the curve)}.
\end{aligned}$$

The left sub-curve has control points (Q_0, Q_1, Q_2, Q_3) and the right sub-curve (Q_3, R_2, R_1, R_0) where:

$$\begin{aligned}
R_0 &= P_3,\\
R_1 &= (1-s)P_2 + sP_3,\\
R_2 &= (1-s)^2P_1 + 2s(1-s)P_2 + s^2P_3.
\end{aligned}$$

Theorem 4.3 (Subdivision Convergence)**.** *After k levels of recursive midpoint subdivision, the maximum distance between the control polygon and the curve is bounded by*

$$\delta_k \ \le\ \frac{1}{4^k}\,\delta_0,$$

where δ_0 is the initial deviation. In particular, $\delta_k \to 0$ exponentially.

This guarantees the verification algorithm terminates: after finitely many subdivisions, the control polygon approximates the curve to any desired tolerance.

4.4 Verification Algorithm

Listing 1: Deterministic bounded verification

```python
def verify_bezier(P0, P1, P2, P3, laws, budget):
    """Returns (fin, None) or (finfr, witness)."""
    stack = [(P0, P1, P2, P3, 0.0, 1.0, 0)]
    while stack:
        Q0, Q1, Q2, Q3, a, b, depth = stack.pop()
        if depth > budget.max_depth:
            return ("finfr", ("depth_exceeded", a))
        # Quick reject: convex hull test
        if hull_disjoint(conv([Q0,Q1,Q2,Q3]), Omega, laws):
            return ("finfr", ("hull_outside", a))
        # Quick accept: hull inside Omega
        if hull_inside(conv([Q0,Q1,Q2,Q3]), Omega, laws):
            continue
        # Subdivide
        L, R = de_casteljau_split(Q0, Q1, Q2, Q3, 0.5)
        mid = (a + b) / 2.0
        stack.append((*R, mid, b, depth + 1))
        stack.append((*L, a, mid, depth + 1))
    return ("fin", None)
```

4.5 Ledger: Reproducible Computation

Each proposal and its result are committed to an append-only ledger $\mathcal{E}$:

$$e_j = (\mathit{hash}_j,\ P_0^{(j)},\ P_1^{(j)},\ P_2^{(j)},\ P_3^{(j)},\ v_{\mathcal{L}},\ \mathit{result}_j,\ W_j).$$

Identical inputs yield identical results, making the entire computation history reproducible and auditable.

5 Mathematical Foundations II: Analysis and Topology

5.1 Continuity and Compactness (Undergraduate Analysis)

Theorem 5.1 (Image of Compact Set)**.** *If $\gamma : [0,1] \to \mathbb{R}^d$ is continuous and $[0,1]$ is compact, then the image $\gamma([0,1])$ is compact.*

Since $[0,1]$ is compact and Bézier curves are polynomial (hence continuous), $\gamma([0,1])$ is a compact subset of X. This means:

(a) The image is closed and bounded.
(b) Any continuous function on the image attains its maximum and minimum (*extreme value theorem*).
(c) Constraint checking over the image reduces to checking over a compact set.

Theorem 5.2 (Admissibility is a Closed Condition)**.** *If each law L_i is continuous (i.e., $L_i^{-1}(\top)$ is closed in X), then Ω is closed, and the set of admissible trajectories forms a closed subset of $C([0,1], X)$ in the supremum norm.*

Proof. $\Omega = \bigcap_i L_i^{-1}(\top)$ is an intersection of closed sets, hence closed. If $\gamma_n \to \gamma$ uniformly and each $\gamma_n(t) \in \Omega$ for all t, then $\gamma(t) = \lim_n \gamma_n(t) \in \Omega$ since Ω is closed. □

5.2 Lipschitz Bounds on Violation Detection (Graduate)

Definition 5.1 (Constraint Margin)**.** For a state x and law L_i with an associated continuous violation measure $\ell_i : X \to \mathbb{R}$ (where $\ell_i(x) \geq 0$ iff $L_i(x) = \top$), define the *margin*:

$$\mu(x) = \min_i \ell_i(x).$$

The state is lawful iff $\mu(x) \geq 0$.

Proposition 5.3 (Lipschitz Violation Bound)**.** *If μ is Lipschitz with constant K and $\mu(\gamma(0)) = m_0 > 0$, then γ remains lawful on $[0, t_{\min}]$ where*

$$t_{\min} \geq \frac{m_0}{K \cdot \|\gamma'\|_\infty}.$$

Proof. For $s \in [0,t]$: $|\mu(\gamma(s)) - \mu(\gamma(0))| \leq K\|\gamma(s) - \gamma(0)\| \leq K\|\gamma'\|_\infty \cdot s$. So $\mu(\gamma(s)) \geq m_0 - K\|\gamma'\|_\infty s \geq 0$ when $s \leq m_0/(K\|\gamma'\|_\infty)$. □

This gives a *certified safe interval* at the start of any trajectory, reducing the region that must be subdivided.

6 Discrete Mathematics of Constraint Systems

Laws compose. This section develops the algebraic structure governing that composition.

6.1 Partial Orders and Posets

Definition 6.1 (Partial Order)**.** A *partial order* on a set S is a binary relation $\leq$ that is:

(i) **Reflexive**: $\forall a \in S,\ a \leq a$.
(ii) **Antisymmetric**: $a \leq b \ \wedge\ b \leq a \ \Rightarrow\ a = b$.
(iii) **Transitive**: $a \leq b \ \wedge\ b \leq c \ \Rightarrow\ a \leq c$.

The pair $(S, \leq)$ is a *poset*.

The Poset of Lawful Regions

Let $R = \{\Omega_{\mathcal{S}} \mid \mathcal{S} \subseteq \mathcal{L}\}$ be the collection of all lawful regions formed by subsets of the law family. Ordered by inclusion $\subseteq$, $(R, \subseteq)$ is a poset. Adding more laws moves us *down* in this poset (smaller region); removing laws moves us *up* (larger region).

6.2 Lattices of Laws

Definition 6.2 (Lattice)**.** A poset $(S, \leq)$ is a *lattice* if every pair of elements $a, b \in S$ has:

- A *meet* (greatest lower bound): $a \sqcap b$.
- A *join* (least upper bound): $a \sqcup b$.

Theorem 6.1 (Law Lattice)**.** *The set of lawful regions R, ordered by $\subseteq$, forms a complete lattice where:*

$$\Omega_{\mathcal{S}} \sqcap \Omega_{\mathcal{T}} = \Omega_{\mathcal{S} \cup \mathcal{T}} = \Omega_{\mathcal{S}} \cap \Omega_{\mathcal{T}},$$
$$\Omega_{\mathcal{S}} \sqcup \Omega_{\mathcal{T}} = \Omega_{\mathcal{S} \cap \mathcal{T}} \supseteq \Omega_{\mathcal{S}} \cup \Omega_{\mathcal{T}}.$$

The bottom element is $\mathbf{0} = \Omega_{\mathcal{L}}$ (all laws enforced) and the top element is $\mathbf{1} = X$ (no laws enforced).

Proof. The map $\mathcal{S} \mapsto \Omega_{\mathcal{S}} = \bigcap_{L \in \mathcal{S}} L^{-1}(\top)$ reverses inclusion: $\mathcal{S} \subseteq \mathcal{T} \Rightarrow \Omega_{\mathcal{T}} \subseteq \Omega_{\mathcal{S}}$. This is a Galois connection between $(\mathcal{P}(\mathcal{L}), \subseteq)$ and $(R, \supseteq)$, which induces a complete lattice structure on the image. □

6.3 Boolean Algebra of Independent Laws

Definition 6.3 (Boolean Algebra)**.** A *Boolean algebra* is a complemented distributive lattice $(B, \sqcap, \sqcup, {}^{-}, \mathbf{0}, \mathbf{1})$ satisfying:

(i) Distributivity: $a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$.

(ii) Complementation: $a \sqcap \overline{a} = \mathbf{0}$ and $a \sqcup \overline{a} = \mathbf{1}$.

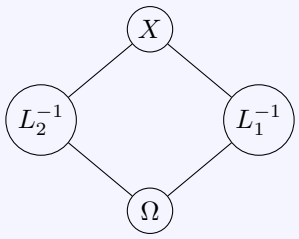
When laws are *independent* (no law is logically implied by others), the law family generates a free Boolean algebra $\mathbf{2}^m$ with 2^m elements. Each element corresponds to a *constraint configuration*: a choice of which laws to enforce and which to relax.

Worked: Two Independent Laws

Let $\mathcal{L} = \{L_1, L_2\}$. The Boolean algebra $\mathbf{2}^2$ has four elements:

Element	Laws enforced	Region
$\mathbf{1}$	$\varnothing$	X
$\overline{L_1}$	$\{L_2\}$	$L_2^{-1}(\top)$
$\overline{L_2}$	$\{L_1\}$	$L_1^{-1}(\top)$
$\mathbf{0}$	$\{L_1, L_2\}$	$L_1^{-1}(\top) \cap L_2^{-1}(\top) = \Omega$

The lattice diagram:



6.4 Monotonicity and Galois Connections

Definition 6.4 (Galois Connection)**.** A *Galois connection* between posets $(A, \leq_A)$ and $(B, \leq_B)$ is a pair of monotone maps $f : A \to B$ and $g : B \to A$ such that

$$f(a) \leq_B b \quad \Longleftrightarrow \quad a \leq_A g(b).$$

Proposition 6.2 (Laws–Regions Galois Connection)**.** *Define $f : \mathcal{P}(\mathcal{L}) \to \mathcal{P}(X)^{\mathrm{op}}$ by $f(\mathcal{S}) = \Omega_{\mathcal{S}}$ and $g : \mathcal{P}(X)^{\mathrm{op}} \to \mathcal{P}(\mathcal{L})$ by $g(R) = \{L \in \mathcal{L} \mid R \subseteq L^{-1}(\top)\}$. Then (f, g) is a Galois connection.*

This means: *the laws that a region satisfies* and *the region that a set of laws defines* are dual notions connected by an order-reversing correspondence. In practical terms, you can reason forward ("what region does this law set define?") or backward ("what laws does this region satisfy?") and get consistent answers.

7 Formal Logic for Constraint-First Computation

We now embed the framework in formal logic, giving it a deductive structure.

7.1 Propositional Logic Review

Propositional Connectives

Symbol	Name	Read as
$\neg\varphi$	Negation	"not φ"
$\varphi \wedge \psi$	Conjunction	"φ and ψ"
$\varphi \vee \psi$	Disjunction	"φ or ψ"
$\varphi \Rightarrow \psi$	Implication	"if φ then ψ"
$\varphi \Leftrightarrow \psi$	Biconditional	"φ if and only if ψ"

In our framework, each law $L_i(x)$ is a *propositional atom* at a given state x. The lawfulness predicate is the conjunction:

$$\textsc{Lawful}(x) \equiv \bigwedge_{i=1}^{m} L_i(x).$$

7.2 Predicate Logic and Quantifiers

Quantifiers

Symbol	Name	Read as
$\forall x.\, \varphi(x)$	Universal	"for all x, $\varphi(x)$ holds"
$\exists x.\, \varphi(x)$	Existential	"there exists an x such that $\varphi(x)$"

The core judgments of our framework in predicate logic:

`fin` (Commitment):

$$\forall\, t \in [0,1].\ \bigwedge_{i=1}^{m} L_i(\gamma(t)) = \top \quad \Longrightarrow \quad \texttt{fin}(\gamma).$$

`finfr` (Rejection):

$$\exists\, t^\star \in [0,1].\ \exists\, i^\star \in \{1,\dots,m\}.\ L_{i^\star}(\gamma(t^\star)) = \bot \quad \Longrightarrow \quad \texttt{finfr}(\gamma, W).$$

Excluded middle for proposals:

$$\forall\, \gamma.\ \texttt{fin}(\gamma) \mathbin{\dot\vee} \texttt{finfr}(\gamma, W), \tag{7}$$

where $\dot\vee$ denotes exclusive or (exactly one holds).

7.3 Sequent Calculus for Admissibility

We introduce inference rules in the sequent style. A sequent $\Gamma \vdash \varphi$ reads "from assumptions Γ, we derive φ."

Inference Rules for Le Bézier du calcul

Rule 1: Law Introduction (Law-Intro)

$$\frac{x \in X \qquad L_i(x) = \top}{\Gamma,\, x \vdash L_i(x)}$$

Rule 2: Lawful Conjunction (LAWFUL-∧)

$$\frac{\Gamma \vdash L_1(x) \qquad \Gamma \vdash L_2(x) \qquad \cdots \qquad \Gamma \vdash L_m(x)}{\Gamma \vdash x \in \Omega}$$

Rule 3: Trajectory Admissibility (TRAJ-ADM)

$$\frac{\Gamma \vdash \gamma : [0,1] \to X \qquad \forall\, t \in [0,1],\ \Gamma \vdash \gamma(t) \in \Omega}{\Gamma \vdash \texttt{fin}(\gamma)}$$

Rule 4: Hull Pre-Check (HULL-CHECK)

$$\frac{\Gamma \vdash \mathrm{conv}(\{P_k\}) \subseteq \Omega}{\Gamma \vdash \forall\, t.\ \gamma(t) \in \Omega} \quad \text{(when } \Omega \text{ is convex)}$$

Rule 5: Witness Construction (WITNESS)

$$\frac{\Gamma \vdash \gamma(t^\star) \notin \Omega \qquad t^\star = \inf\{t \mid \gamma(t) \notin \Omega\}}{\Gamma \vdash \texttt{finfr}(\gamma,\ (i^\star, t^\star, \gamma(t^\star), \Delta))}$$

Rule 6: Subdivision (SUBDIV)

$$\frac{\Gamma \vdash \texttt{fin}(\gamma|_{[0,s]}) \qquad \Gamma \vdash \texttt{fin}(\gamma|_{[s,1]})}{\Gamma \vdash \texttt{fin}(\gamma)}$$

7.4 Typing Judgments for Proposals

We can view the framework through the lens of a type system. Define types:

$$\begin{aligned}
&\texttt{State} : X,\\
&\texttt{LawfulState} : \{x : X \mid x \in \Omega\} \quad \text{(refinement type)},\\
&\texttt{Proposal} : \texttt{LawfulState} \times X^2 \times \texttt{State} \to \texttt{Trajectory},\\
&\texttt{Trajectory} : [0,1] \to X,\\
&\texttt{Verdict} : \texttt{fin} \mid \texttt{finfr}(\texttt{Witness}).
\end{aligned}$$

The *typing rule* for a well-formed proposal:

$$\frac{P_0 : \texttt{LawfulState} \qquad P_1, P_2 : X \qquad P_3 : \texttt{State}}{(P_0, P_1, P_2, P_3) : \texttt{Proposal}}$$

The verification function has the type:

$$\texttt{verify} : \texttt{Proposal} \to \texttt{Verdict}.$$

Theorem 7.1 (Soundness)**.** *If* $\texttt{verify}(\gamma) = \texttt{fin}$*, then* $\forall\, t \in [0,1],\ \gamma(t) \in \Omega$*. The type system is* sound*: well-typed programs do not go wrong.*

Proof. By the structure of the verification algorithm (Section 4.4): the algorithm only returns `fin` when every sub-segment has been certified (either by hull containment in Ω or by subdivision to within the certified tolerance). By Theorem 4.3, the residual approximation error after k subdivisions is bounded by $\delta_0/4^k$, which can be made smaller than any positive margin $\mu_{\min}$ on Ω. □

8 Combinatorics of Constraint Configurations

8.1 Counting Law Subsets

With m laws, there are 2^m possible *constraint configurations* (subsets of laws to enforce). The number of non-empty configurations is $2^m - 1$.

Worked: Combinatorial Explosion

A web application with $m = 20$ independent safety laws has $2^{20} = 1{,}048{,}576$ possible configurations. Testing all of them is feasible. With $m = 50$, there are $2^{50} \approx 1.13 \times 10^{15}$ configurations—infeasible to enumerate but tractable via the lattice structure (Section 6.2), since we need only check the *atoms* (individual laws) and propagate via monotonicity.

8.2 The Inclusion-Exclusion Principle for Violations

Theorem 8.1 (Inclusion-Exclusion for Unlawful Volume)**.** *Let* $V_i = \{x \in X \mid L_i(x) = \bot\}$ *be the* violation set *of law* i*. The total unlawful volume is:*

$$|X \setminus \Omega| = \left|\bigcup_{i=1}^{m} V_i\right| = \sum_i |V_i| - \sum_{i<j} |V_i \cap V_j| + \cdots + (-1)^{m+1} |V_1 \cap \cdots \cap V_m|.$$

This is relevant when estimating the *probability* that a random proposal lands in Ω (useful for adaptive proposal strategies).

8.3 Graph Structure of Law Dependencies

Define the *law dependency graph* $G = (\mathcal{L}, E)$ where $(L_i, L_j) \in E$ iff the laws share variables (i.e., they constrain overlapping dimensions of X).

Proposition 8.2. *If* G *is disconnected with components* $C_1, \ldots, C_k$*, then*

$$\Omega = \Omega_{C_1} \times \Omega_{C_2} \times \cdots \times \Omega_{C_k},$$

and verification can proceed independently *on each component.*

This decomposition theorem is the discrete-mathematical basis for *parallelizing* the verification algorithm.

9 Use Cases

We present six domains where the Bézier-trajectory model applies, then unify them in Section 10.

9.1 Use Case 1: Integer Geometry and Number Theory

State space: $X = \mathbb{Z}^n$.
Laws: Integrality, positivity, divisibility, Diophantine constraints.
Trajectory: A Bézier path through $\mathbb{R}^n$; admissibility requires rounding to $\mathbb{Z}^n$ at sample points.

Pythagorean Triple Search

Start at $P_0 = (3, 4, 5)$. Target $P_3 = (5, 12, 13)$. The Bézier curve interpolates through $\mathbb{R}^3$; at each sampled t_k, we round to the nearest integer triple and check $L_1 : a^2 + b^2 = c^2$, $L_2 : a, b, c > 0$, $L_3 : \gcd = 1$. The trajectory is a *verified search*: if `fin`, we have found a

lawful path connecting two primitive triples. If `finfr`, the witness tells us where the path left the Pythagorean surface and suggests a control-point adjustment.

9.2 Use Case 2: Interactive Systems (UI)

State space: $X = \mathit{UIState}$ (layout, visibility, permissions, focus).
Laws: Accessibility constraints, permission checks, layout invariants, z-order consistency.
Trajectory: A smooth transition between UI states (e.g., opening a modal, navigating a form wizard).

The key insight is that UI animations are *already* parametric curves in state space. By making the animation path a Bézier trajectory and the design-system invariants into laws, we guarantee that no frame of the animation violates accessibility, permission, or layout rules. Glitches, flash-of-unstyled-content, and unauthorized-state exposure become *structurally impossible*.

9.3 Use Case 3: Database Transactions

State space: $X = \mathit{DBState}$ (the set of all possible database states).
Laws: Foreign-key integrity, uniqueness constraints, check constraints, business rules.
Trajectory: A proposed transaction as a Bézier interpolation from current state to target state.

Traditional databases use ACID transactions with rollback. In our model, the transaction is the Bézier curve, the constraints are the laws, and `finfr` replaces rollback—except that no illegal intermediate state is ever written. The witness pinpoints *which* constraint would be violated and *when* during the transition, enabling precise error messages.

9.4 Use Case 4: Agent / LLM Safety

State space: $X = \mathit{ActionSpace} \times \mathit{WorldState}$.
Laws: Safety constraints (no harmful actions), capability boundaries, authorization scopes.
Trajectory: An agent's proposed action sequence.

An LLM agent proposes a plan (sequence of actions). Rather than executing actions one-by-one and checking post-hoc, the plan is encoded as a Bézier trajectory through action-world space. The governor verifies the *entire plan* before any action is taken. If the plan would at any point violate a safety constraint, `finfr` returns a witness identifying the problematic action and suggesting an alternative.

9.5 Use Case 5: Compiler Optimization Passes

State space: $X = \mathit{IR}$ (intermediate representations of a program).
Laws: Semantic preservation (before/after equivalence), type safety, memory safety.
Trajectory: A sequence of optimization passes as a Bézier path through IR space.

Each optimization pass transforms the IR. The trajectory connects the unoptimized IR to the optimized IR; laws ensure that every intermediate IR is semantically equivalent to the original. A `finfr` witness identifies the first pass that breaks semantic preservation.

9.6 Use Case 6: Cryptographic Protocol Verification

State space: $X = \mathit{ProtocolState}$ (message histories, key states, nonce counters).
Laws: No key reuse, nonce freshness, message ordering, secrecy invariants.
Trajectory: A protocol execution trace.

Protocol verification is traditionally done with model checkers. In our framework, a protocol run is a trajectory, and the security properties are laws. The Bézier structure ensures the analysis

is bounded (finite subdivision depth), and witnesses provide counterexample traces when a property is violated.

10 Master Worked Example: Safe Autonomous Agent Navigation

We now develop a single comprehensive example that exercises every component of the framework: algebra, calculus, discrete math, formal logic, and the full proposal–verification–commit cycle.

10.1 Problem Setup

An autonomous agent (a robot, a drone, or an LLM-controlled cursor) must navigate from position A to position B in a 2D workspace $\mathbb{R}^2$. The workspace contains:

- A **rectangular obstacle** $O_1 = [2, 4] \times [1, 3]$.
- A **circular restricted zone** $O_2 = \{(x, y) \mid (x - 7)^2 + (y - 4)^2 \leq 1\}$.
- A **boundary** $\mathcal{B} = [0, 10] \times [0, 6]$ (the agent must stay inside).

Start: $P_0 = A = (1, 1)$. Target: $P_3 = B = (9, 5)$.

10.2 Defining the Laws

Law 1 (Boundary): $L_1(x, y) : 0 \leq x \leq 10 \ \wedge \ 0 \leq y \leq 6$.

Law 2 (Rectangular obstacle avoidance): $L_2(x, y) : \neg(2 \leq x \leq 4 \ \wedge \ 1 \leq y \leq 3)$.

Law 3 (Circular zone avoidance): $L_3(x, y) : (x - 7)^2 + (y - 4)^2 > 1$.

The lawful region:

$$\Omega = \mathcal{B} \setminus (O_1 \cup O_2) = \{(x, y) \in [0, 10] \times [0, 6] \mid L_2(x, y) = \top \ \wedge \ L_3(x, y) = \top\}.$$

10.3 Formal Logic Encoding

In predicate logic:

$$\begin{aligned} \text{Lawful}(x, y) \ \equiv & (0 \leq x \leq 10) \ \wedge \ (0 \leq y \leq 6) \\ & \wedge \ \neg(2 \leq x \leq 4 \ \wedge \ 1 \leq y \leq 3) \\ & \wedge \ (x - 7)^2 + (y - 4)^2 > 1. \end{aligned}$$

The `fin` judgment for a trajectory γ:

$$\frac{\gamma : [0, 1] \to \mathbb{R}^2 \qquad \forall t \in [0, 1].\ \text{Lawful}(\gamma(t))}{\vdash \texttt{fin}(\gamma)} \ (\textsc{Traj-Adm})$$

10.4 First Attempt: Naïve Proposal

Ada proposes a straight-line trajectory (degenerate Bézier with collinear control points):

$$P_0 = (1, 1), \quad P_1 = (3.67, 2.33), \quad P_2 = (6.33, 3.67), \quad P_3 = (9, 5).$$

Convex hull check: $\text{conv}(\{P_0, P_1, P_2, P_3\})$ is the line segment from $(1, 1)$ to $(9, 5)$. Does this intersect $O_1 = [2, 4] \times [1, 3]$?

The line passes through $(3, 2)$ at $t = 0.25$, which satisfies $2 \leq 3 \leq 4$ and $1 \leq 2 \leq 3$. So $L_2(3, 2) = \bot$.

Verification: The algorithm subdivides and finds $t^\star \approx 0.125$ where $\gamma(t^\star) \approx (2, 1.5)$, which enters O_1.

Witness: $W = (2,\ 0.125,\ (2, 1.5),\ \Delta)$ where Δ suggests moving P_1 upward (away from the obstacle).

Result: `finfr`.

10.5 Second Attempt: Adjusted Proposal

Following the witness, Ada adjusts P_1 upward:

$$P_0 = (1, 1), \quad P_1 = (2, 4.5), \quad P_2 = (6, 5.5), \quad P_3 = (9, 5).$$

Now the curve arcs above the rectangular obstacle. Let us verify each law.

Convex hull: $\text{conv}(\{(1, 1), (2, 4.5), (6, 5.5), (9, 5)\})$. This quadrilateral lies within $[1, 9] \times [1, 5.5] \subset \mathcal{B}$, so L_1 is satisfied for all points in the hull.

Law L_2 (rectangle avoidance): We need $\gamma(t) \notin [2, 4] \times [1, 3]$ for all t. Compute the y-component of $\gamma(t)$:

$$\begin{aligned} y(t) &= (1-t)^3(1) + 3(1-t)^2 t(4.5) + 3(1-t)t^2(5.5) + t^3(5) \\ &= 1 - 3t + 3t^2 - t^3 + 13.5t - 27t^2 + 13.5t^3 + 16.5t^2 - 16.5t^3 + 5t^3 \\ &= 1 + 10.5t - 7.5t^2 + t^3. \end{aligned}$$

At $t = 0.2$: $y(0.2) = 1 + 2.1 - 0.3 + 0.008 = 2.808$. And $x(0.2)$:

$$x(t) = (1-t)^3(1) + 3(1-t)^2 t(2) + 3(1-t)t^2(6) + t^3(9).$$

$x(0.2) = 0.512 + 3(0.64)(0.2)(2) + 3(0.8)(0.04)(6) + 0.008(9) = 0.512 + 0.768 + 0.576 + 0.072 = 1.928$.

So at $t = 0.2$, the point is approximately $(1.93, 2.81)$, which is *outside* $[2, 4] \times [1, 3]$ since $x < 2$. Continuing:

At $t = 0.3$: $x(0.3) \approx 2.68$, $y(0.3) \approx 3.67$. Since $y > 3$, the point is above the rectangle.

By monotonicity of $y(t)$ on the relevant interval and the fact that $y(t) > 3$ whenever $x(t) \in [2, 4]$, we conclude L_2 is satisfied everywhere.

Law L_3 (circle avoidance): The curve's closest approach to $(7, 4)$ occurs near $t \approx 0.75$.

$$x(0.75) \approx 6.72, \quad y(0.75) \approx 5.09.$$

Distance: $\sqrt{(6.72-7)^2 + (5.09-4)^2} = \sqrt{0.0784 + 1.1881} \approx 1.126 > 1$. So L_3 is satisfied.

After full subdivision verification: **all three laws hold for all** $t \in [0, 1]$.

Result: `fin`. The agent commits to the trajectory and navigates safely from A to B.

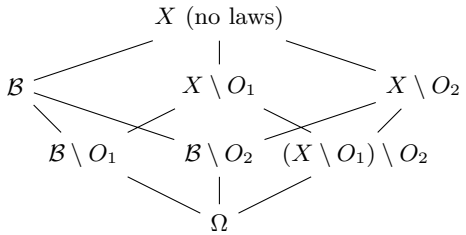
10.6 Ledger Entry

$$e = (\texttt{0xa3f...},\ (1,1),\ (2,4.5),\ (6,5.5),\ (9,5),\ v_{\mathcal{L}}{=}1,\ \texttt{fin},\ \texttt{null}).$$

10.7 Discrete Math Perspective

The three laws are *independent* (they constrain different geometric regions). The law dependency graph has no edges: $G = (\{L_1, L_2, L_3\}, \varnothing)$. By Proposition 8.3, verification can proceed in parallel on each law.

The Boolean algebra is $\mathbf{2}^3$ with 8 elements. The lattice of lawful regions:



10.8 Type-System View

$$
\begin{aligned}
&P_0 = (1,1) : \texttt{LawfulState} \quad (\checkmark\text{: satisfies all three laws}),\\
&P_1 = (2, 4.5) : X,\\
&P_2 = (6, 5.5) : X,\\
&P_3 = (9,5) : \texttt{State},\\
&(P_0, P_1, P_2, P_3) : \texttt{Proposal},\\
&\texttt{verify}(P_0, P_1, P_2, P_3) = \texttt{fin} : \texttt{Verdict}.
\end{aligned}
$$

11 Exercises with Solutions

The following worked exercises are presented in the style of an academic monograph: each exercise is stated, then solved in full, demonstrating the relevant techniques.

Exercise 11.1 (Bernstein Basis Identity — HS Algebra)**.** Prove that the quadratic Bernstein basis ($n = 2$) sums to 1:

$$b_{0,2}(t) + b_{1,2}(t) + b_{2,2}(t) = 1 \quad \text{for all } t \in [0,1].$$

Solution to Exercise 11.1

Expanding:

$$
\begin{aligned}
b_{0,2}(t) + b_{1,2}(t) + b_{2,2}(t) &= (1-t)^2 + 2t(1-t) + t^2\\
&= 1 - 2t + t^2 + 2t - 2t^2 + t^2\\
&= 1 + (-2t + 2t) + (t^2 - 2t^2 + t^2)\\
&= 1 + 0 + 0 = 1.
\end{aligned}
$$

Alternatively, this is the binomial theorem: $(1 - t + t)^2 = 1^2 = 1$. □

Exercise 11.2 (Convex Hull Verification — Linear Algebra)**.** Let $P_0 = (0,0)$, $P_1 = (1,2)$, $P_2 = (3,1)$ and let $\Omega = \{(x,y) \mid x + y \leq 5 \ \wedge \ x \geq 0 \ \wedge \ y \geq 0\}$. Determine whether the convex hull of the control points lies entirely within Ω.

Solution to Exercise 11.2

The convex hull of $\{(0,0),(1,2),(3,1)\}$ is the triangle with these vertices. We check each vertex against the laws:

$$
\begin{aligned}
&(0,0):\ 0+0 = 0 \leq 5\ \checkmark, \quad x = 0 \geq 0\ \checkmark, \quad y = 0 \geq 0\ \checkmark.\\
&(1,2):\ 1+2 = 3 \leq 5\ \checkmark, \quad x = 1 \geq 0\ \checkmark, \quad y = 2 \geq 0\ \checkmark.\\
&(3,1):\ 3+1 = 4 \leq 5\ \checkmark, \quad x = 3 \geq 0\ \checkmark, \quad y = 1 \geq 0\ \checkmark.
\end{aligned}
$$

Since Ω is defined by *linear* inequalities, it is a **convex set**. A convex set contains the convex hull of any of its subsets. Since all three vertices are in Ω and Ω is convex, the entire triangle $\text{conv}(\{P_0, P_1, P_2\}) \subseteq \Omega$.
By Theorem 4.1 and the convexity of Ω, the quadratic Bézier curve through these control points is admissible. We can return `fin` without sampling. □

Exercise 11.3 (First-Violation Witness — Calculus)**.** Consider the cubic Bézier curve with $P_0 = (0,0)$, $P_1 = (1,3)$, $P_2 = (2,-1)$, $P_3 = (3,0)$ and a single law $L(x,y) : y \geq 0$. Find the first-violation time $t^\star$.

Solution to Exercise 11.3

The y-component:

$$\begin{aligned} y(t) &= (1-t)^3(0) + 3(1-t)^2 t(3) + 3(1-t)t^2(-1) + t^3(0) \\ &= 9t(1-t)^2 - 3t^2(1-t) \\ &= 3t(1-t)\big[3(1-t) - t\big] \\ &= 3t(1-t)(3-4t). \end{aligned}$$

The violation occurs when $y(t) < 0$. The factors: $3t \geq 0$ for $t \geq 0$; $(1-t) \geq 0$ for $t \leq 1$; $(3-4t) = 0$ at $t = 3/4$.
For $t > 3/4$: the factor $(3-4t) < 0$ while $3t(1-t) > 0$, so $y(t) < 0$.
Therefore:

$$t^\star = \frac{3}{4} = 0.75.$$

At $t^\star$: $x(0.75) = (0.25)^3(0) + 3(0.25)^2(0.75)(1) + 3(0.25)(0.75)^2(2) + (0.75)^3(3) = 0.140625 + 0.84375 + 1.265625 = 2.25$.
Witness: $W = (1,\ 0.75,\ (2.25, 0),\ \Delta)$ where Δ suggests raising P_2 (the control point with the largest Bernstein weight near $t = 0.75$: $b_{2,3}(0.75) = 3(0.75)^2(0.25) = 0.421875$). □

Exercise 11.4 (Lattice Construction — Discrete Math)**.** Given three laws L_1, L_2, L_3 over $X = \{a, b, c, d, e\}$ with:

$$\begin{aligned} L_1^{-1}(\top) &= \{a, b, c\}, \\ L_2^{-1}(\top) &= \{b, c, d\}, \\ L_3^{-1}(\top) &= \{a, c, d, e\}. \end{aligned}$$

Compute Ω and construct the lattice of all lawful sub-regions.

Solution to Exercise 11.4

$\Omega = L_1^{-1}(\top) \cap L_2^{-1}(\top) \cap L_3^{-1}(\top) = \{a, b, c\} \cap \{b, c, d\} \cap \{a, c, d, e\} = \{c\}$.
All sub-regions:

$$\begin{aligned} \Omega_\varnothing &= X = \{a, b, c, d, e\}, \\ \Omega_{\{1\}} &= \{a, b, c\}, \quad \Omega_{\{2\}} = \{b, c, d\}, \quad \Omega_{\{3\}} = \{a, c, d, e\}, \\ \Omega_{\{1,2\}} &= \{b, c\}, \quad \Omega_{\{1,3\}} = \{a, c\}, \quad \Omega_{\{2,3\}} = \{c, d\}, \\ \Omega_{\{1,2,3\}} &= \{c\}. \end{aligned}$$

Ordered by inclusion, this is a lattice isomorphic to $\mathbf{2}^3$ (the Boolean cube), with meets and joins:

$$\Omega_{\mathcal{S}} \sqcap \Omega_{\mathcal{T}} = \Omega_{\mathcal{S} \cup \mathcal{T}}, \qquad \Omega_{\mathcal{S}} \sqcup \Omega_{\mathcal{T}} = \Omega_{\mathcal{S} \cap \mathcal{T}}. \qquad \square$$

Exercise 11.5 (Sequent Derivation — Formal Logic)**.** Using the inference rules from Section 7.3, construct a complete derivation tree showing that the second attempt in the master example (Section 10.5) yields `fin`.

Solution to Exercise 11.5

Let γ be the adjusted trajectory with $P_0 = (1,1)$, $P_1 = (2,4.5)$, $P_2 = (6,5.5)$, $P_3 = (9,5)$.

Step 1: Establish $P_0 : \texttt{LawfulState}$.

$$\frac{L_1(1,1) = \top \quad L_2(1,1) = \top \quad L_3(1,1) = \top}{(1,1) \in \Omega} \text{ (Lawful-}\wedge\text{)}$$

Step 2: Apply hull pre-check for L_1.

$$\frac{\text{conv}(\{P_k\}) \subseteq [1,9] \times [1,5.5] \subseteq [0,10] \times [0,6] = \mathcal{B}}{\forall t.\ L_1(\gamma(t)) = \top} \text{ (Hull-Check)}$$

Step 3: Subdivision verification for L_2 and L_3. After k subdivision levels:

$$\frac{\texttt{fin}(\gamma|_{[0,0.5]}) \quad \texttt{fin}(\gamma|_{[0.5,1]})}{\forall t.\ L_2(\gamma(t)) = \top \ \wedge\ L_3(\gamma(t)) = \top} \text{ (Subdiv)}$$

(Each half is further subdivided until hull containment in Ω is certified.)
Step 4: Combine via Traj-Adm.

$$\frac{\gamma : [0,1] \to \mathbb{R}^2 \qquad \forall t.\ \gamma(t) \in \Omega}{\vdash \texttt{fin}(\gamma)} \text{ (Traj-Adm)} \qquad \square$$

Exercise 11.6 (Lipschitz Bound — Graduate Analysis)**.** For the master example, compute a Lipschitz constant for the margin function μ and determine the certified safe interval $[0, t_{\min}]$.

Solution to Exercise 11.6

The margin function is $\mu(x,y) = \min(\mu_1, \mu_2, \mu_3)$ where:

$$\mu_1(x,y) = \min(x, 10-x, y, 6-y),$$
$$\mu_2(x,y) = \begin{cases} \min(|x-2|, |x-4|, |y-1|, |y-3|) & \text{if } (x,y) \notin O_1, \\ -d((x,y), \partial O_1) & \text{if } (x,y) \in O_1, \end{cases}$$
$$\mu_3(x,y) = \sqrt{(x-7)^2 + (y-4)^2} - 1.$$

Each μ_i is Lipschitz with constant $K_i = 1$ (distance functions have Lipschitz constant 1). Therefore μ is Lipschitz with $K = 1$.
At $P_0 = (1,1)$: $\mu_1 = \min(1,9,1,5) = 1$; $\mu_2 = |2-1| = 1$ (distance to rectangle); $\mu_3 = \sqrt{36+9} - 1 = \sqrt{45} - 1 \approx 5.71$. So $\mu(P_0) = m_0 = 1$.
The derivative at $t = 0$: $\gamma'(0) = 3(P_1 - P_0) = 3(1, 3.5) = (3, 10.5)$. So $\|\gamma'(0)\| = \sqrt{9 + 110.25} = \sqrt{119.25} \approx 10.92$.
We bound $\|\gamma'\|_\infty \le 3 \max_k \|P_{k+1} - P_k\| = 3 \cdot \|(6,5.5) - (2,4.5)\| = 3\sqrt{16+1} = 3\sqrt{17} \approx 12.37$.
Certified safe interval:

$$t_{\min} \ge \frac{m_0}{K \cdot \|\gamma'\|_\infty} = \frac{1}{1 \cdot 12.37} \approx 0.081.$$

The trajectory is guaranteed admissible on $[0, 0.081]$ without subdivision. Verification need only cover $[0.081, 1]$. $\square$

12 New Mathematics: The Bézier Admissibility Algebra

We introduce novel algebraic structures arising from the interaction of Bézier geometry with constraint systems.

12.1 The Admissibility Monoid

Definition 12.1 (Trajectory Concatenation)**.** Given two admissible trajectories $\gamma_1 : [0,1] \to \Omega$ (from A to B) and $\gamma_2 : [0,1] \to \Omega$ (from B to C), define the concatenation:

$$(\gamma_1 \cdot \gamma_2)(t) = \begin{cases} \gamma_1(2t) & t \in [0, \frac{1}{2}], \\ \gamma_2(2t-1) & t \in [\frac{1}{2}, 1]. \end{cases}$$

Theorem 12.1 (Admissibility Monoid)**.** *Let $\mathcal{A}_\Omega(x,y)$ denote the set of admissible Bézier trajectories from x to y in Ω. Define*

$$\mathcal{M}_\Omega = \bigsqcup_{x,y \in \Omega} \mathcal{A}_\Omega(x,y)$$

*with concatenation $\cdot$ as the binary operation and the constant trajectory $\gamma_x(t) = x$ as the identity at x. Then $\mathcal{M}_\Omega$ is a **category** (a "monoid with many objects"):*

(i) ***Closure****: If γ_1 and γ_2 are admissible and composable, so is $\gamma_1 \cdot \gamma_2$.*
(ii) ***Associativity****: $(\gamma_1 \cdot \gamma_2) \cdot \gamma_3 \sim \gamma_1 \cdot (\gamma_2 \cdot \gamma_3)$ (up to reparametrization).*
(iii) ***Identity****: $\gamma_x \cdot \gamma = \gamma$ and $\gamma \cdot \gamma_y = \gamma$ for $\gamma \in \mathcal{A}_\Omega(x,y)$.*

Proof. Closure: both halves remain in Ω by assumption, and the concatenation is continuous at $t = 1/2$ since $\gamma_1(1) = B = \gamma_2(0)$. Associativity and identity follow from reparametrization of $[0,1]$. □

12.2 The Witness Semilattice

Definition 12.2 (Witness Ordering)**.** For witnesses $W_1 = (i_1, t_1, p_1, \Delta_1)$ and $W_2 = (i_2, t_2, p_2, \Delta_2)$, define

$$W_1 \preceq W_2 \quad \Longleftrightarrow \quad t_1 \leq t_2.$$

The *earliest witness* is the meet: $W_1 \sqcap W_2 = \min(W_1, W_2)$ by time.

Proposition 12.2. *The set of all witnesses for a given trajectory, ordered by $\preceq$, forms a **meet-semilattice**. The meet operation selects the first violation—which is exactly what the verification algorithm returns.*

12.3 Degree Elevation and Law Refinement Duality

Theorem 12.3 (Degree-Refinement Duality)**.** *Degree elevation of a Bézier curve (increasing n without changing the curve's shape) is* dual *to law refinement (adding laws without changing the lawful region) in the following sense: both operations increase descriptive precision without altering the underlying object.*

Formally, let $\iota_n : \mathcal{B}_n \hookrightarrow \mathcal{B}_{n+1}$ be degree elevation and $\rho : \mathcal{L} \hookrightarrow \mathcal{L}'$ be a law refinement with $\Omega_\mathcal{L} = \Omega_{\mathcal{L}'}$. Then the following diagram commutes:

$$\begin{array}{ccc} \mathcal{B}_n \times \mathcal{L} & \xrightarrow{\texttt{verify}} & \{\texttt{fin}, \texttt{finfr}\} \\ \downarrow{\scriptstyle \iota_n \times \rho} & & \| \\ \mathcal{B}_{n+1} \times \mathcal{L}' & \xrightarrow{\texttt{verify}} & \{\texttt{fin}, \texttt{finfr}\} \end{array}$$

Proof. Degree elevation preserves the curve: $\iota_n(B_n)(t) = B_{n+1}(t)$ for all t. Law refinement preserves the region: $\Omega_{\mathcal{L}'} = \Omega_{\mathcal{L}}$. Since verification depends only on the curve's image and the lawful region, the verdict is invariant under both operations. □

This duality suggests a deep connection between the *geometric resolution* of the trajectory and the *logical resolution* of the constraint system—a direction for future investigation.

12.4 The Admissibility Functor

Definition 12.3. Define the **admissibility functor** $\mathcal{F} : \mathbf{Law} \to \mathbf{Cat}$ from the category of law families (with refinement morphisms) to the category of small categories, by

$$\mathcal{F}(\mathcal{L}) = \mathcal{M}_{\Omega_{\mathcal{L}}} \quad \text{(the admissibility category)}.$$

A refinement $\rho : \mathcal{L} \hookrightarrow \mathcal{L}'$ with $\mathcal{L} \subseteq \mathcal{L}'$ induces a faithful functor $\mathcal{F}(\rho) : \mathcal{M}_{\Omega_{\mathcal{L}'}} \hookrightarrow \mathcal{M}_{\Omega_{\mathcal{L}}}$ (more laws means fewer admissible trajectories, giving an inclusion of the stricter category into the more permissive one).

This functorial perspective opens the door to applying tools from category theory—limits, colimits, natural transformations—to the study of constraint-first computation.

13 What is Novel Here?

The novelty claim is *not* "Bézier curves exist" or "constraints exist." It is the unified execution model and the new mathematical structures it reveals:

1. **Execution = trajectory**, not discrete jump plus recovery.
2. **Only lawful states are representable** (invalid states are unexpressible).
3. **Bézier-restricted proposals** provide previewability, local control, and convex-hull bounding.
4. **Witness + repair loop** turns failure into a precise geometric debugging object.
5. **Ledgered proofs** make verification artifacts reproducible and auditable.
6. **Law lattices and Boolean algebras** give compositional structure to constraint systems.
7. **Sequent calculus and typing judgments** formalize the proof obligations.
8. **The admissibility monoid/category** is a new algebraic object capturing safe composition.
9. **Degree-refinement duality** reveals a structural parallel between geometric and logical resolution.
10. **The admissibility functor** connects law refinement to categorical structure.

14 Conclusion

Le Bézier du calcul is computation as bounded geometric execution. By treating constraints as the definition of legality and Bézier trajectories as the medium of change, we obtain systems that:

- Halt safely (`finfr`) rather than mutate illegally.
- Commit (`fin`) only when the *full path* is admissible.
- Provide witnesses—precise geometric objects—when proposals fail.
- Compose algebraically via the admissibility monoid.
- Decompose via the law lattice and dependency graph.
- Formalize via sequent calculus and refinement types.

Version 4.0 demonstrates that this framework is not merely an engineering pattern but a mathematical object in its own right, with structures (lattices, Boolean algebras, categories, functors) that invite further investigation. The master example shows that these structures are not abstract luxuries: they directly inform how proposals are constructed, verified, parallelized, and debugged.

Constraints are not obstacles. They are the geometry of the possible.

Acknowledgments. This work was developed at parcRI Research and Ada Computing Company, Houston, TX.

Version history. V1.0 (initial framework), V2.0 (witnesses and ledger), V3.0 (examples and algorithm), V4.0 (discrete math, formal logic, extended theory, and new mathematics).