Inference Under Constraint

Logic Experiments in Lean

Flyxion
Independent Researcher

2026

Preface

This is a successor volume to *Proof and Countermodel: An Introduction to Logic with Lean*. It assumes that book's method — every claim proved where it holds, paired with an explicit countermodel where a plausible-looking converse or strengthening fails — but not its content. A reader who has completed the introductory sequence needs no further preparation; a reader who has not will find the method explained here only briefly, since it was that earlier book's whole subject.

Where *Proof and Countermodel* stayed classical throughout and stopped at elementary set theory, this book goes three places that book deliberately left aside: logics in which excluded middle or explosion fail, modal structures in which truth depends on which state one is reasoning from, and — the point the whole volume is organized around — the idea that admissibility, repair, and continuation are themselves best understood as a logic, with their own consequence relation, rather than as metaphors borrowed from elsewhere. A closing chapter sketches metatheory: soundness for the fragments developed, decidability where it holds, and a small demonstration of syntax represented and inspected from inside a formal system.

The title names the book's actual concern more directly than *Proof and Countermodel* needed to: what follows from what, once the space of admissible inferences is itself allowed to be governed by constraints that classical logic does not impose.

Contents

Part I

Non-Classical Foundations

Chapter 1

Three-Valued and Paraconsistent Logic

Classical logic assumes every proposition is exactly one of true or false, and that a contradiction proves anything. Both assumptions can be dropped, separately or together, and dropping them yields logics that are still precise and still mechanically checkable — just not classical. This chapter works through the smallest cases: a third truth value standing for a gap in information, and a fourth standing for a genuine glut.

1.1 A Third Value: Kleene's Strong Logic

Experiment: Excluded middle fails with a gap value

Introduce a type with three values — true, false, and *unknown* — and define conjunction, disjunction, and negation by Kleene's rule: an operation returns `unknown` whenever the classical truth table would need to consult an unknown input to decide the result, and returns a definite value whenever one input already settles the answer regardless of the other (for instance, `false and unknown` is `false`, since a false conjunct settles the conjunction no matter what the other conjunct is).

```
inductive K3 where
  | tt | ff | unknown
  deriving DecidableEq, Repr

def K3.knot : K3 → K3
  | tt => ff
  | ff => tt
  | unknown => unknown

def K3.kor : K3 → K3 → K3
  | tt, _ => tt
  | _, tt => tt
  | ff, ff => ff
  | _, _ => unknown

theorem excluded_middle_fails :
    ∃ p : K3, K3.kor p p.knot ≠ K3.tt := by
  exact ⟨K3.unknown, by
    decide ⟩
```

Unlike the classical case, $p \vee \neg p$ is not a validity here: when p is `unknown`, so is $\neg p$, and the disjunction of two unknowns is itself unknown, never forced up to `tt`.

1.2 A Fourth Value: Gluts and the Logic of Paradox

[TODO: extend `K3` to a four-valued type adding a `both` value (Priest's Logic of Paradox, LP) standing for a proposition that is simultaneously true and false rather than neither; show that LP validates excluded middle again (every value is at least true) while classical explosion — from $p \wedge \neg p$ infer anything — fails, using `both` as the countermodel to explosion. Connect explicitly to the Priest–Flyxion constraint-regime logic project as the setting this chapter feeds into.]

1.3 Why Explosion Fails: A Countermodel

Countermodel: Ex falso quodlibet is not valid in LP

[TODO: a concrete instance — $p \wedge \neg p \to q$ — evaluated at $p = \texttt{both}$ and an arbitrary $q = \texttt{ff}$, showing the premise can hold (in the sense of "at least true") while the conclusion fails outright.]

1.4 Exercises

Exercise 1.1. Define Kleene conjunction `K3.kand` and verify by `decide` that `K3.knot` distributes over it the way De Morgan's law requires.

Part II

Modal and Epistemic Structures

Chapter 2

Modal Logic

Modal logic adds operators — canonically $\Box$ (necessity) and $\Diamond$ (possibility) — that make a statement's truth depend on which *state* it is evaluated at, connected to other states by an accessibility relation. This chapter introduces the accessibility picture directly through a structure already familiar from outside logic proper: a fact moving from merely recorded, to admitted, to committed, to consequential, is a chain of states each reachable from the last, and modal vocabulary describes exactly that kind of movement.

2.1 States and Accessibility

[TODO: $\Box$/$\Diamond$ as quantification over accessible states; the accessibility relation as a plain Lean relation on a state type; worked experiment building a three-state Kripke frame and evaluating a formula at each state by hand before automating it.]

2.2 A Worked Structure: Record, Admit, Commit

Experiment: The forward chain

A fact can be recorded without being admitted, admitted without being committed, and committed without being consequential — but never the reverse. State this as a chain of implications between four propositional fields of one structure.

```
structure State where
  recorded : Prop
  admitted : Prop
  committed : Prop
  consequential : Prop

def valid (s : State) : Prop :=
  (s.admitted → s.recorded) ∧
  (s.committed → s.admitted) ∧
  (s.consequential → s.committed)

theorem consequence_requires_record
    (s : State) (h : valid s) (hc : s.consequential) :
    s.recorded :=
  h.1 (h.2.1 (h.2.2 hc))
```

Read modally: if "consequential" is accessible from "committed," "committed" from "admitted," and "admitted" from "recorded," then $\Box$ at the recorded state reaches every later state in the chain — but nothing forces the reverse accessibility.

Countermodel: None of the converses hold

Recording does not imply admission; admission does not imply commitment.

```
def recordedOnly : State where
  recorded := True
  admitted := False
  committed := False
  consequential := False

theorem record_does_not_imply_admit :
    ¬ ∀ s : State, valid s → s.recorded → s.admitted := by
  intro h
  exact h recordedOnly (by simp [valid, recordedOnly]) (by simp [
   recordedOnly])
```

`recordedOnly` is a single state satisfying `valid` in which every later stage of the chain is false — exactly the kind of witness a countermodel to a converse needs, and exactly the reason modal accessibility is not assumed symmetric by default.

2.3 Necessity, Possibility, and Frame Conditions

[TODO: standard frame conditions (reflexivity, transitivity, symmetry) and the modal axioms they correspond to (T, 4, B); worked experiment showing that transitivity of the record/admit/commit accessibility relation is exactly what licenses treating `consequential` as forcing `recorded` two steps removed, via `extends_trans`-style composition from Chapter 3.]

2.4 Exercises

Exercise 2.1. Show that admission does not imply commitment, following the pattern of `record_does_not_impl` above.

Part III

Admissibility as a Logic

Chapter 3

Admissibility as a Logic

The preceding two chapters relaxed classical logic in two directions independently: extra truth values, and truth relativized to a state. This chapter's claim is that a third, related idea — admissibility — is not a metaphor borrowed from those two but a logic in its own right, with its own notion of what follows from what. A history of witnessed facts only grows; a constraint can narrow which future continuations are admissible without rewriting anything already witnessed. Both are ordinary logical claims once stated precisely, and both are provable rather than merely evocative.

3.1 Histories and Extension

Experiment: Extension is a preorder

```
structure History (alpha : Type) where
  witnessed : Set alpha

def Extends {alpha : Type} (earlier later : History alpha) : Prop :=
  earlier.witnessed ⊆ later.witnessed

theorem extends_refl {alpha : Type} (h : History alpha) :
    Extends h h := fun x hx => hx

theorem extends_trans {alpha : Type} {h1 h2 h3 : History alpha}
    (h12 : Extends h1 h2) (h23 : Extends h2 h3) :
    Extends h1 h3 := fun x hx => h23 (h12 hx)

theorem witnessed_facts_persist {alpha : Type}
    {earlier later : History alpha}
    (h : Extends earlier later) {fact : alpha}
    (w : fact ∈ earlier.witnessed) : fact ∈ later.witnessed :=
  h w
```

Reflexivity and transitivity make `Extends` a preorder on histories — the same structural fact that, read as an accessibility relation, would license the transitive modal reasoning flagged at the end of Chapter 2.

3.2 Constraint as Monotone Narrowing

Experiment: Constraining never enlarges what is admissible

```
def Constrain {alpha : Type} (admissible constraint : Set alpha) :
   Set alpha :=
  admissible ∩ constraint

theorem constrain_subset {alpha : Type}
    (admissible constraint : Set alpha) :
    Constrain admissible constraint ⊆ admissible := fun x hx => hx.1
```

This is deliberately almost too simple to call a theorem — the point is that it *is* one: "constraint can only narrow, never expand, what a process may still do" is not asserted rhetorically anywhere in this framework, it is this one-line fact, checked the same way every other claim in this book has been checked.

3.3 A Consequence Relation for Admissibility

[TODO: define a genuine consequence relation $\vdash_A$ on statements about a history/constraint pair, prove it reflexive and transitive (structural rules any logic needs), and show it is *not* monotone in the classical sense — adding a constraint can remove previously admissible conclusions — connecting explicitly to why this counts as a logic distinct from, not merely inspired by, classical or modal logic.]

3.4 Exercises

Exercise 3.1. Prove that `Extends` is *not* antisymmetric in general: exhibit two distinct histories that extend each other.

Part IV

Metatheory

Chapter 4

Metatheory

The preceding three chapters develop logics; this closing chapter asks questions *about* them. Soundness, decidability, and the representability of syntax as an ordinary object are not full formal developments here — that would be its own book — but each gets a concrete, checkable instance rather than a purely expository treatment.

4.1 Soundness for the Propositional Fragment

[TODO: state soundness for the Kleene fragment of Chapter 1 — every provable formula is true under every valuation, restricted to the truth values `K3` actually admits — and prove the base cases directly; discuss why soundness for a non-classical logic still requires checking each connective's introduction rule against its own semantics, not the classical one.]

4.2 Decidability

Experiment: Propositional validity over `K3` is decidable

[TODO: since `K3` is a finite type, any formula in finitely many propositional variables has a decidable validity question over it by exhaustive case analysis — the same `decide`-based argument used throughout *Proof and Countermodel*, now applied to a non-classical semantics rather than `Bool`.]

4.3 Syntax as an Object: A Small Gödel-Numbering Demonstration

[TODO: encode a small formula type as a natural number via a Cantor-pairing-style scheme, define a decidable predicate testing whether a number encodes a well-formed formula, and demonstrate — without attempting anything like a full incompleteness proof — that a formal system can talk about its own syntax as data rather than only as expressions it is currently manipulating.]

4.4 Where This Leaves the Admissibility Program

[TODO: closing discussion connecting back to Chapter 3 — what a soundness or decidability result for the admissibility consequence relation $\vdash_A$ would need to establish, and why it is left as future work rather than attempted here.]

Lean Setup and Conventions

[TODO: as in *Proof and Countermodel*'s appendix — minimal `lakefile.toml`/`lean-toolchain` setup; a note on the inductive-type style used for `K3` in Chapter 1, since it differs from the `Bool`-valued register that dominated the introductory book.]

Bibliography

[1] G. Priest, *An Introduction to Non-Classical Logic: From If to Is*, 2nd ed., Cambridge University Press, 2008.

[2] G. Priest, *In Contradiction: A Study of the Transconsistent*, 2nd ed., Oxford University Press, 2006.

[3] P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*, Cambridge Tracts in Theoretical Computer Science **53**, Cambridge University Press, 2001.

[4] S. C. Kleene, *Introduction to Metamathematics*, Van Nostrand, 1952.