CNNA – Gesamt-Plan für Refactoring und Strengthening des strikten Neuaufbaupfads

Knappfassung auf Basis des aktuellen Repos

Jan Seeck (antaris) als Editor, ChatGPT als Ghostwriter

Stand: 12. April 2026

1 Leitsatz

Empfehlung: kein Totalneustart des Repos, aber ein **aktiver Pfad-Neuaufbau** im selben Repo. Der Refactor-Schritt 1 ist mit grünem P27 als Architektur-, Cutover- und Entkopplungsschritt abgeschlossen. Der nun folgende Schritt 2 ist das **Strengthening**: Aus dem bereinigten Pillar-A-Pfad soll ein tatsächlich rechnender, produktiv nutzbarer Generator werden, der nicht nur eine isolierte Smoke-Test-Instanz, sondern einen konkreten robusten Referenzfall des IDEAL-ToC und parallel dazu eine Familie legaler Substratinstanzen tragen kann. Der alte Pfad dient nur noch als **Regressionsorakel**; nach erfolgreichem modulweisem Cutover ist er aus dem aktiven Build-Pfad nach `legacy_sources/CNNA_v0.100_unstable` verschoben. Die zwischenzeitliche dünne Step-1-Gate-Stufe war nur Abschlussaggregator und ist nach grünem P27 selbst in die Archivspur überführt; die aktive Produktionslogik liegt nun vollständig in den fachlich zuständigen Pillar-A-Modulen. Strengthening bedeutet dabei ausdrücklich nicht, einen Smoke-Test-Seitenpfad einzuziehen, sondern dieselbe geared Hauptkette von der Wurzel bis zum Handoff so zu verschärfen, dass sie als später tragender Produktionspfad bestehen bleibt.

Prinzipien

- **Derived-only lokal:** jedes aktive Modul trägt sein starkes Artefakt selbst.
- **Wurzelprinzip:** derived-only beginnt beim fundamentalsten Objekt des aktiven Pfads und läuft strikt nach vorn.
- **Geared global:** jeder Nachfolger ist der kanonische Consumer des Vorgängers; keine freien Seiteneingänge und keine Bypässe.
- **Import-Vorabprüfung:** tragende Imports müssen selbst bereits derived-only, nicht-zirkulär und nicht vorausnehmend sein.
- **Legacy-Politik:** nur direkt oder fast direkt migrierbare Legacy-Dateien werden portiert; alles andere wird neu geschrieben.
- **Archivdisziplin:** nach erfolgreichem Cutover wird alter Parallelcode aus dem aktiven Build-Pfad in `legacy_sources/CNNA_v0.100_unstable` verschoben; er bleibt dort als Regressionsorakel erhalten, wird aber nicht dauerhaft als aktive Doppelspur mitgeführt.
- **Ideal-zuerst-Prinzip für das ToC-Substrat:** Das ToC wird im aktiven Pfad zuerst als ideales unendliches Substrat modelliert. Reale endliche Objekte sind davon strikt abgeleitete Approximanten und nicht umgekehrt.
- **Fixpunktprinzip für den Grenzpfad:** Die intendierte Aussage $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$ ist nicht als rohe Gleichsetzung zweier Typen zu lesen, sondern als gerichtete Rekonstruktion des idealen ToC aus einer Approximantenfamilie. Diese Lesart umfasst ausdrücklich das Verschwinden aller Rand-, Cutoff- und Umgebungsartefakte im idealen Fixpunkt sowie die Kontraktion aller endlichen Zellen ins Infinitesimale unter wachsendem Umgebungsdruck.

- **Dualpräsentationsprinzip für** IDEAL_∞**:** abstrakter Idealvertrag und adressierte Präsentation werden nicht roh identifiziert, sondern über eine strikt strukturtreue Äquivalenz mit beidseitigen Transfersätzen verbunden.
- **Kanonizitätsprinzip für die Box-Inhalte:** P4 schließt zunächst nur die kanonische Adressierung der idealen Zellseite. Die kanonische Etikettierung des gesamten "Box-Inhalts" – insbesondere Kanten, Schnitte, Approximanten, Rand, Komplement/Umgebung, IR-Anteil und UV-tail – darf erst als derived-only Folge der P5-invarianten Transportdaten und ihrer späteren Consumer entstehen.
- **Generatorprinzip für Schritt 2:** der produktive Smoke-Test darf kein Sonderpfad sein. Referenzfall und Substratvariationen müssen denselben Downstream-Pfad konsumieren, der später auch die Pillar-Handoffs trägt.
- **Familienprinzip für das Substrat:** neben einem robusten konkreten Referenzfall des IDEAL-ToC muss der aktive Pfad eine Familie legaler Substratinstanzen tragen; die Variation ist nur dann zulässig, wenn dieselbe Contract-/Addressing-/Equiv-Logik und dieselben späteren Consumer erhalten bleiben.
- **Keine freien Operatorzeugen im Produktionspfad:** freie Witnesses wie Gewichte, Inversen oder Regularisierungsshifts sind im Strengthening nicht bloß hübsch zu kapseln, sondern soweit möglich in legale Provenienzketten des aktiven Pfads zu internalisieren.
- **Inputvektorprinzip des Generators:** produktive Pillar-A-Läufe variieren nicht nur über das Substrat, sondern über den Dreiklang *(Substrat, WeightPolicy, Cutoff)*; universelle Struktursätze sind deshalb von genuinely variationsabhängigen Aussagen explizit zu trennen.
- **Computability-Budget-Prinzip:** für jedes Zahnrad des Root-to-Handoff-Generators ist explizit festzuhalten, ob es `computable`, nur intern lokal `noncomputable` oder nach aktuellem Stand noch offen ist; der globale Generator darf nicht schleichend durch viele scheinbar lokale Ausnahmen in einen diffusen `noncomputable`-Status kippen.
- **Parallel-Dualstrang-Prinzip für algebraische Seeds:** wo eine A-seitige endliche algebraische oder operatorische Oberfläche zugleich als ausführbare Generatorwurzel und als spätere analytische Consumer-Oberfläche benötigt wird, ist sie als expliziter Parallelstrang zu modellieren: ein operativer, öffentlich computabler Produktivstrang über einem ausführbaren Koeffiziententyp (im Strengthening zunächst `ExecComplex`) und ein davon getrennter, gegebenenfalls `noncomputable`er Spiegelstrang über **C**. Beide Stränge dürfen nur über eine bewiesene strukturtreue und injektive Brücke miteinander verbunden werden. Der operative Generatorpfad konsumiert ausschließlich den computablen Strang; der **C**-Strang dient Analyse, Transfer und späteren Pfeilern, darf aber nicht stillschweigend zum operativen Ausgang des strikten Pfads werden.
- **Instanzpropagationsprinzip für endliche Träger:** jedes Modul, das endliche Carrier, Teilmengen oder Prädikate über endlichen Typen exponiert, muss die nötigen `Fintype`-, `DecidableEq`- und `Decidable`-Instanzen explizit mitliefern oder theorematisch zurückgewinnen; vermeidbare `classical`-Abkürzungen sind im aktiven Pfad nicht zulässig.
- **A-seitige Vorstufenpflicht für spätere Pfeiler:** wenn spätere Pfeiler spektrale, thermische, zustandsräumliche, kanal- oder limesseitige Werkzeuge benötigen, die direkt aus A-generierten endlichen Operator- und Trägerdaten hervorgehen, dann sind diese als finite Seeds bereits in Pillar A bereitzustellen und nicht in B/C/D/E nachzubauen.
- **Algebraisches Wurzelobjektprinzip:** soweit spätere Pfeiler *-Algebra, Adjungierte/Hermitizität, endliche Hilbertraumstruktur, Kommutatorlogik, Zeitentwicklung oder Zustandsraum direkt aus endlichen A-seitigen Matrixcarriern und Operatoren gewinnen können, sind diese Strukturen bereits in A als fachlich getrennte Seeds zu liefern.

- **Legacy-Extraktionsprinzip jenseits von A:** aus REALOQS dürfen allgemeine mathematische Lemmata, Bright-Recovery-Vergleichsdaten, BInterface-Ideen und spätere Seed-Kandidaten extrahiert werden; unzulässig ist dagegen jeder Rückfall in boundary-first-Semantik, A→B-Zirkularität oder fachlich querliegende Sammelmodule.
- **Pfeilersemantik-Regel:** A liefert nur solche theoremisierte Axiomsamen, die aus A-generierten Größen derived-only erzeugt werden können; alles, was darüber hinaus echte AQFT-, OQS-, Geometrie- oder Matter-Fachlogik ist, wird in späteren Pfeilern aufgebaut und im Plan ausdrücklich als Folgearbeit getrennt geführt.
- **Fachmodulprinzip:** auch im Strengthening trägt jedes neue Modul eine zentrale Aussage bzw. Fachlogik. Höhere Modulanzahl ist zulässig, wenn dadurch Verzahnung, Wartbarkeit und Lesbarkeit steigen; verdeckte Sammelmodule sind zu vermeiden.
- **Gate-statt-Zeitplan-Prinzip:** es gibt bewusst keinen Kalender-, Deadline- oder Aufwandskalender. Angearbeitet wird stets das naechste fachlich anstehende Zahnrad; Fortschritt wird ueber grüne Builds, geschlossene Gates, explizit reduzierte offene Whitelist-Reste und gegluecke Referenz-/Variationslaeufe gemessen, nicht ueber Datumsversprechen.
- **Toolchain-Pinning-Prinzip:** der aktive Strengthening-Pfad arbeitet gegen eine gepinnte Lean-/Mathlib-Toolchain des Repos; fuer den aktuell geprueften Stand ist dies Lean 4 `v4.28.0` mit `mathlib4` auf `v4.28.0` samt im `lake-manifest` fixierter Revisionslage. Upgrades duerfen nur in einer eigenen Kompatibilitaetsphase mit gruener Vollregression erfolgen und nicht still mitten im Abarbeitungspfad.
- **Editor-/Reviewer-Realitaetsprinzip:** der Plan ist so formuliert, dass er editorgefuehrt und single-thread robust abarbeitbar bleibt; AI ist Heuristik- und Ghostwriter-Schicht, spaetere fachliche Reviewer und moegliche weitere Contributor werden ausdruecklich erwartet, aber der Plan setzt ihre sofortige Verfuegbarkeit nicht voraus.
- **Audit-Durchsetzungsprinzip:** Regelkonformitaet ist nicht nur proseartig zu fordern, sondern lokal build- und grep-pruefbar zu halten: `lake build`, Importgraph, sowie scriptbare Audits auf `sorry`/`admit`/freie Axiome, `noncomputable`, `classical` und pauschale `@[simp]`-Flaechen gehoeren zu den Gates; CI ist nuetzlich, aber nicht Voraussetzung fuer die Geltung der Regel.

2 Projektvollzug, Referenzobjekt, Toolchain und Fallbacks

Warum es keinen Zeitplan gibt

Dieser Plan ist bewusst *kein* Kalenderplan. Die Roadmap selbst arbeitet pfeilerweise ueber Gates und partielle Ordnung statt ueber fixe Meilenstein-Daten; Strengthening folgt derselben Logik. Bearbeitet wird jeweils das fachlich naechste anstehende Modul bzw. die naechste Generatorengstelle. Ob ein Arbeitspaket Tage oder Monate braucht, ist nicht im Voraus zu versprechen; auf Kurs ist das Projekt dann, wenn der aktive Build grün bleibt, Gates sauber schliessen, Whitelist-Reste explizit schrumpfen und Referenz- sowie Variationslaeufe die jeweils behauptete Verzahnung wirklich tragen.

Arbeits- und Reviewer-Modell

Der aktuell realistische Vollzugsmodus ist editorgefuehrt: Jan Seeck (antaris) als Editor und Projektfuehrung, ChatGPT als heuristische Strukturierungs- und Ghostwriter-Schicht, spaetere fachliche Reviews fuer Mathematik, AQFT, OQS, Geometrie und Matter als ausdruecklich gewuenschte, aber nicht fuer jedes Zwischen-Gate vorausgesetzte Stufe. Der Plan ist deshalb absichtlich so geschrieben, dass er auch ohne grosses Parallelteam kohärent abarbeitbar bleibt; jede Phase soll fuer sich lokal prüfbar und ruecksetzbar sein.

Gepinnte Toolchain statt gleitender Zielscheibe

Die aktuell gepruefte Repo-Lage ist toolchain-gepinnt: `lean-toolchain` fixiert Lean 4 `v4.28.0`, das `lakefile` bindet `mathlib4` an `v4.28.0`, und das `lake-manifest` fixiert die konkrete Revisionslage. Der Plan nennt diese Pinning-Strategie explizit, damit `mathlib`-Updates nicht unbemerkt Beweise oder Typklassenlagen verschieben. Jede kuenftige Versionserhoehung ist als eigene Kompatibilitaets- und Regressionsphase zu behandeln; waehrend einer laufenden S-Phase gilt die gepinnte Toolchain als normativer Zielzustand.

Was das konkrete IDEAL-ToC im Plan ist

Der Strengthening-Plan braucht einen *konkreten* robusten Referenzfall des IDEAL_∞, ohne daraus die gesamte Stammtheorie auf einen einzigen Substratstil zu verengen. Referenzobjekt ist deshalb der einfachste fachlich belastbare Fall eines **wurzelbasierten, unendlichen, regulär verzweigenden ToC** mit adressierbarer Zellfamilie; im praktischen Referenzlauf ist dies der reguläre b-aere rooted Tree-of-Cliques-Fall. Die zweite legale Substratfamilie bleibt keine unbekannte Blackbox, sondern ist im Plan bereits als **levelabhaengig verzweigender Baum** mit Branching-Funktion $\beta : \mathbf{N} \to \mathbf{N}$ vorgezeichnet. Weitere Substratfamilien sind zulaessig, aber erst dann architektonisch relevant, wenn sie denselben Contract-/Addressing-/Equiv-Pfad instanziieren.

Physikalischer Zieltyp des Programms

Das Projekt behauptet nicht, in Pillar A bereits eine volle emergente Raumzeit- oder AQFT-Theorie zu schliessen. Der belastbare A-seitige Zieltyp ist strenger und schmaler: A soll einen theoremisierten, berechenbaren und zielneutralen Generator liefern, aus dem spaetere algebraische, thermische, dynamische, kanalische, limesseitige und Diagnose-Samen derived-only hervorgehen. Spaetere starke Physikaussagen – etwa kontinuierliche Raumzeitsymmetrien, modulare Darstellungsarchitekturen, Typ-III-, CPT-, Kramers- oder Matter-/Gauge-Lesarten – gehoeren erst in B/C/D/E. Explorativ ist das Programm nur im Sinn spaeter physikalischer Emergenz; im Sinn der A-seitigen Generator- und Seed-Architektur ist es gerade *nicht* offen oder beliebig.

Konvergenzbegriff fuer $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$

Der Plan versteht diese Lesart zunaechst **algebraisch und gerichtet**, nicht als bereits abgeschlossene topologische Vollendung. `DirectedLimit` und spaetere `InfiniteCarrier-`/ `PreNet`-Seeds sollen stabile endliche Ausschnitte, Restriktions- und Uebergangsmorphismen sowie Eindeutigkeits- und Verträglichkeitsdaten formalisieren. Erst auf spaeteren Pfeilern wird daraus eine quasilokale, topologische oder operatoralgebraische Vervollstaendigung. Damit ist der Limesbegriff in A wohldefiniert genug fuer Export und Diagnose, ohne analytisch mehr zu behaupten als der aktuelle Pfad traegt.

Warum raw und stabilized getrennt bleiben

Der rohe Pfad ist im Plan nicht dekorativer Ballast. Er bleibt als Audit-, Diagnose-, Monotonie- und Regressionsflaeche sichtbar, damit die Stabilisierung nachpruefbar als kontrollierte Transformation derselben Provenienz erscheint und nicht als semantischer Austausch des Operators. Operativ konsumieren spaetere Zahnräder nur den stabilisierten Pfad; der rohe Pfad bleibt fuer Vergleich, Regularisierungsprovenienz, Kontraktionsexperimente und moegliche Fallback-Analysen erhalten.

Fallback- und Eskalationsregeln

- **Solver-Fallback:** Wenn `InteriorInverse` oder `interfaceInverse` in einer Phase nicht voll internalisierbar sind, darf als *einziger* Fallback ein lokaler interner Solver-/Reduktionsvertrag mit whitelist-faehigem, verborgenem `noncomputable`-Zeugen stehen bleiben. Unzulaessig bleibt dagegen, solche Inversen als freie oeffentliche Strong-Felder oder gar als spaeten Seiteninput zu behalten.

- **Mathlib-Fallback:** Falls `mathlib` benoetigte Spektral-, Matrixexponential- oder Algebra-Fakten nicht in brauchbarer Form liefert, werden zunaechst lokale CNNA-Hilfssaetze ueber der gepinnten Toolchain gebaut. Ein Fork oder Upstream-Patch ist erst letzte Eskalationsstufe, wenn die lokale Kapselung die Fachmodulgrenze nicht mehr sauber halten kann.
- **Variations-Fallback:** Falls eine weitergehende Substratfamilie noch nicht schliessbar ist, darf der Hauptpfad nicht blockiert werden: Referenzfamilie plus bereits legalisierte Variationsfamilie muessen weiter grün laufen; jede zusaetzliche Familie wird dann als separater nachgezogener Variationsblock behandelt.
- **Planstil-Fallback:** Die wiederkehrenden P0–P27-Schlussformeln im Bestandsblock sind bewusst Audit-Ledger und keine überfluessige Stilwiederholung. Wo alle Phasen dieselbe Restriktionslogik pruefen muessen, ist die nahezu formelhafte Wiederholung Teil der Nachvollziehbarkeit; verkuerzt werden darf nur, wenn dieselbe Gate-Pruefbarkeit erhalten bleibt.

3 Harte Repo-Befunde

- `CNNA/Gates/PillarAStep1Bridge.lean`: 1792 LOC, 40 `def`, 222 `theorem`, 28 `native_decide`, 1115 `concrete`-Treffer. Folge: die Brücke ist zu schwer.
- `CNNA/Core/Step1StrongCore.lean` enthält bereits echte Kopplung. Folge: eine brauchbare Wirbelsäule existiert.
- `CNNA/Core/ABHandoffStrong.lean` ist noch zu schmal. Folge: die A→B-Abgangswelle ist noch zu kurz.
- `CNNA/AQFT/*` ist überwiegend Stub-Hülle. Folge: Schritt 2 darf nicht auf der jetzigen AQFT-Schicht aufsetzen.
- Der Legacy-Bestand von REALOQS zeigt zugleich, dass extttStarAlgebra, extttState, extttGNS, extttKMS, extttLocalNet, extttStateNet, Kanal- und Lindblad-Pfade später tatsächlich benötigt werden, im Altbaum aber fachlich teils quer zu Derived-Modulen, extttBInterfaces, Bright-Recovery und Beispielen verteilt sind. Folge: Strengthening muss die A-seitig generierbaren algebraischen und handoffnahen Vorstufen in eigene Fachmodule zurückholen, statt sie später in B/C erneut kreuz und quer aufzubauen.
- `CNNA/Core/ToC/Concrete.lean` importiert `DirichletLaplacian`; `CNNA/OQS/DtN.lean` importiert `RegionNet`. Folge: der heutige Pfad ist schichtungsmäßig verschmiert und muss entkoppelt werden.
- Der neue Pfad `CNNA/PillarA/Foundation/SubstrateClass.lean` trägt bereits einen ersten idealen Substratkern mit Wurzel auf Ebene 0, Eltern-/Kinder-Kohärenz, Nichtterminalität, dynamischen Branching-Daten, `refine/coarsen`-Vorstufe und expliziten unendlichen Fäden; `CNNA/PillarA/Foundation/MatrixNorms.lean` ist kein Stub mehr. `MatrixNorms` ist damit der erste natürliche Anwendungsfall des algebraischen Dualstrangs: auf der operativen `ExecComplex`-Seite muss das Modul eine explizit computable Shift-, Vergleichs- und Positivitätsoberfläche (insbesondere Frobenius-Quadrat, Nulltest, Positivität bei $\neq 0$ und daraus konsumierbare Shift-Stufe) tragen, während analytische Frobenius-/Normfakten über $\mathbf{C}$ nur als getrennter Spiegelstrang sichtbar bleiben dürfen. Folge: P1 ist als Foundation-Vorstufe geschlossen, aber die leere-Ursprung-/Fixpunktsemantik gehört noch präzise in P3; zugleich ist die bisherige MatrixNorms-Sonderregel in die allgemeine Dualstrang- und Whitelist-Logik zu überführen.
- `CNNA/Core/ToC/Contract.lean` importiert `ToC/Concrete`. Folge: P3 ist nicht als reiner ToC-Vertrag realisiert.
- Der aktuelle aktive ToC-Pfad modelliert noch kein ideales unendliches ToC, sondern ein endliches,

ad-hoc parametriertes Ersatzobjekt. Folge: die Grenzlesart $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$ ist ontisch noch nicht sauber aufgesetzt.

- Mit grünem P3 liegt nun ein erster formalisierter abstrakter Vertrag für IDEAL_∞ auf dem neuen Pfad vor. Folge: P0–P3 werden nicht zurückgenommen; stattdessen ist vor der vollen Approximantenfamilie explizit eine Adressierungs- und Äquivalenzschicht einzuplanen.

- Mit grünem P4 steht nun eine erste kanonische Adressierung der idealen Zellseite. Folge: diese Kanonizität darf noch nicht auf den gesamten "Box-Inhalt" extrapoliert werden; erst P5–P17 dürfen daraus schrittweise kanonische Identifikationen für Schnitte, Approximanten, Rand-/Komplementdaten sowie IR/UV-Anteile ableiten.

- Mit nun grünem P0–P8-Stand trägt der aktive neue Finite-Pfad bereits eine frühe Disziplin für `simp`-Politik, explizite Rekonstruktionssätze und entscheidbare adressierte Schnitte: die Zylinder-/Teilbaum-Schnittseite ist im aktiven Pfad computable, `RegionCore` und `BoundaryPorts` sind als getrennte Consumer-Stufen geschlossen, und der verbleibende nichtkonstruktive Rest sitzt nur noch in den unvermeidbaren Matrixnorm-Fakten. Folge: P9 darf auf einem bereits bereinigten konsumierenden Box-Pfad aufsetzen und soll diese Disziplin nicht wieder aufweichen.

- Mit nun grünem P0–P9-Stand ist `Finite/Approximant` als reine Consumer-Stufe von `BoundaryPorts` geschlossen: `ApproximantStrong` baut weder einen zweiten `Concrete`-Kern noch einen Seitenpfad zurück nach `RegionCore` auf, sondern fixiert den bereits kanonisch gewonnenen endlichen Box-Inhalt nur noch zum gewählten Cutoff. Die anfänglich zu breite `@[simp]`-Oberfläche wurde wieder zurückgenommen; im aktiven P9-Pfad verbleiben weder neue `noncomputable`-Restquellen noch klassische Hilfsabkürzungen. Folge: P10 bleibt rein bedingtes Supportmodul und wird nur aktiviert, wenn P11 es tatsächlich benötigt; andernfalls läuft der Hauptstrang direkt zu P11.

- Mit nun grünem P0–P10-Stand ist `Finite/Selection` als explizit isoliertes Supportmodul über `ApproximantStrong` geschlossen: `SelectionStrong` erzeugt keinen Alternativpfad, rekonstruiert weder `Concrete` noch `RegionCore`/`BoundaryPorts` neu und bleibt strikt Consumer des bereits kanonisch fixierten Box-Inhalts. Im aktiven P10-Pfad wurden keine neuen `noncomputable`-Restquellen, keine klassischen Hilfen und keine pauschalen `@[simp]`-Wrapper eingeführt; die modulnahe Notation bleibt parameter-sichtbar und rein lesbar. Folge: P11 kann `Selection` bei echter Teilstruktur-Auswahl konsumieren, muss dies aber architektonisch nicht; der Hauptstrang bleibt auf den Dirichlet-Kern fokussiert.

- Mit nun grünem P0–P11-Stand ist `Finite/DirichletLaplacian` als reiner Operator-Consumer über dem bereits kanonisch fixierten endlichen Box-Inhalt geschlossen: `DirichletLaplacianStrong` baut keinen neuen `Concrete`-/`Region`-Kern auf, benutzt `Selection` nicht als versteckte Basis und führt keine neuen klassischen Hilfen, pauschalen `@[simp]`-Flächen oder zusätzlichen `noncomputable`-Restquellen in den aktiven Pfad ein. Der operative Träger des Dirichlet-Kerns ist dabei nicht nur eine Topschicht, sondern ein gebundener endlicher Box-Träger über allen Levels $\leq$ Cutoff; dies fixiert zugleich die saubere Blockzerlegung für den nachfolgenden DtN-Kern. Folge: P12 kann nun direkt als binärer DtN-Consumer von `DirichletLaplacianStrong` aufsetzen, ohne Rückimport von `RegionNet` oder Rückweichung auf Alt-/Bridge-Pfade.

- Mit nun grünem P0–P12-Stand ist `DtN/DtN` als reiner binärer DtN-Consumer über `DirichletLaplacianStrong` auf dem neuen Pillar-A-Pfad geschlossen: `DtNStrong` konsumiert nur den bereits kanonisch fixierten Dirichlet-Kern, führt keine Rückimporte von `RegionNet`, keine neuen `@[simp]`-Flächen und keine neuen `noncomputable`-Restquellen in den aktiven Pfad ein. Mit `InteriorInverse`, `boundaryOperator`, `interiorSolve` und den expliziten Abtriebssätzen zur Randfluss- und Innenblock-Gleichung steht damit die proof-carrying Oberfläche für P13 bereit. Folge: `DtNStabilized` darf nun nur noch auf diesem neuen P12-Pfad aufsetzen; der alte `CNNA/OQS/DtN`-Pfad bleibt bis zum Downstream-Cutover reines Regressionsorakel.

- Mit nun grünem P0–P13-Stand ist `DtN/DtNStabilized` als explizite Stabilisierungsstufe über

dem neuen binären DtN-Pfad geschlossen: `DtNStabilizedStrong` trennt rohen Randoperator, symmetrisierte Hilfsstufe und primäre stabilisierte Operatoroberfläche explizit, fällt weder auf `CNNA/OQS/DtN.lean` zurück noch konsumiert es versehentlich den **C**-Spiegelstrang von `MatrixNorms`. Die operative Shift-Regularisierung bleibt explizit computable und stammt ausschließlich aus der `ExecComplex`-Seite von `MatrixNorms`; lesbare Notation darf diese Provenienz nicht verdecken. Folge: P14 kann auf einer semantisch schärferen Operatoroberfläche aufsetzen, ohne spätere Symmetrie-, Cutoff- oder Umgebungsdiagnostik bereits in P13 zu verwischen.

- Mit nun grünem P0–P14-Stand sind `Sectors/BranchPatch` und `Sectors/ComplementSectorFamily` als separate ontische Vorstufe über dem neuen P13-Pfad geschlossen: `BranchPatchStrong` konsumiert nur `DtNStabilizedStrong`, `ComplementSectorFamilyStrong` nur `BranchPatchStrong`; die kanonische äußere Umgebung/Komplementseite der Box wird über den äußeren Support explizit familienfähig gemacht, und die Datenoberfläche unterscheidet theorematisch zwischen root-zentrierten Ausschnitten ohne äußere Umgebung und fensterartigen Ausschnitten mit echter Umgebung/Komplementseite. Im aktiven P14-Pfad wurden keine neuen `noncomputable`-Produktionsdefinitionen, keine klassischen Hilfen und keine pauschalen `@[simp]`-Flächen eingeführt. Folge: P15 kann `SectorSplit` auf einer bereits kanonisch markierten Umgebungslage aufbauen, statt diese Unterscheidung erneut implizit zu rekonstruieren.

- Mit nun grünem P0–P15-Stand ist `Sectors/SectorSplit` als geometrisch-ontische Sektorzerlegung über der in P14 kanonisch gewonnenen Umgebungs-/Komplementseite geschlossen: `SectorSplitStrong` konsumiert nur `ComplementSectorFamilyStrong`, rekonstruiert die helle Seite strikt aus `source.patch` und baut keine konkurrierende Umgebungsdiagnostik auf. Die aktive Oberfläche stellt helle, Interface- und dunkle Carrier sowie die daraus derived-only rekonstruierte Interface-Region/-BoundaryPorts bereit; Präsenz bzw. Abwesenheit einer äußeren Umgebung ist theorematisch über die dunkle Seite unterscheidbar. Im aktiven P15-Pfad wurden keine neuen `noncomputable`-Produktionsdefinitionen, keine klassischen Hilfen und keine pauschalen `@[simp]`-Flächen eingeführt. Folge: P16 kann `BranchingWitness`/`SelectedBranching` direkt als proof-carrying Consumer von `SectorSplitStrong` formulieren und darf dabei weder die Umgebungsdiagnostik noch die Sektortrennung erneut aus Rohdaten rekonstruieren.

- Mit nun grünem P0–P16-Stand sind `Sectors/BranchingWitness` und `Sectors/SelectedBranching` als proof-carrying Witness-/Auswahlstufe direkt über `SectorSplitStrong` geschlossen: `BranchingWitnessStro` rekonstruiert aus der bereits kanonischen Sektorzerlegung eine explizite Menge zulässiger Branching-Levels samt Admissibility-Oberfläche; `SelectedBranchingStrong` konsumiert genau diese Witness-Stufe und schließt eine kanonische Minimalwahl, ohne Umgebungsdiagnostik, Sektortrennung oder Branching-Minimalität erneut aus Rohdaten aufzubauen. Im aktiven P16-Pfad wurden keine neuen `noncomputable`-Produktionsdefinitionen, keine klassischen Hilfen und keine pauschalen `@[simp]`-Flächen eingeführt; modulnahe Notation exponiert `BranchingWitness` und `SelectedBranching` parameter-sichtbar. Folge: P17 muss UV-/Branching-Selektion als Consumer dieser bereits kanonisierten Branching-Oberfläche formulieren und darf weder Aktivitätszähler noch rohen `SectorSplit`-Zugriff als Ersatz für die Witness-/Auswahlstufe verwenden.

- Mit nun grünem P0–P17-Stand sind `Sectors/UVSpectralSelector` und `Sectors/BranchingSelector` als deterministische/proof-carrying Selektorstufe über der bereits kanonisierten Witness-/Auswahloberfläche geschlossen: `UVSpectralSelectorStrong` markiert IR/UV-Anteile ausschließlich als derived-only Provenienzfläche über `SelectedBranchingStrong`, und `BranchingSelectorStrong` bündelt genau diese UV-Oberfläche mit der kanonischen Branching-Auswahl, ohne rohe Aktivitätszähler oder direkten `SectorSplitStrong`-Zugriff als Ersatzpfad zu verwenden. Die im P17-Pfad auftretenden abhängigen Level-/Carrier-Beziehungen werden explizit theorematisch transportiert und nicht als definitorische Gleichheit kaschiert. Im aktiven P17-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die lesbare Notationsoberfläche für `UVSpectralSelector` und `BranchingSelector` ist freigegeben und importseitig eingebunden. Folge: P18 kann `ParameterClosure` nun als harten Consumer von

`DtNStabilized` plus kanonisierter Branching-/UV-Selektoroberfläche formulieren.

- Mit nun grünem P0–P18-Stand ist `Closure/ParameterClosure` als harte Closure-Parameterstufe über `DtNStabilizedStrong` und `BranchingSelectorStrong` geschlossen: `ParameterClosureStrong` bündelt die bereits geschlossene Shift-/Stabilisierungsoberfläche des Operatorpfads mit der kanonisierten Branching-/UV-Seite, ohne einen Rückweg auf rohe `SectorSplit`- oder Witness-Daten als semantischen Ersatzpfad aufzubauen. Die aus P13 kommende computable Provenienz der Stabilisierung bleibt explizit sichtbar, und im aktiven P18-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die lesbare Notationsoberfläche für `ParameterClosure` ist lokal freigegeben und importseitig eingebunden. Folge: P19 kann `RegularizationClosure` nun direkt auf dieser Closure-Provenienz formulieren.

- Mit nun grünem P0–P19-Stand ist `Closure/RegularizationClosure` als eigentliche Regularisierungs-/Schließungsstufe über `ParameterClosureStrong` geschlossen: `RegularizationClosureStrong` konsumiert ausschließlich die bereits ausgewiesene computable Closure-/Shift-Provenienz, führt keine zweite nichtkonstruktive Regularisierungslogik ein und öffnet keinen Rückweg auf rohe `DtNStabilized`-, `BranchingSelector`- oder ältere OQS-Regularisierungspfade. Im aktiven P19-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die lesbare Notationsoberfläche für `RegularizationClosure` ist lokal freigegeben und importseitig eingebunden. Folge: P20 kann `RegionNet` weiter unabhängig als reinen Consumer von `SectorSplitStrong` formulieren, während spätere Closure-/Handoff-Consumer die abgeschlossene Regularisierungsseite nur noch über `RegularizationClosureStrong` und nicht rückwärts über `ParameterClosureStrong` oder ältere Altpfade konsumieren dürfen.

- Mit nun grünem P0–P20-Stand ist `Network/RegionNet` als reine Netzstufe über `SectorSplitStrong` geschlossen: `RegionNetStrong` konsumiert ausschließlich die bereits kanonische Sektorzerlegung, rekonstruiert weder Closure- noch Branching- oder ältere OQS/Bridge-Oberflächen als semantische Ersatzpfade und bündelt helle, Interface- und dunkle Regions-/Boundary-/Carrier-Daten nur als derived-only Projektionen derselben Netzquelle. Im aktiven P20-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die modulnahe Notationsoberfläche für `RegionNet` und `RegionKind` ist freigegeben und importseitig eingebunden. Folge: P21 kann `GeneralizedDtN` nun parallel zum Netzpfad als gekoppelten Consumer von `DtNStrong` und `SectorSplitStrong` formulieren, ohne `RegionNetStrong` oder die Closure-Seite als verdeckten Operatorersatz zu benutzen.

- Mit nun grünem P0–P21-Stand ist `Coupling/GeneralizedDtN` als gekoppelte Mehrsektor-DtN-Stufe parallel zum Netzpfad geschlossen: `GeneralizedDtNStrong` konsumiert ausschließlich `DtNStrong` und `SectorSplitStrong`, bindet deren Konsistenz explizit theorematisch und stellt sektorielle Regions-/Boundary-/Interior-/Carrier-Projektionen sowie blockweise Randoperator-Oberflächen als derived-only Kopplungsdaten bereit, ohne `RegionNetStrong`, `RegularizationClosureStrong` oder ältere Bridge-/OQS-Pfade als verdeckte Operatorersatzflächen einzublenden. Im aktiven P21-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die modulnahe Notationsoberfläche für `GeneralizedDtN` und `CoupledSectorKind` ist freigegeben und importseitig eingebunden. Folge: P22 kann `MultiSchur` nun als nächsten Coupling-Consumer formulieren; falls dafür lokal zusätzliche, fachlich zu `GeneralizedDtN` gehörige Strukturen in P21 nachträglich benötigt werden, dürfen diese dort ergänzt werden, sofern die Fachbereiche strikt getrennt in ihren jeweiligen Modulen verbleiben und `MultiSchur` keine semantische Fachlogik zurück in `GeneralizedDtN` verlagert.

- Mit nun grünem P0–P22-Stand ist `Coupling/MultiSchur` als Schur-Kompositionsstufe über der gekoppelten Mehrsektoroberfläche geschlossen: `MultiSchurStrong` konsumiert ausschließlich `GeneralizedDtNStrong`, trägt Restriktion, Gluing und die reduzierte Schur-Komposition explizit auf der aus P21 gewonnenen Kopplungsoberfläche und verlagert keine fachliche `MultiSchur`-Logik zurück in `GeneralizedDtN`. Rückwirkend in P21 ergänzte Hilfsstrukturen bleiben auf fachlich zu `GeneralizedDtN` gehörige Restriktions-/Projektionsdaten beschränkt; im aktiven P22-Pfad wurden

keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt. Folge: P23 kann `InfiniteCarrier` nun als nächsten späten Network-Consumer formulieren, ohne die Coupling-Komposition erneut in Handoff- oder Netzmodule zu verschieben.

- Rauchtestbefund nach grünem P27: Die aktuelle Kopplungsseite trägt zwar den rohen binären `DtN`-Pfad und die Stabilisierung legal über `SectorSplitStrong`/`BranchPatchStrong` mit, die operative P21/P22-Oberfläche erzwingt diese Dualität aber noch nicht scharf genug. Folge: Die Architektur ist nicht durch einen Seitenpfad von `MultiSchur` nach `RegularizationClosureStrong` zu fixen; stattdessen ist `GeneralizedDtNStrong` selbst so fortzuschreiben, dass es raw/stabilized als explizit getrennte, theorematisch gebundene Operatoroberflächen derselben legalen P21-Provenienz trägt, während `MultiSchurStrong` ausschließlich den stabilisierten Restriktionspfad konsumiert und der rohe Pfad als Audit-/Diagnostikfläche sichtbar bleibt.

- Mit nun grünem P0–P23-Stand sind `Network/InfiniteCarrier` und `Network/SectorChannels` als späte Träger-/Kanalstufe über dem bereits geschlossenen A-Kern geschlossen: `InfiniteCarrierStrong` konsumiert ausschließlich die bereits geschlossene Netzstufe und bündelt keine neue ontische Unendlichkeitsquelle, sondern nur bereits vorhandene Foundation-/ToC-Unendlichkeitsdaten über eine explizite truncation-stabile Bindung an die aktive endliche Schicht; dadurch bleiben stabile endliche Ausschnitte und gerichtete Übergänge für spätere Rekonstruktions- und Limesanalysen exportierbar, ohne bereits AQFT-Quasilokalität oder Typ-III-Aussagen vorwegzunehmen. `SectorChannelsStrong` bleibt flankierender Supportconsumer der bereits geschlossenen Kopplungsoberfläche und rekonstruiert weder neue Netz-, Closure- noch Handoff-Struktur. Im aktiven P23-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die lesbare Notationsoberfläche für `InfiniteCarrier` und `SectorChannels` ist freigegeben und importseitig eingebunden. Folge: P24 kann `SectorSysEnv`, `RelativeEntropyFlow` sowie den Geometrieblock `LCPMeasure` → `Foliation` → `SpacetimePaths` als flankierende Supportschicht formulieren, ohne den Hauptpfad rückwärts zu beeinflussen.

- Mit nun grünem P0–P24-Stand sind `Network/SectorSysEnv`, `Network/RelativeEntropyFlow` sowie der Geometrieblock `Geometry/LCPMeasure`, `Geometry/Foliation` und `Geometry/SpacetimePaths` als flankierende Supportschicht über dem bereits geschlossenen A-Kern geschlossen: `SectorSysEnvStrong` konsumiert ausschließlich `SectorChannelsStrong` und führt die System/Interface/Environment-Seite nur als derived-only Reindexierung der bereits geschlossenen Netz-/Kanaloberfläche; `RelativeEntropyFlow` konsumiert ausschließlich `SectorSysEnvStrong` und bleibt entropische Flussbuchhaltung ohne neue physikalische Grundaxiomatik. Im Geometrieblock liefert `LCPMeasure` adressnahe Präfix-/Nähedaten, `Foliation` fixiert darüber nur die kanonische Blätterung nach Level, und erst `SpacetimePaths` konsumiert diese Blätterung für Weltlinien-, Tube- und Kausalbegriffe; Σ_k bleibt dabei Carrier-Schicht und nicht Slice-Ersatz. Im aktiven P24-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die lesbare Notationsoberfläche für `SectorSysEnv`, `RelativeEntropyFlow`, `LCPMeasure`, `Foliation` und `SpacetimePaths` ist freigegeben und importseitig eingebunden. Folge: P25 kann `SectorExport` nun als Exportconsumer der reifen A-Daten formulieren, ohne flankierende Netzwerk- oder Geometriebegriffe rückwärts in den Kernpfad zu verschieben.

- Mit nun grünem P0–P25-Stand sind `Handoff/SectorExport` und `Handoff/Step1StrongCore` als erste Handoff-Stufe über dem bereits geschlossenen A-Kern geschlossen: `SectorExportStrong` projiziert die reifen A-Daten aus dem bereits geschlossenen Netz-, Closure- und Coupling-Pfad auf eine zielneutrale Exportoberfläche, ohne zur Sammelstelle roher Einzelinputs zu werden oder flankierende Netzwerk- bzw. Geometriebegriffe rückwärts in den Kernpfad zu verschieben. Die Provenienz der exportierten Daten bleibt explizit sichtbar; insbesondere wird die Regularisierungsseite nur über `RegularizationClosureStrong` konsumiert und nicht semantisch rückwärts über `ParameterClosureStrong` oder ältere Regularisierungspfade ersetzt. `Step1StrongCore` fasst darauf aufbauend nur bereits geschlossene Teilstränge kompakt zusammen, ohne selbst

neue Fachlogik, neue Rohdatenpfade oder einen verdeckten Vorgriff auf den gerichteten A→B-Handoff einzuführen. Im aktiven P25-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt; die lesbare Notationsoberfläche für `SectorExport` und `Step1StrongCore` ist freigegeben und importseitig eingebunden. Folge: P26 kann nun `Handoff/Step1MathData`, `Handoff/ABHandoffStrong` und `Handoff/PillarAStep1Closed` als proof-carrying Abschluss der A-Abgangswelle formulieren; dabei bleibt `SectorExportStrong` die zielneutrale öffentliche Exportfläche von A, während `ABHandoffStrong` nur der gerichtete Adapter `A_to_B` ist und die zwischenzeitliche dünne Step-1-Gate-Stufe ausschließlich von `PillarAStep1Closed` gespeist werden darf.

- Mit nun grünem P0–P26-Stand sind `Handoff/Step1MathData`, `Handoff/ABHandoffStrong` und `Handoff/PillarAStep1Closed` sowie die zwischenzeitliche dünne Step-1-Gate-Stufe als proof-carrying Abschluss der A-Abgangswelle über dem bereits geschlossenen A-Kern geschlossen: `Step1MathDataStrong` bündelt das volle A-Mathematikpaket nur über `Step1StrongCore`, `ABHandoffStrong` bleibt ausschließlich der gerichtete Adapter `A_to_B`, `PillarAStep1Closed` ist das einzig legitime Endobjekt des geared Pfads, und die dünne Gate-Stufe konsumiert ausschließlich dieses Abschlussobjekt. Im aktiven P26-Pfad wurden keine neuen `@[simp]`-Flächen, keine neuen `noncomputable`-Produktionsdefinitionen und keine klassischen Hilfen eingeführt. Die globale Aliasoberfläche in `PillarA/Notation` für `Step1MathData`, `ABHandoffStrong` und `PillarAStep1Closed` ist freigegeben; die modulnahe Spiegelung in `Handoff/Notation` sowie ihre konsequente Konsumption wurde im Zuge des P27-Cutovers nachgezogen. Folge: der Altpfad-Cutover konnte abgeschlossen werden, indem alte Bridge-, Core-, OQS- und AQFT-Pfade aus dem aktiven Build-Pfad nach `legacy_sources/CNNA_v0.100_unstable` verschoben wurden; dies ist eine Archivierung außerhalb des Build-Pfads und keine fachlich aktive Doppelspur.

- Mit nun grünem P27-Stand ist der Archiv-Cutover abgeschlossen: Insbesondere `DirichletLaplacianBridge`, `StrongLegacyAdapters`, `RegularizationClosure_old` sowie weitere alte Core-/OQS-/AQFT-/Integration-/Bridge-Pfade und außerdem `Gates`, `Meta` und `Seeds` wurden nach `legacy_sources/CNNA_v0.100_` verschoben. Sie bleiben dort als Regressionsorakel erhalten, dürfen aber nicht mehr in den aktiven Build oder als semantisch gleichwertige Ersatzpfade zurückwirken.

- Der aktive Root-Build ist damit bewusst minimal: `CNNA/BuildAll.lean` importiert nur noch `CNNA/Notation` und `CNNA/PillarA/BuildAll`, während `CNNA/PillarB` bis `CNNA/PillarE` im aktiven Baum nur noch `BuildAll.lean`-Stubs tragen. Aktiv enthalten ist damit nur noch der neue derived-only Pillar-A-Pfad ausgehend vom ToC.

- Der aktive P27-Baum ist damit zwar architektonisch bereinigt und fast durchgängig generisch über `Cell` und `[SubstrateClass Cell]` parametrisiert, besitzt aber noch keine explizite Referenzfamilie eines konkreten IDEAL-ToC, die zusammen mit `Addressing` und `IdealAddressEquiv` als tatsächlich konsumierte Wurzelinstanziierung im aktiven Pfad läuft. Folge: der geared Pfad ist als Architektur geschlossen, aber noch kein produktiv nutzbarer Generator.

- Der Operatorpfad trägt im aktuellen Stand noch freie Kernzeugnisse wie `weight` in `Finite/DirichletLaplacian` `InteriorInverse` in `DtN/DtN`, `epsilon` in `DtN/DtNStabilized` und `interfaceInverse` in `Coupling/MultiSch` Folge: Schritt 2 darf diese Witnesses nicht bloß durch einen Smoke-Test von außen speisen, sondern muss sie soweit legal möglich in die Provenienzketten des aktiven Pfads zurückziehen.

- Die Roadmap fordert zugleich, dass das ToC nicht auf ein einziges privilegiertes Substrat festgelegt wird und dass Closure-Parameter wie `b`, `Lmax`, β und sekundäre Regularisierer nicht als primitive Restparameter im Kern verbleiben. Folge: das Strengthening benötigt neben einer robusten Referenzfamilie auch eine explizite Variations- und Vergleichsoberfläche über gültige Substratfamilien.

4 Vollständigkeitsprüfung: reine Pillar-A-Core-Module

Fundamente: `SubstrateClass`, `ExecComplex`, `ExecComplexLemmas` (optional), `MatrixNorms`, `Interfaces`, `Determinism`, `HermitianStructure`, `FiniteHilbert`, `MatrixAlgebra`.
ToC-Idealkern und Präsentationsbrücke: `ToC/Contract`, `ToC/Addressing`, `ToC/IdealAddressEquiv`, `ToC/Concrete`.
Endliche Vorstufe: `Finite/CutSpec`, `RegionCore`, `BoundaryPorts`, `Approximant`, `Selection`, `DirichletLaplacian`.
Sektor-/Branching-Vorstufe: `BranchPatch`, `ComplementSectorFamily`, `SectorSplit`, `BranchingWitness`, `SelectedBranching`, `UVSpectralSelector`, `BranchingSelector`.
OQS-Kern / Closure: `DtN`, `DtNStabilized`, `ParameterClosure`, `RegularizationClosure`, `GeneralizedDtN`, `MultiSchur`.
Netz / Geometrie / Handoff in Pillar A: `RegionNet`, `InfiniteCarrier`, `SectorChannels`, `SectorSysEnv`, `RelativeEntropyFlow`, `Geometry/LCPMeasure`, `Geometry/Foliation`, `Geometry/SpacetimePath` `SectorExport`, `Step1StrongCore`, `Step1MathData`, `ABHandoffStrong`, `PillarAStep1Closed`.
Alt-/Übergangsmodule: `DirichletLaplacianBridge`, `StrongLegacyAdapters`, `RegularizationClosure_old`

5 Pillar-Ordnungsregel für Refactor und Strengthening

- Neue A-Kernmodule gehen nach `CNNA/PillarA/...`; spätere neue B–E-Module würden nach `CNNA/PillarB/...` bis `CNNA/PillarE/...` gehen. Im aktuellen grünen P27-Stand bleiben `CNNA/PillarB` bis `CNNA/PillarE` im aktiven Baum jedoch bewusst auf `BuildAll.lean`-Stubs reduziert.
- Die Altordner `Core/OQS/AQFT/...` sowie weitere ersetzte Altpfade bleiben nur als Cutover-Quelle stehen und werden nach erfolgreicher Umstellung aus dem aktiven Build-Pfad nach `legacy_sources/CNNA_v0.100_unstable` verschoben; im grünen P27-Stand gilt dies zusätzlich für `Integration`, `Gates`, `Meta` und `Seeds`.
- Ab Start des Refactors wird keine neue Fachlogik mehr in alte Top-Level-Ordner geschrieben, sofern bereits ein neuer Pillarpfad existiert.
- Keine leeren Scaffold-Landschaften vorab; neue Unterordner und Module werden erst dann angelegt, wenn ihre Phase beginnt.
- Im ToC-Pfad gilt strikt: `Contract` modelliert den abstrakten idealen unendlichen Träger, `Addressing` seine adressierte Präsentation, `IdealAddressEquiv` deren strukturtreue Äquivalenz, und `Concrete` nur reale Approximantenfamilien dieses Trägers. Endliche Proxies dürfen weder den Idealvertrag noch die Adressierungs-Äquivalenz ersetzen.
- Kanonische Beschriftungen werden entlang dieses Pfads stufenweise geschlossen: P4 liefert die kanonische Zelladressierung des aktiven Ideals; P5 macht diese Präsentation transportinvariant; erst die späteren Consumer von P6 bis mindestens P17 dürfen daraus kanonische Identifikationen für Schnitte, Approximanten, Rand-/Komplementdaten, Environment-Splits sowie IR/UV-Anteile ableiten.
- Im Strengthening-Schritt dürfen neue A-Kernmodule nur dann ergänzt werden, wenn sie eine reale Generatorlücke schließen: computable Referenzfamilie, zweite legale Substratfamilie, explizite WeightPolicy-Achse, kanonischer Operatorseed, kontrollierte Solver-/Regularisierungsoberfläche, Generator- bzw. Variationsauswertung. Neue Module sind nicht als zweiter semantischer Pfad neben dem aktiven Baum zulässig; der einzige zulässige explizite Parallelfall ist der auditierbare algebraische Dualstrang aus computablem `ExecComplex`-Produktivpfad und getrenntem **C**-Spiegelstrang, dessen operative Konsumrichtung strikt festliegt.
- Referenzfall und Substratfamilie dürfen insbesondere keinen parallelen Downstream-Strang erzeugen: ab dem ersten legalen `ToCStrong`- bzw. Approximantenkern müssen beide dieselben späteren

Consumer bedienen. Smoke- und Analyseläufe gelten nur dann als architektonisch zulässig, wenn sie genau diesen Produktionspfad benutzen.

Zielstruktur für Pillar A

- `Foundation/`: `SubstrateClass`, `ExecComplex`, `ExecComplexLemmas` (optional), `MatrixNorms`, `Interfaces`, `Determinism`, `HermitianStructure`, `FiniteHilbert`, `MatrixAlgebra`
  mit Fokus auf idealen geschichteten Träger, Wurzelstruktur auf Ebene 0, lokaler Endlichkeit, Eltern-/Kinder-Kohärenz, nichtterminaler Verzweigung, dynamischen Branching-Daten, `refine/coarsen`-Vorstufe, expliziten unendlichen Fäden sowie einer nun explizit öffentlich computablen algebraischen A-Wurzel über `ExecComplex`; `MatrixNorms` bleibt der erste Dualstrang-Consumer dieser Wurzel, und die präzise leere-Ursprung-Semantik wird daran anschließend in `ToC/Contract` fixiert
- `ToC/`: `Contract`, `Addressing`, `IdealAddressEquiv`, `Concrete`
  wobei `Contract` den abstrakten idealen ToC-Fixpunkt trägt, `Addressing` zunächst nur die kanonische Zelladressierung des aktiven Ideals aufbaut, `IdealAddressEquiv` diese Präsentation strikt strukturtreu und transportinvariant zum abstrakten Vertrag zurückbindet, und `Concrete` erst danach reale Approximantenfamilien sowie weitere kanonische Identifikationen als Consumer dieser Äquivalenz formuliert
- `Finite/`: `CutSpec`, `RegionCore`, `BoundaryPorts`, `Approximant`, `Selection`, `DirichletLaplacian`
- `DtN/`: `DtN`, `DtNStabilized`
- `Sectors/`: `BranchPatch`, `ComplementSectorFamily`, `SectorSplit`, `BranchingWitness`, `SelectedBranching`, `UVSpectralSelector`, `BranchingSelector`
- `Closure/`: `ParameterClosure`, `RegularizationClosure`
- `Network/`: `RegionNet`, `InfiniteCarrier`, `SectorChannels`, `SectorSysEnv`, `RelativeEntropyFlow`
- `Geometry/`: `LCPMeasure`, `Foliation`, `SpacetimePaths`
- `Coupling/`: `GeneralizedDtN`, `MultiSchur`
- `Handoff/`: `SectorExport`, `Step1StrongCore`, `Step1MathData`, `ABHandoffStrong`, `PillarAStep1Closed`
- Aktiver Root-Build nach P27: `CNNA/BuildAll` → `CNNA/Notation` → `CNNA/PillarA/BuildAll`; `Gates`, `Meta` und `Seeds` liegen nur noch in der Archivspur, `PillarB`–`PillarE` bleiben im aktiven Baum auf `BuildAll.lean`-Stubs reduziert

6 Legacy-Politik

6.1 Direkt oder fast direkt migrierbar

- `AQFT/StarAlgebra.lean`, `AQFT/CStarNorm.lean`, `AQFT/State.lean`, `AQFT/RelativeEntropy.lean`
- `AQFT/LocalNet.lean`, `AQFT/TimeReversal.lean`
- `AQFT/QuasiLocal/Carrier.lean`, `Structure.lean`, `StarAlgebra.lean`, `Completion.lean`

6.2 Nur nach Adapter-Stufe migrierbar

- `AQFT/StateNet.lean`
- `AQFT/GNS.lean`, `AQFT/KMS.lean`
- `AQFT/QuasiLocal/StateLift.lean`

6.3 Nicht migrieren, nur als Vorlage/Orakel verwenden

- `AQFT/Derived/BoundaryMatrixLocalNet.lean`
- `AQFT/Derived/BoundaryMatrixGNS.lean`
- `AQFT/Derived/BoundaryMatrixSplitProperty.lean`
- `AQFT/Derived/BoundaryMatrixQuasiLocalClosure.lean`
- `AQFT/TimeOrientation.lean`

6.4 Für P1–P6 besonders relevante Legacy-Anker

- `legacy_sources/REALOQS/PillarA/Core/SubstrateClass.lean`
- `legacy_sources/REALOQS/PillarA/Core/MatrixNorms.lean`
- `legacy_sources/REALOQS/PillarA/Core/Determinism.lean`
- `legacy_sources/REALOQS/PillarA/Ideal/Substrate.lean`
- `legacy_sources/REALOQS/PillarA/Ideal/Adapter/SubstrateInstance.lean`
- `legacy_sources/REALOQS/PillarA/Ideal/TreeOfCliques/Vertex.lean`, `Boundary.lean`, `CoarsenBoundar`
- `legacy_sources/REALOQS/PillarA/Ideal/Foliation.lean`
- `legacy_sources/REALOQS/PillarA/Ideal/LCPMeasure.lean`
- `legacy_sources/REALOQS/PillarA/Ideal/SpacetimePaths.lean`
- `legacy_sources/REALOQS/PillarA/Core/InfiniteCarrier.lean`

Diese Dateien sind keine Blaupause zum unkritischen Portieren. Sie markieren aber den mathematisch relevanten Suchraum: adressbasierte unendliche Schichtung, Ebenenmengen, Kinder-/Randabbildungen, Koarsenierung, Präfix-/Frontier-Ideen, eine kanonische Blätterung nach Level sowie ein expliziter Begriff eines unendlichen Trägers. Für `Addressing`, `IdealAddressEquiv` und spätere `CutSpec`-/Geometriemodule sind sie Inspirationsquelle, nicht automatische Pfadvorgabe. Besonders für den Geometrieblock ist festzuhalten: Das Legacy-`Foliation`-Modul trägt nur einen kleinen Präfix-/Längen-Kern, während die eigentliche Foliationssemantik dort faktisch in `SpacetimePaths` über $\tau := \text{length}$, Schichten Σ_n und Adaptedness-Begriffe liegt. Im Refactor wird dies *nicht* roh portiert: `Geometry/Foliation` fixiert nur die kanonische Blätterung der bereits vorhandenen Levelstruktur, während Weltlinien-, Tube- und Kausalbegriffe Consumer in `Geometry/SpacetimePaths` bleiben.

6.5 P27-Archivpolitik für CNNA-Altpfade

Nach grünem P27 werden die alten CNNA-Altpfade *nicht* physisch gelöscht, sondern gesammelt nach `legacy_sources/CNNA_v0.100_unstable` verschoben. Dazu zählen insbesondere alte Bridge-, Core-, OQS-, AQFT- und Integration-Pfade des bisherigen Schritt-1-Strangs sowie die zwischenzeitlichen Top-Level-Ordner `Gates`, `Meta` und `Seeds`, soweit sie durch den neuen Pillar-A-Handoff-Pfad ersetzt wurden. Die Archivspur bleibt damit als Regressionsorakel und historische Referenz erhalten, liegt aber außerhalb des aktiven Build-Pfads und darf weder neue Fachlogik aufnehmen noch als aktiver Import- oder Ersatzpfad zurückwirken. Der aktive Root-Build ist damit auf `CNNA/BuildAll` → `CNNA/Notation` → `CNNA/PillarA/BuildAll` reduziert; `CNNA/PillarB` bis `CNNA/PillarE` tragen im aktiven Baum nur noch `BuildAll.lean`-Stubs.

7 Operative Refactor- und Strengthening-Arbeitsordnung

Der Hauptstrang bis `PillarAStep1Closed` ist zweistufig.
Ontische Zusatzregel für P1–P6: Der ToC-Kern wird zuerst als ideales unendliches Substrat modelliert. Darauf folgen eine adressierte Präsentation dieses Ideals und eine strikt strukturtreue Äquivalenz zwischen abstrakter und adressierter Präsentation. Erst danach wird die reale endliche ToC-Schicht als gerichtete Approximantenfamilie dieses Idealobjekts formuliert. Weder $\mathrm{REAL}_\infty = \mathrm{IDEAL}_\infty$ noch $\mathrm{IDEAL}_\infty^{\text{abstrakt}} = \mathrm{IDEAL}_\infty^{\text{adressiert}}$ werden im Plan als rohe Typgleichheit verstanden; Ziel sind theorematische Rekonstruktions- bzw. Äquivalenzaussagen über stabile endliche Ausschnitte, gerichtete Übergänge und bidirektionale Transfers. Die in P1 bereits eingeführten unendlichen Fäden sind dabei Foundation-Vorstruktur; `InfiniteCarrier` in P23 ist *nicht* der erste Ort der Unendlichkeit, sondern ein später, netz- und handofffähiger Trägerrahmen über bereits geschlossenem A-Kern.

Stufe A: Wurzel → früher A-Kernschluss (bis `RegularizationClosure`)

```
SubstrateClass -> MatrixNorms -> Interfaces -> Determinism -> ToC/Contract
-> ToC/Addressing -> ToC/IdealAddressEquiv -> ToC/Concrete -> Finite/CutSpec
-> RegionCore -> BoundaryPorts -> Approximant -> DirichletLaplacian -> DtN
-> DtNStabilized -> BranchPatch -> ComplementSectorFamily -> SectorSplit ->
BranchingWitness -> SelectedBranching + UVSpectralSelector -> BranchingSelector
-> ParameterClosure -> RegularizationClosure
```

Stufe B: vom geschlossenen frühen A-Kern zur Abgangswelle

```
SectorSplit -> RegionNet
```

parallel dazu

```
DtN + SectorSplit -> GeneralizedDtN -> MultiSchur
```

und danach

```
InfiniteCarrier -> SectorExport -> Step1StrongCore -> Step1MathData -> ABHandoffStrong
-> PillarAStep1Closed
```

Harte Klassifikation

- **Unverzichtbarer Hauptstrang:** Nr. 1–12, 14–29, 36–40. Modul 29 (`InfiniteCarrier`) ist notwendig, aber spät und nur aus bereits geschlossenen A-Strukturen abgeleitet.
- **Bedingter Support:** Nr. 13 (`Selection`) bleibt auch nach grünem P10 ein Supportmodul: es ist nun zwar derived-only geschlossen, wird aber nur dann aktiv als Dependency des Hauptstrangs genutzt, wenn die Operatorstufe eine echte Teilstruktur-Auswahl benötigt.
- **Flankierender Support:** Nr. 30–35 (`SectorChannels`, `SectorSysEnv`, `RelativeEntropyFlow`, `Geometry/LCPMeasure`, `Geometry/Foliation`, `Geometry/SpacetimePaths`) sind Nebenstrangmodule; sie dürfen den Hauptpfad nicht rückwärts beeinflussen.
- **Konstruktionsregel:** `SectorExport`, `Step1MathData` und `ABHandoffStrong` dürfen keine Sammelstellen roher Einzelinputs werden; sie sind als Komposition bereits geschlossener Teilstränge zu formulieren.
- **Branching-Consumer-Regel ab P16:** `BranchingSelector` ist als Consumer von `SelectedBranching` plus `UVSpectralSelector` zu formulieren; rohes `SectorSplitStrong` bleibt im Folgepfad für seine eigenen direkten Consumer wie `RegionNet` oder `GeneralizedDtN` reserviert und darf nicht als Ersatzoberfläche für die bereits geschlossene Witness-/Auswahlstufe dienen.
- **Coupling-Dualpfad-Regel ab P21/P22:** Die Kopplungsseite darf die in P13 explizit geschlossene Trennung von rohem und stabilisiertem Randoperator nicht wieder implizit zusammenziehen. `GeneralizedDtNStrong` bleibt ausschließlich Consumer von `DtNStrong` und `SectorSplitStrong`,

muss aber daraus zwei explizite, theorematisch gebundene Operatoroberflächen — raw und stabilized — derselben legalen P21-Provenienz tragen. `MultiSchurStrong` bleibt reiner P22-Consumer von `GeneralizedDtNStrong` und arbeitet operativ nur auf dem stabilisierten Restriktionspfad; der rohe Pfad bleibt als Audit-/Diagnostikoberfläche sichtbar. Ein direkter Closure-Seitenpfad von `GeneralizedDtN`/`MultiSchur` nach `RegularizationClosureStrong` ist kein zulässiger Architekturfix.

8 Menschenlesbare Notation als Parallelstrang

- Für grün geschlossene aktive Frühmodule des Pfads P1–P4 ist grundsätzlich eine kleine modulnahe `Notation.lean`-Datei vorzusehen, sofern das Modul direkte Kernobjekte des aktiven Pfads exponiert; spätere Module folgen derselben Regel bei Bedarf.
- Verwendet wird nur menschenlesbare, möglichst in Mathematik oder Physik etablierte Notation; keine kosmetische Fantasienotation und keine privaten Kürzel für nicht etablierte Begriffe.
- Die Lean-Objekte selbst sollen bereits menschenlesbare, fachlich sprechende Namen tragen; Notation ergänzt einen guten Kernnamen, sie kompensiert keine kryptischen Objektbezeichner. Für Reviewer soll der aktive Pfad auch ohne tiefes Nachschlagen in Lean/Mathlib in seiner fachlichen Grundstruktur lesbar bleiben.
- Die Notation wird erst freigegeben, wenn der zugrunde liegende Objektkern bereits derived-only geschlossen ist.
- Notation ist **scoped** und rein lesbare Makro-Ebene; sie darf keine semantische Verschiebung des Kerns erzeugen. Bevorzugt werden lesbare Aliasnamen und nur dort Symbolnotation, wo diese fachlich wirklich etabliert und für Reviewer unmittelbar interpretierbar ist.
- Für die frühe Schichtungssemantik wird die Carrier-Ebene strikt von verpackten Slice-Objekten getrennt: Σ_k bezeichnet eine Level-k-Schicht als Trägermenge (im Root-Fall insbesondere Σ_0 als Wurzelschicht), während Slice-Objekte wie `rootSlice` oder allgemein `LayerSlice` ausgeschrieben bleiben. Dieselbe Symbolik darf nicht zugleich für Carrier und Slice-Datum verwendet werden; insbesondere wird Ω_0 nicht doppelt für Wurzelschicht und Wurzelslice belegt.
- Für familienparametrische Objekte müssen Notationen ihre Parameter sichtbar oder explizit bindbar lassen; insbesondere darf eine Notation keine Typklasseninstanz schon beim Import erzwingen.
- Nullstellige Notation für noch offene familienparametrische Objekte ist zu vermeiden; insbesondere dürfen Objekte mit Parametern wie `Cell` oder Instanzen wie `[SubstrateClass Cell]` nicht als scheinbar geschlossene Konstanten maskiert werden.
- Notation darf keine abhängigen Level-/Carrier-Transporte oder Äquivalenztransfers verdecken; wo solche Transporte fachlich tragen, bleiben sie explizit im Kern oder werden als eigene Rekonstruktions- bzw. Transfersätze formuliert.
- Jede Refactor-Phase prüft zusätzlich, ob für die neu geschlossenen Objekte die passende lesbare Notation lokal ergänzt werden muss.
- Mit grünem P16 sind in `Sectors/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `BranchingWitness` und `SelectedBranching` freigegeben; spätere Selector-/Closure-Module dürfen diese nutzen, müssen aber Familienparameter sichtbar lassen und dürfen weder zulässige Levelmengen noch Minimalitätsbeweise hinter nullstelliger Symbolik verstecken.
- Mit grünem P17 sind in `Sectors/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `UVSpectralSelector` und `BranchingSelector` freigegeben; diese dürfen die in P17 fachlich tra-

genden abhängigen Level-/Carrier-Transporte nicht verdecken und keine scheinbare definitorische Gleichheit zwischen Branching- und UV-Seite suggerieren.

- Mit grünem P18 sind in `Closure/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `ParameterClosure` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen die aus `DtNStabilized` kommende computable Shift-/Stabilisierungsprovenienz nicht verdecken und keine scheinbar geschlossene Regularisierungsstufe suggerieren, die erst P19 liefert.

- Mit grünem P19 sind in `Closure/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `RegularizationClosure` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen keine zweite oder semantisch stärkere Regularisierungslogik suggerieren und die aus `ParameterClosure`/`DtNStabilized` übernommene computable Provenienz weder verdecken noch als definitorisch neue Schicht maskieren.

- Mit grünem P20 sind in `Network/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `RegionNet` und `RegionKind` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen weder eine zusätzliche Netzsemantik über `SectorSplitStrong` hinaus suggerieren noch die Provenienz der hellen/Interface/dunklen Projektionen als definitorisch unabhängige Schicht maskieren.

- Mit grünem P21 sind in `Coupling/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `GeneralizedDtN` und `CoupledSectorKind` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen weder eine zusätzliche Netz- oder Closure-Semantik über `DtNStrong` und `SectorSplitStrong` hinaus suggerieren noch die blockweise Kopplungsoberfläche als definitorisch unabhängige zweite Operatorstufe maskieren. Falls P22 rückwirkend zusätzliche P21-Strukturen benötigt, dürfen die zugehörigen Aliasoberflächen lokal ergänzt werden, solange dadurch keine `MultiSchur`-Fachlogik in die Notations- oder Kopplungsdatei von `GeneralizedDtN` vorverlagert wird.

- Mit grünem P22 sind in `Coupling/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `MultiSchur` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen weder eine zusätzliche Netz-, Closure- noch Handoff-Semantik über `GeneralizedDtNStrong` hinaus suggerieren noch die in P22 explizit getragenen Restriktions-, Gluing- und Schur-Kompositionsdaten als definitorisch unabhängige dritte Operatorstufe maskieren. Die Aliasoberfläche darf insbesondere nicht den Eindruck erwecken, als sei die reduzierte Schur-Seite bereits ein selbständiger Ersatz für spätere Export- oder Netzconsumer.

- Mit grünem P24 sind in `Network/Notation`, `Geometry/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `SectorSysEnv`, `RelativeEntropyFlow`, `LCPMeasure`, `Foliation` und `SpacetimePaths` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen weder die System/Interface/Environment-Reindexierung noch die Präfix-/Level-Transporte des Geometrieblocks als definitorisch unabhängige neue Kernschichten maskieren und müssen die Trennung Σ_k als Carrier-Schicht gegenüber ausgeschriebenen Slice-Objekten ausdrücklich wahren; τ- bzw. Σ-Notation darf insbesondere keine zusätzliche Dynamik oder Kausalstruktur bereits in `Foliation` suggerieren, die erst `SpacetimePaths` als Consumer tragen darf.

- Mit grünem P25 sind in `Handoff/Notation` bzw. `PillarA/Notation` lesbare Aliasoberflächen für `SectorExport` und `Step1StrongCore` freigegeben; diese müssen die Familienparameter sichtbar lassen, dürfen weder die zielneutrale Exportfläche von A mit dem gerichteten Handoff verwechseln noch tiefe Provenienz- oder Transportketten als scheinbar definitorisch neue Fachschicht maskieren. Die Aliasoberfläche darf insbesondere nicht den Eindruck erwecken, als enthalte `Step1StrongCore` bereits das volle A→B-Handoff oder als sei `SectorExport` selbst schon ein gerichteter Adapter.

- Mit grünem P26 sind in `PillarA/Notation` lesbare Aliasoberflächen für `Step1MathData`, `ABHandoffStrong` und `PillarAStep1Closed` global freigegeben; diese müssen die drei Stufen weiterhin klar trennen: `Step1MathData` als volles proof-carrying A-Mathematikpaket, `ABHandoffStrong` nur als

gerichteter Adapter `A_to_B`, und `PillarAStep1Closed` als einzig legitimes Endobjekt des geared Pfads bzw. der zwischenzeitlichen dünnen Gate-Stufe. Keine Aliasoberfläche darf diese Trennung verwischen oder der Gate-Stufe eigene Produktionslogik suggerieren. Die modulnahe Spiegelung in `Handoff/Notation` für `ABHandoffStrong` und `PillarAStep1Closed` sowie ihre konsequente Konsumption im Handoff-Pfad wurde mit dem grünen P27-Archiv-Cutover nachgezogen.

- Notationsdateien dürfen keine Importzyklen erzeugen, keine spätere Fachlogik vorwegnehmen und keine theorematisch tragenden Parametrisierungen durch Symbolik überspielen. Für P4 bleibt die sichtbare Review-Oberfläche bewusst auf `root`, `parent`, `children`, `ancestor`, `Prefix`, `cellOf`, `addressOf` konzentriert.

9 Vorabprüfung vor Modulschluss

- alle direkt importierten tragenden Module sind selbst bereits derived-only,
- die Imports sind nicht-zirkulär,
- die Imports sind nicht vorausnehmend,
- der neue Builder verwendet nur die zugelassenen starken Vorgänger und keine losen Rohdaten,
- ein expliziter Abtriebssatz zum nächsten zulässigen Consumer ist formuliert,
- im ToC-Pfad ist geprüft, dass der Idealvertrag nicht durch endliche `Concrete`-Daten vorweggenommen wird und dass `Concrete` den Idealträger nur approximiert, aber nicht definiert,
- vor `ToC/Concrete` ist geklärt, ob der aktive Idealpfad über eine explizite Adressierungsstruktur oder über eine Klasse adressierbarer Ideale läuft; `ToC/Addressing` und `ToC/IdealAddressEquiv` müssen dabei die Transportdaten explizit ausweisen,
- `ToC/IdealAddressEquiv` erhält mindestens Level, Wurzel, Eltern-/Kinder-Struktur, Nichtterminalität, Fäden sowie `refine/coarsen`; spätere `CutSpec`-, Frontier-, Komplement- und Boundary-Daten werden nur auf dieser Basis übertragen,
- ab P5 ist geprüft, dass jede spätere Objektklasse des "Box-Inhalts" – insbesondere Kanten, Schnitte, Approximanten, Rand-/Komplementdaten, Environment-Splits sowie IR/UV-Anteile – nur über diese invarianten Transportdaten identifiziert wird und keine präsentationslokale Ersatz-Kanonizität einführt,
- ab P7 ist für jeden Consumer des endlichen Box-Inhalts der Primärträger explizit festgelegt – abstrakt-konkrete P6-Seite, adressierte Präsentation oder äquivalent rückgeführte Seite – und adressseitige Beschreibungen werden nur als transportierte Präsentationen derselben kanonischen Daten zugelassen; bei abhängigen bzw. indizierten Finite-Objekten sind diese Transporte im Kern explizit durch Rekonstruktions- oder Cast-Sätze auszuweisen und nicht als definitorische Gleichheit zu kaschieren,
- ab P7 ist zusätzlich zu prüfen, dass kanonische adressierte Schnitte des aktiven Finite-Pfads – insbesondere Zylinder- und Teilbaum-Schnitte – über entscheidbare Präfixdaten computable formuliert sind; vermeidbare `noncomputable`-Restquellen im aktiven Box-Pfad sind vor P9 zu beseitigen und nicht als Dauerzustand in spätere Consumer mitzuschleppen,
- ab P9 sind Wrapper- und Projektionsoberflächen des aktiven Box-Pfads grundsätzlich nicht pauschal mit `@[simp]` zu markieren; automatische Aufklappung ist nur für klar stabile Projektionen zulässig und im Zweifel zu vermeiden,
- ab P13 ist zusätzlich zu prüfen, dass Regularisierungs- und Shift-Stufen des aktiven Operatorpfads ihre computable Provenienz explizit ausweisen; analytische `noncomputable`-Hilfsgrößen

dürfen separat bestehen bleiben, aber nicht stillschweigend als konsumierte Kernoberfläche von `DtNStabilized` oder seinen Notationsaliasen weitergereicht werden,

- ab P16 ist zusätzlich zu prüfen, dass spätere Selector-, Closure- und Export-Consumer den Branching-Zustand nur über die proof-carrying Oberfläche von `BranchingWitness/SelectedBranching` konsumieren; zulässige Branching-Levels, Aktivitätsdiagnostik und Minimalitätsaussagen dürfen nicht erneut aus rohen Sektorcarriern, bloßen Aktivitätszählern oder ad-hoc-Minima auf `SectorSplitStrong` rekonstruiert werden,
- ab P17 ist zusätzlich zu prüfen, dass abhängige Level-/Carrier-Beziehungen zwischen Branching- und UV-Selektoroberfläche explizit theorematisch transportiert und nicht als definitorische Gleichheit kaschiert werden; Notation und Aliasnamen dürfen diese Transporte nicht unsichtbar machen,
- ab P18 ist zusätzlich zu prüfen, dass Closure-Consumer den Branching-/UV-Zustand nur über `BranchingSelectorStrong` und die Stabilisierung nur über die explizit ausgewiesene computable Oberfläche von `DtNStabilizedStrong/ParameterClosureStrong` konsumieren; delegierte Projektionen auf Witness-, Split- oder Rohoperator-Ebene dürfen nicht als semantischer Ersatzpfad für die Closure-Stufe dienen,
- ab P19 ist zusätzlich zu prüfen, dass spätere Closure-, Export- und Handoff-Consumer die abgeschlossene Regularisierungsseite ausschließlich über `RegularizationClosureStrong` konsumieren; Rückgriffe auf `ParameterClosureStrong`, rohe `DtNStabilized`-Projektionen oder ältere OQS-Regularisierungspfade sind dabei nicht als semantisch gleichwertige Ersatzpfade zulässig,
- ab P20 ist zusätzlich zu prüfen, dass direkte Netz-Consumer des Sektorpfads ihre operative Oberfläche ausschließlich aus `SectorSplitStrong` gewinnen; Closure-, Branching-, DtN- oder ältere Bridge-/OQS-Pfade dürfen dort nicht als semantisch gleichwertige Ersatzoberflächen eingeblendet werden, und sektorielle Regions-/Boundary-/Carrier-Projektionen bleiben derived-only Projektionen derselben Netzstufe,
- ab P21 ist zusätzlich zu prüfen, dass Coupling-Consumer ihre operative Oberfläche ausschließlich aus `DtNStrong` und `SectorSplitStrong` gewinnen; `RegionNetStrong`, `RegularizationClosureStrong` oder ältere Bridge-/OQS-Pfade dürfen dort nicht als semantisch gleichwertige Operatorersatzflächen eingeblendet werden. `GeneralizedDtNStrong` hat dabei raw/stabilized als explizit getrennte, theorematisch gebundene Operatoroberflächen derselben legalen P21-Provenienz sichtbar zu tragen; ein semantisch mehrdeutiger Einheitsoperator ist als operative Kernoberfläche zu vermeiden. Falls P22 rückwirkend zusätzliche P21-Strukturen benötigt, sind diese lokal in `GeneralizedDtN` nachzutragen, ohne die Fachgrenze zum `MultiSchur`-Modul zu verwischen,
- im aktiven geared Hauptpfad sind neue `noncomputable`-Produktionsdefinitionen grundsätzlich unzulässig, sofern nicht lokal theorematisch ausgewiesen ist, dass sie mathematisch unvermeidbar sind, keine computable Ersatzformulierung existiert und die konsumierte Consumer- bzw. Exportoberfläche dennoch vollständig computable bleibt,
- ab P23 ist geprüft, dass `InfiniteCarrier` keine neue ontische Unendlichkeitsquelle einführt, sondern nur bereits geschlossene Foundation-/ToC-Unendlichkeitsdaten netzfähig bündelt.
- ab P24 ist zusätzlich zu prüfen, dass flankierende Netzwerk- und Geometrieconsumer ihre operative Oberfläche ausschließlich aus den bereits geschlossenen Supportvorgängern gewinnen: `SectorSysEnvStrong` nur aus `SectorChannelsStrong`, `RelativeEntropyFlowStrong` nur aus `SectorSysEnvStrong`, und der Geometrieblock strikt entlang `LCPMeasure` → `Foliation` → `SpacetimePaths`; dabei darf Σ_k nur Level-Carrier bezeichnen, während Weltlinien-, Tube- und Kausalbegriffe nicht vorzeitig in `Foliation` oder in Notationsaliasen vorweggenommen werden.
- ab P25 ist zusätzlich zu prüfen, dass Export- und Handoff-Consumer ihre operative Oberfläche ausschließlich aus den bereits geschlossenen reifen A-Strängen gewinnen: `SectorExportStrong` nur als

Projektion reifer Netz-, Closure- und Coupling-Daten, `Step1StrongCore` nur als kompakte Zusammenfassung bereits geschlossener Teilstränge. Weder flankierende Netzwerk-/Geometrieconsumer noch rohe Vorgänger, Ersatzpfade oder ältere Bridge-/OQS-Handoffs dürfen dort als semantisch gleichwertige Exportoberflächen eingeblendet werden. Insbesondere ist P25 kein Reparaturort für P21/P22: Die Explizitheit raw/stabilized muss bereits in der Coupling-Schicht geschlossen sein und darf im Export nur noch theorematisch sichtbar projiziert, nicht semantisch nachgebessert werden.

- ab P26 ist zusätzlich zu prüfen, dass `Step1MathData` und `ABHandoffStrong` keine Sammelstellen roher Einzelinputs werden: `Step1MathData` bündelt nur das volle proof-carrying A-Mathematikpaket über dem bereits geschlossenen Handoff-Kern, `ABHandoffStrong` bleibt ausschließlich der gerichtete Adapter `A_to_B`, und `PillarAStep1Closed` ist das einzig legitime Endobjekt des geared Pfads. Die zwischenzeitliche dünne Step-1-Gate-Stufe konsumiert nur `PillarAStep1Closed` und enthält keine eigene Produktionslogik mehr.
- ab P27 ist zusätzlich zu prüfen, dass nach erfolgreichem Konsumentenwechsel alte Bridge-, Core-, OQS-, AQFT-, Integration-, Gates-, Meta- und Seeds-Pfade nicht mehr im aktiven Build-Pfad verbleiben, sondern gesammelt nach `legacy_sources/CNNA_v0.100_unstable` verschoben werden; diese Archivspur bleibt reines Regressionsorakel außerhalb des Build-Pfads und darf weder neue Fachlogik aufnehmen noch als aktiver Import- oder Ersatzpfad zurückwirken. Zugleich bleibt aktiv nur der neue derived-only Pillar-A-Pfad, während `PillarB` bis `PillarE` höchstens `BuildAll.lean`-Stubs tragen.
- ab S10 ist zusätzlich zu prüfen, dass Update-, Diagnose-, Symmetrie-, thermische und dynamische Seeds ausschließlich über bereits geschlossene A-Kern- bzw. algebraische Seed-Vorgänger konsumieren; sie dürfen weder Closure- noch Coupling-Logik als Rückkanal benutzen noch B/C/D-Fachsemantik vorwegnehmen.

Folge: Modulschluss ist niemals nur lokal. Er ist stets ein Urteil über die tragfähige Importkette bis zur Wurzel.

10 Exakte Programmierreihenfolge der reinen Pillar-A-Core-Module

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
1	`Foundation/SubstrateClass`	idealer geschichteter Basiskontrakt: Wurzel auf Ebene 0, lokale Endlichkeit, Eltern-/Kinder-Kohärenz, nichtterminale Verzweigung, dynamische Branching-Daten, `refine/coarsen`-Vorstufe, explizite unendliche Fäden; liefert die einzige primitive diskrete Wurzel des aktiven Pfads, aus der spätere Trägerdaten derived-only hervorgehen; die präzise leere-Ursprung-Semantik bleibt als Anschluss an `ToC/Contract` markiert	Wurzel für `MatrixNorms`, `Interfaces` und den idealen ToC-Vertrag

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
2	`Foundation/MatrixNorms`	nur fundamentale Matrix-/Normfakten; mindestens Frobenius-Kern mit Nichtnegativität, Nulltest und Positivität bei $\neq 0$; Regularisierungsbausteine für den aktiven Pfad sind so auszuweisen, dass ihre konsumierte Shift-Stufe explizit computable bleibt und analytische `noncomputable`-Hilfsgrößen nicht stillschweigend in `DtNStabilized` weitergereicht werden	versorgt `DirichletLaplacian`, `DtN`, `DtNStabilized`
3	`Foundation/Interfaces`	kleinste gemeinsame Typschnittstellen über dem Foundation-Kern; exponiert nur mathematisch tragende Schnittstellen und keine bequemen Sammel-APIs oder Labelhalden	versorgt `ToC/Contract`, `RegionCore`, `BoundaryPorts`
4	`Foundation/Determinism`	deterministische Auswahl- und Eindeutigkeitssätze aus Policy-/Key-Daten auf Basis des Foundation-Kerns; liefert kanonische Auswahl als Satz und keine bloßen Hilfskonstruktionen oder Export-Hüllen-Ersatzlogik	versorgt `ToC/Concrete`, `BranchingSelector`, `Step1MathData`
5	`ToC/Contract`	reiner abstrakter Idealvertrag des unendlichen ToC: idealer Träger, kohärente Ketten auf Basis der Foundation-Fäden, präzisierte leere-Ursprung-/Fixpunktsemantik, keine terminale Stufe, keine `Concrete`- oder `Dirichlet`-Importe; trägt noch keine adressierte oder endliche Präsentation	treibt `ToC/Addressing` und später `ToC/Concrete`
6	`ToC/Addressing`	adressierte Präsentation des aktiven Ideals mit Präfix-/Ancestor-Struktur und eindeutiger Zellzuordnung; schließt zunächst die kanonische Zelladressierung, aber noch nicht den gesamten Box-Inhalt; sichtbare Review-Oberfläche bleibt auf `root`, `parent`, `children`, `ancestor`, `Prefix`, `cellOf`, `addressOf` konzentriert; keine ontische Umdefinition des abstrakten Vertrags	versorgt `ToC/IdealAddressEquiv`, `Finite/CutSpec`

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
7	`ToC/IdealAddressEquiv`	strikt strukturtreue Äquivalenz zwischen abstraktem und adressiertem IDEAL$_\infty$: `encode`, `decode`, Links-/Rechtsinverse und Erhaltung von Level, Wurzel, Eltern/Kindern, Fäden, `refine/coarsen`; ab hier müssen spätere Objektklassen über P5-invariante Transportdaten identifiziert werden	kanonische Transferbrücke für `ToC/Concrete`, `Finite/CutSpec` und spätere Transfersätze
8	`ToC/Concrete`	REAL-Approximantenfamilie des idealen ToC mit gerichteten Übergängen in `L_max`; konkrete endliche Realisierung darf lokal zunächst statisches `b` verwenden, bleibt aber Consumer der Äquivalenzschicht und der Determinismus-Sätze; liefert den ersten `ToCStrong`-Kern mit policy-keyed Trunkierungen, auf dem spätere Box-Objekte aufsetzen, statt präsentationslokale Ersatzrepräsentationen als Primärträger einzuführen	versorgt `Finite/CutSpec`, `RegionCore`, `Approximant`, `UVSpectralSelector`
9	`Finite/CutSpec`	allgemeine endliche Schnittspezifikation über P6-Approximanten und adressierte bzw. äquivalent rückgeführte ToC-Daten; umfasst mindestens Trunk-, Fenster-, Zylinder-/Teilbaum- und adaptive Frontier-Schnitte; legt die kanonischen Schnitte des Box-Inhalts fest und weist explizit aus, auf welcher Seite der Schnitt primär lebt, während adressseitige Beschreibungen nur transportierte Präsentationen desselben Schnitts bleiben; adressierte Zylinder-/Teilbaum-Schnitte sind im aktiven Pfad über entscheidbare Präfixdaten computable zu formulieren und nicht bloß als tolerierte `noncomputable`-Restfälle stehen zu lassen	treibt `RegionCore`, `BoundaryPorts`, `Approximant` und spätere Environment-Splits
10	`Finite/RegionCore`	kleinster Regionskern aus `CutSpec` und ToC-Daten; Primärträger von Region und Innenbereich ist explizit festgelegt und wird nicht stillschweigend mit seiner adressierten Präsentation identifiziert	treibt `BoundaryPorts`, `Approximant`, `BranchPatch`

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
11	`Finite/BoundaryPorts`	kanonische Randport-Extraktion und damit erste kanonische Randmarkierung der endlichen Box; Randdaten werden aus dem in `CutSpec/RegionCore` fixierten Primärträger gewonnen und nicht über adresslokale Ersatz-Kanonizität eingeführt; die Oberfläche soll auch den für spätere Consumer benötigten Innenbereich so exponieren, dass kein Seitenimport zurück nach `RegionCore` nötig wird	liefert Randdaten für `SectorSplit`
12	`Finite/Approximant`	endliche ToC-Abschneidung mit Herkunftssätzen aus `CutSpec`; ist Consumer des in P6 bereitgestellten Approximantenkerns und des in `CutSpec/RegionCore/BoundaryPorts` kanonisch fixierten endlichen Box-Inhalts; fixiert diesen zum gegebenen Cutoff, ohne einen zweiten `Concrete`-Kern oder ad-hoc Determinismuspfad aufzubauen	treibt `Selection`, `DirichletLaplacian`, `BranchPatch`, Selektoren
13	`Finite/Selection`	ausgewählte endliche Teilstruktur	bleibt Supportmodul, kein Alternativpfad
14	`Finite/DirichletLaplacian`	Symmetrie-, Nichtnegativitäts- und Blockzerlegungssätze über dem bereits kanonisch fixierten endlichen Box-Inhalt; der operative Indexträger ist ein gebundener endlicher Box-Träger über allen Levels $\leq$ Cutoff, nicht bloß eine Topschicht und nicht ein ungebundener globaler Träger; Rand-/Innen-Partition und die zugehörigen Blockmatrizen sind als derived-only Folge des Approximantenkerns explizit auszuweisen; analytische Energieterme sind so zu formulieren, dass keine vermeidbare `noncomputable`-Definition in den aktiven Pfad zurückkehrt	einziger direkter Motor für `DtN`; `Selection` darf nur bei echter Teilstruktur-Auswahl konsumiert werden und bleibt sonst unbenutzt
15	`DtN/DtN`	binärer DtN-Kern aus `DirichletLaplacianStrong`	treibt `DtNStabilized` und später `GeneralizedDtN`

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
16	`DtN/DtNStabilized`	expliziter Shift-/Regularisierungssatz; trennt rohen Randoperator, symmetrisierte Hilfsstufe und primäre stabilisierte Operatoroberfläche explizit und konsumiert nur eine computable Shift-Stufe aus `MatrixNorms`, sodass keine vermeidbare `noncomputable`-Restquelle in den aktiven Pfad zurückkehrt	versorgt `ParameterClosure`, `RegularizationClosure`, `SectorExport`
17	`Sectors/BranchPatch`	rein ontische lokale Patch-Struktur	Vorstufe für `SectorSplit`
18	`Sectors/ComplementSectorFamily`	Komplementfamilie aus ToC-/Regionsdaten; erste kanonische Umgebung/Komplementseite der Box	zweiter Input für `SectorSplit`
19	`Sectors/SectorSplit`	helle/interface/dunkle Zerlegung, geometrisch	treibt `RegionNet`, `BranchingWitness`, `GeneralizedDtN`, `SectorChannels`, `SectorSysEnv`
20	`Sectors/BranchingWitness`	proof-carrying Witness-Stufe für zulässige Branching-Level über `SectorSplitStrong`; rekonstruiert eine kanonische Kandidatenmenge samt Admissibility, ohne Umgebungsdiagnostik oder Sektortrennung neu aufzubauen	treibt `SelectedBranching`; spätere Selectoren dürfen Witness-Daten nicht durch Rohzugriffe auf `SectorSplitStrong` ersetzen
21	`Sectors/SelectedBranching`	kanonische Auswahl eines zulässigen Branching-Levels mit Minimalitätsbeweis über der Witness-Stufe	primärer Branching-Input für `BranchingSelector`
22	`Sectors/UVSpectralSelector`	vor-operatorischer UV-/Spektralselektor; markiert den UV-tail nur als derived-only Folge der kanonischen Vorstufen	liefert UV-Selektorinformation an `BranchingSelector` und als Provenienzoberfläche weiter an `ParameterClosure`
23	`Sectors/BranchingSelector`	deterministische/proof-carrying Zusammenführung von `SelectedBranching` und `UVSpectralSelector`	letzter Selektor vor der Closure
24	`Closure/ParameterClosure`	Schließungsparameter aus `DtNStabilized` plus kanonisierter Branching-/UV-Selektoroberfläche; die computable Provenienz der Auswahl bleibt sichtbar	treibt `RegularizationClosure` und `SectorExport`
25	`Closure/RegularizationClosure`	eigentliche Regularisierungs-/Schließungsstufe	letzter strenger Abschluss des frühen Operatorpfads
26	`Network/RegionNet`	Netzstruktur ausschließlich aus `SectorSplitStrong`	Input für `InfiniteCarrier` und `SectorExport`

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
27	`Coupling/GeneralizedDtN`	Mehrsektor-DtN-Kopplung aus `DtNStrong` + `SectorSplitStrong`; trägt raw/stabilized als explizit getrennte, theorematisch gebundene Operatoroberflächen derselben legalen P21-Provenienz sowie die dazugehörigen Restriktions-/Projektionsdaten, ohne Closure- oder Netzersatzpfade einzublenden	treibt `MultiSchur` und `SectorExport`
28	`Coupling/MultiSchur`	Schur-Komposition, Restriktion und Gluing ausschließlich als Consumer der in P21 bereits explizit geschlossenen Kopplungsoberfläche; operativer Schur-Pfad läuft auf dem stabilisierten Restriktionsblock, während der rohe Pfad nur Audit-/Diagnostikoberfläche bleibt	liefert Kopplungsdaten an den Handoff-Strang
29	`Network/InfiniteCarrier`	netz- und handofffähiger unendlicher Trägerrahmen, noch ohne AQFT-Quasilokalität; *nicht* erste Einführung von Unendlichkeit, sondern spätere Bündelung bereits vorhandener Foundation-/ToC-Unendlichkeitsdaten	spät ableiten; versorgt `SectorExport`, `Step1MathData`
30	`Network/SectorChannels`	kanalartige Sektorverbindungen	Supportconsumer von `SectorSplit` bzw. `GeneralizedDtN`
31	`Network/SectorSysEnv`	System/Environment-Zerlegung über dem Sektornetz	versorgt `RelativeEntropyFlow`
32	`Network/RelativeEntropyFlow`	entropische Flussbuchhaltung	späteres Supportmodul; kein Rückeinfluss auf den Kernpfad
33	`Geometry/LCPMeasure`	adressnahe Geometriestufe für längste gemeinsame Präfixe bzw. zellnahe Nähedaten	flankierender Support für spätere Geometrie- und Analysepfade
34	`Geometry/Foliation`	kanonische Blätterung des adressierten bzw. über P5 transportierten Trägers nach Level; definiert gebündelten Träger, Foliationsfunktion τ, Blätter Σ_n als Level-n-Trägermengen sowie elementare Sätze über Blattzugehörigkeit und Gleichheit bei gleicher Schicht unter Präfixrelation; Slice-Objekte bleiben davon begrifflich getrennt und werden weiterhin ausgeschrieben geführt; führt keine zusätzliche Dynamik oder Kausalstruktur ein	geometrische Vorstufe für `SpacetimePaths` und flankierender Support für spätere Analysepfade
35	`Geometry/SpacetimePaths`	pfadartige Weltlinien-/Raumzeitdarstellung über dem adressierten Kern	Support für spätere Pfad- und Flussanalysen

Nr.	Modul	Muss lokal erfüllen	Muss wie verzahnen
36	`Handoff/SectorExport`	vollständiger A-Export aus reifen A-Daten	projiziert nur reife A-Daten und nimmt keine B-Struktur vorweg
37	`Handoff/Step1StrongCore`	kompakte starke Zusammenfassung des Kernstrangs	treibt `Step1MathData`, `ABHandoffStrong`
38	`Handoff/Step1MathData`	volles proof-carrying Mathematikpaket	bündelt die gesamte A-interne Kraftübertragung
39	`Handoff/ABHandoffStrong`	verbreiterter, reifer A→B-Handoff	eigentliche Abgangswelle
40	`Handoff/PillarAStep1Closed`	Endobjekt / Endtheorem des geared Pfads	einzig legitimer Input der zwischenzeitlichen dünnen Gate-Stufe und aktiver Root-Endpunkt nach P27

11 Refactor-Phasen P0–P27 (Schritt 1)

Phase	Module / Bereich	Pflichtresultat
P0	alle in Abschnitt 9 geführten reinen Pillar-A-Core-Module; zusätzlich `DirichletLaplacianBridge`, `StrongLegacyAdapters`, `RegularizationClosure_old`; Inventar, Zielpfade, Freeze; Altmodule markieren	vollständige Klassifikation aller reinen Pillar-A-Core-Module inklusive `ToC/Addressing`, `ToC/IdealAddressEquiv`, `Finite/CutSpec` und der Geometriemodule; alter Pfad nur noch Regressionsorakel
P1	`Foundation/SubstrateClass`, `Foundation/MatrixNorms`; begleitend `Foundation/Notation`	fundamentale Wurzel steht unter `PillarA/Foundation`; `SubstrateClass` trägt bereits den idealen geschichteten Kern des $\texttt{IDEAL}_\infty$-ToC und markiert die einzige primitive diskrete Wurzel des aktiven Pfads; `MatrixNorms` ist kein Stub mehr und hält die vom aktiven Pfad konsumierte Regularisierungsstufe explizit computable, statt analytische `noncomputable`-Hilfsgrößen ungeprüft nach vorne weiterzureichen
P2	`Foundation/Interfaces`, `Foundation/Determinism`; begleitend `Foundation/Notation`	Typschnittstellen und Determinismus-Sätze stehen getrennt *über dem bereits geschlossenen Foundation-Kern*; `Interfaces` exponiert nur mathematisch tragende Schnittstellen, `Determinism` liefert kanonische Auswahl als Satz; keine Bool-Audit-Ersatzschlüsse, keine Labelhalden, Policy-/Key-Bindung ist explizit
P3	`ToC/Contract`; begleitend `ToC/Notation`	reiner abstrakter Idealvertrag des ToC ohne spätere Importe; der ideale unendliche Träger wird *auf Basis des P1-Substratkerns* und der typisierten P2-Schnittstellen präzisiert, trägt die Fixpunktrolle und schließt insbesondere die leere-Ursprung-/Grenzsemantik; P3 trägt den abstrakten Gegenstand, aber noch keine adressierte oder endliche Präsentation; der grüne P3-Stand bleibt bestehen und wird nicht zurückgenommen

Phase	Module / Bereich	Pflichtresultat
P4	`ToC/Addressing`; begleitend `ToC/Notation`	adressierte Präsentation des aktiven $\texttt{IDEAL}_\infty$ steht; Zellen sind adressartig eindeutig und damit kanonisch referenzierbar, ohne den abstrakten Vertrag zu ersetzen; die sichtbare P4-Oberfläche konzentriert sich auf `root`, `parent`, `children`, `ancestor`, `Prefix`, `cellOf`, `addressOf`; die Kanonizität des übrigen Box-Inhalts wird hier noch nicht behauptet
P5	`ToC/IdealAddressEquiv`	strikt strukturtreue Äquivalenz zwischen abstraktem und adressiertem $\texttt{IDEAL}_\infty$ steht; `encode`/`decode`, Inversen und Kernerhaltungssätze sind explizit formuliert; jede spätere Objektklasse darf nur über diese P5-invarianten Transportdaten identifiziert werden
P6	`ToC/Concrete`; begleitend `ToC/Notation`	REAL-Approximantenfamilie des idealen ToC ohne Rückimport aus `Finite/DtN`; `Concrete` bleibt familienparametrischer Consumer der Äquivalenzschicht und der Determinismus-Sätze; abhängige Level-/Carrier-Daten und Äquivalenztransfers werden nicht durch Notation verdeckt; erster `ToCStrong` und Startpunkt späterer kanonischer Identifikationen des endlichen Box-Inhalts
P7	`Finite/CutSpec`	allgemeine endliche Schnittspezifikation steht; mindestens Trunk-, Fenster-, Zylinder-/Teilbaum- und adaptive Frontier-Schnitte sind formulierbar; die Schnitte werden als Consumer der P6-Approximantenfamilie formuliert, der Primärträger der Schnittdaten ist explizit ausgewiesen, und adressseitige Beschreibungen bleiben transportierte Präsentationen derselben Schnitte; notwendige Rückführungen entlang der P5/P6-Äquivalenz sind auch für indizierte Schnittobjekte explizit theorematisch auszuweisen; adressierte Zylinder-/Teilbaum-Schnitte sind dabei im aktiven Pfad über entscheidbare Präfixdaten computable formuliert und nicht bloß tolerierte `noncomputable`-Restfälle; seitlicher Rest/Environment-Split ist architektonisch vorbereitet und als kanonischer Schnitt des Box-Inhalts formulierbar

Phase	Module / Bereich	Pflichtresultat
P8	`Finite/RegionCore`, `Finite/BoundaryPorts`	endlicher Regionskern und Randport-Extraktion stehen getrennt und ohne Vorgriff auf den Operatorpfad; Importe laufen nur über den neuen Foundation-/ToC-Pfad; Primärträger von Region, Innenbereich und Rand ist explizit festgelegt, und adressseitige Beschreibungen werden nur als durch P5/P6 abgesicherte Präsentationen derselben Box-Daten zugelassen; `BoundaryPorts` exponiert dabei auch den für spätere Consumer benötigten Innenbereich, sodass der grüne P8-Stand als geschlossene konsumierbare Finite-Kernstufe für P9 bestehen bleibt
P9	`Finite/Approximant`	`ApproximantStrong` steht als kanonische endliche Abschneidung aus `CutSpec` über einem bereits in P6 bereitgestellten Approximantenkern; `ApproximantStrong` ist damit Consumer und kein zweiter `Concrete`-Kern; es konsumiert den in `CutSpec/RegionCore/BoundaryPorts` kanonisch fixierten endlichen Box-Inhalt, statt Region oder Rand erneut aufzubauen; dabei soll insbesondere kein Seitenimport zurück nach `RegionCore` nötig werden, sofern die konsumierbare `BoundaryPorts`-Oberfläche den benötigten Innenbereich bereits trägt; kein Rückgriff auf Legacy-Interfaces oder ad-hoc Determinismus-Hilfen; der endliche Box-Inhalt ist damit für den gewählten Cutoff kanonisch fixiert; die modulare Oberfläche bleibt nach Bereinigung anfänglicher Wrapper-Automatik explizit, und pauschale `@[simp]`-Aufklappung ist im aktiven Pfad zu vermeiden
P10	`Finite/Selection`	`SelectionStrong` steht als explizit isoliertes Supportmodul über `ApproximantStrong`; kein Alternativpfad, keine Rekonstruktion von `Concrete/RegionCore/BoundaryPorts`, keine neuen `noncomputable`-Restquellen, klassischen Hilfen oder pauschalen `@[simp]`-Rückfälle; modulnahe Notation bleibt parameter-sichtbar und rein lesbar
P11	`Finite/DirichletLaplacian`; begleitend modulnahe Notation	`DirichletLaplacianStrong` steht und konsumiert nur die minimalen Foundation-Matrixfakten sowie den bereits kanonisch fixierten endlichen Box-Inhalt; der zugrunde liegende Operatorträger wird dabei als gebundener endlicher Box-Träger über allen Levels $\leq$ Cutoff explizit fixiert, sodass Rand-/Innen-Blockzerlegung und der Übergang zu P12 nicht auf eine einzelne Topschicht oder auf einen ungebundenen Globalträger ausweichen; `Selection` darf nur bei echter Teilstruktur-Auswahl zusätzlich konsumiert werden und bleibt sonst unbenutzt; einzige direkte Motorstufe für den binären DtN-Kern

Phase	Module / Bereich	Pflichtresultat
P12	`DtN/DtN`; begleitend modulnahe Notation	binärer DtN-Kern ist isoliert; `DtN` importiert kein `RegionNet`, führt mit `InteriorInverse`, `boundaryOperator`, `interiorSolve` und expliziten Abtriebssätzen eine proof-carrying Oberfläche mit und bleibt ohne neue pauschale `@[simp]`-Flächen oder neue `noncomputable`-Restquellen strikt Consumer von `DirichletLaplacianStrong`
P13	`DtN/DtNStabilized`	`DtNStabilizedStrong` schließt den frühen Operatorpfad nach vorn, konsumiert dabei nur den neuen P12-Pfad unter `CNNA/PillarA/DtN/DtN.lean`, trennt rohen Randoperator und symmetrisierte Hilfsstufe explizit und hält die primäre Stabilisierung über eine explizit computable Shift-Stufe; ein Rückfall auf `CNNA/OQS/DtN.lean` oder eine neue `noncomputable`-Kontamination des aktiven Pfads ist ausgeschlossen
P14	`Sectors/BranchPatch`, `Sectors/ComplementSectorFamily`	ontische Patch- und Komplementvorstufe steht separat und ohne spätere AQFT-/Handoff-Last; die Umgebung/Komplementseite wird erstmals kanonisch familienfähig, und die Datenoberfläche unterscheidet explizit zwischen root-zentriertem Ausschnitt ohne äußere Umgebung und Fenstern mit echter Umgebung/Komplementseite, statt beide Fälle erst implizit in späteren Consumern zu vermischen
P15	`Sectors/SectorSplit`; begleitend modulnahe Notation	`SectorSplitStrong` steht als geometrisch-ontische Sektorzerlegung der nun kanonisch markierten Box-Daten; die Zerlegung macht insbesondere die Präsenz bzw. Abwesenheit einer äußeren Umgebung theorematisch unterscheidbar, ohne daraus schon im Kernpfad eine physikalische Symmetrie- oder Brechungsdiagnostik zu machen; dabei ist explizit darauf zu achten, dass `SectorSplit` die in `ComplementSectorFamilyStrong` bereits gewonnene kanonische Außenseite konsumiert und keine konkurrierende Umgebungsdiagnostik erneut aufbaut, auch dann nicht, wenn die Regions-/Boundary-Sicht dieser Außenseite derived-only über `RegionCore.ofCutSpec` und `BoundaryPorts.ofRegion` rekonstruiert wird

Phase	Module / Bereich	Pflichtresultat
P16	`Sectors/BranchingWitness`, `Sectors/SelectedBranching`; begleitend modulnahe Notation	Witness- und Auswahlstufe stehen proof-carrying und ohne zusätzliche Seiteneingänge; `BranchingWitnessStrong` rekonstruiert eine kanonische Menge zulässiger Branching-Levels, `SelectedBranchingStrong` schließt darüber eine Minimalwahl, und beide konsumieren `SectorSplitStrong` direkt, ohne Umgebungsdiagnostik, Sektortrennung oder Branching-Minimalität aus Rohdaten erneut aufzubauen; modulnahe Notation bleibt parameter-sichtbar und rein lesbar
P17	`Sectors/UVSpectralSelector`, `Sectors/BranchingSelector`	Selektorstufe ist deterministisch/proof-carrying und treibt exklusiv die Closure-Parameter; `BranchingSelector` konsumiert die bereits kanonisierte Branching-Oberfläche aus `SelectedBranchingStrong` zusammen mit `UVSpectralSelector` und ersetzt diese nicht durch rohe Aktivitätszähler oder direkten `SectorSplitStrong`-Zugriff; IR/UV-Anteile bleiben auch hier nur abgeleitete kanonische Markierungen, und der UV-tail wird als cutoff-abhängige, aus den kanonischen Vorstufen hergeleitete Provenienzoberfläche exportfähig gemacht, ohne bereits einen Analyse- oder Dynamiksatz zu behaupten; abhängige Level-/Carrier-Beziehungen zwischen Branching- und UV-Seite sind explizit theorematisch zu transportieren und nicht als definitorische Gleichheit zu kaschieren
P18	`Closure/ParameterClosure`	`ParameterClosureStrong` ist harter Consumer von `DtNStabilized` plus kanonisierter Branching-/UV-Selektoroberfläche; die aus P13 kommende computable Provenienz der Shift- und Stabilisierungsdaten darf dabei nicht verlorengehen, und analytische Hilfsgrößen sind nur zulässig, wenn die exportierten Closure-Parameter weiterhin als operative Oberfläche computable bleiben
P19	`Closure/RegularizationClosure`	`RegularizationClosureStrong` schließt die frühe Minimalstrecke des A-Kerns und führt keine neue nichtkonstruktive Regularisierungsschicht ein, sondern schließt explizit an die bereits ausgewiesene computable Shift-/Closure-Provenienz aus P13/P18 an

Phase	Module / Bereich	Pflichtresultat
P20	`Network/RegionNet`; begleitend modulnahe Notation	`RegionNetStrong` ist reine Netzstufe über `SectorSplitStrong`, bündelt helle/Interface/dunkle Regions-, Boundary- und Carrier-Oberflächen nur als derived-only Projektionen derselben kanonischen Sektorquelle und führt keine Closure-, Branching-, `DtN`- oder Altpfade als semantische Ersatzoberflächen ein; modulnahe Notation für `RegionNet` und `RegionKind` bleibt parameter-sichtbar und rein lesbar
P21	`Coupling/GeneralizedDtN`; begleitend modulnahe Notation	`GeneralizedDtNStrong` koppelt Operatorpfad und Sektorstruktur strikt als Consumer von `DtNStrong` und `SectorSplitStrong`, führt sektorielle Restriktions-/Carrier-/Boundary-/Interior-Projektionen sowie blockweise Randoperator-Oberflächen nur als derived-only Kopplungsdaten derselben Stufe und blendet weder `RegionNetStrong`, noch Closure-, noch ältere Bridge-/OQS-Pfade als Operatorersatzflächen ein; zusätzlich schließt P21 raw/stabilized als explizit getrennte, theorematisch gebundene Operatoroberflächen derselben legalen Coupling-Provenienz und stellt die zugehörigen Restriktions-/Projektionsdaten bereit, ohne `MultiSchur`-Fachlogik vorwegzunehmen; modulnahe Notation für `GeneralizedDtN` und `CoupledSectorKind` bleibt parameter-sichtbar und rein lesbar und darf diese Dualität nicht wieder semantisch verwischen
P22	`Coupling/MultiSchur`; begleitend modulnahe Notation	`MultiSchurStrong` liefert Restriktion, Gluing und reduzierte Schur-Komposition als echten Folgeconsumer von `GeneralizedDtNStrong`; etwaige rückwirkend in P21 ergänzte Hilfsstrukturen verbleiben fachlich im Kopplungsvorgänger, während die Schur-Fachlogik ausschließlich in `MultiSchur` selbst liegt; der operative Interface-/Schur-Pfad arbeitet ausschließlich auf dem in P21 explizit geschlossenen stabilisierten Restriktionsblock, während der rohe Pfad als Audit-/Diagnostikoberfläche sichtbar bleibt; modulnahe Notation für `MultiSchur` bleibt parameter-sichtbar und rein lesbar
P23	`Network/InfiniteCarrier`, `Network/SectorChannels`	späte Träger-/Kanalstufe steht; `InfiniteCarrier` bleibt spät abgeleitet, bündelt nur bereits vorhandene Foundation-/ToC-Unendlichkeitsdaten und blockiert den Frühpfad nicht; die Bündelung ist dabei so zu formulieren, dass gerichtete Übergänge und stabile endliche Ausschnitte für spätere Rekonstruktions- bzw. Limesanalysen exportierbar bleiben, ohne bereits AQFT-Quasilokalität oder Typ-III-Aussagen vorwegzunehmen

Phase	Module / Bereich	Pflichtresultat
P24	`Network/SectorSysEnv`, `Network/RelativeEntropyFlow`, `Geometry/LCPMeasure`, `Geometry/Foliation`, `Geometry/SpacetimePaths`; begleitend modulnahe Notation	spätere Support-Netzwerkschicht und adressnahe Geometriestufe stehen ohne Rückeinfluss auf den Kernpfad; `SectorSysEnvStrong` bleibt derived-only Reindexierung über `SectorChannelsStrong`, `RelativeEntropyFlowStrong` reine entropische Flussbuchhaltung über `SectorSysEnvStrong`; innerhalb des Geometrieblocks wird zuerst `LCPMeasure`, dann `Foliation` als kanonische Level-Blätterung und erst danach `SpacetimePaths` als Consumer dieser Blätterung geschlossen; dabei bezeichnet Σ_k die jeweilige Level-Schicht als Carrier, während Slice-Objekte weiterhin ausgeschrieben bleiben; Weltlinien-, Tube- und Kausalbegriffe gehören nicht in `Foliation`, sondern erst in `SpacetimePaths`; die gesamte Schicht bleibt analytisch bzw. geometrisch flankierend
P25	`Handoff/SectorExport`; begleitend modulnahe Notation; `Handoff/Step1StrongCore`	`SectorExportStrong` projiziert nur reife A-Daten aus dem bereits geschlossenen Netz-, Closure- und Coupling-Pfad auf eine zielneutrale Exportoberfläche und wird nicht zur Sammelstelle roher Inputs; der Export hält die Provenienz seiner operativen Daten explizit sichtbar, konsumiert die Regularisierungsseite nur über `RegularizationClosureStrong` und nicht rückwärts über `ParameterClosureStrong` oder ältere Regularisierungspfade, damit spätere Analysepfade nicht auf informelle Rekonstruktionen angewiesen sind und operative Exportdaten von bloß analytischen Hilfsdaten unterschieden bleiben; zugleich repariert P25 keine P21/P22-Unschärfe mehr nachträglich, sondern projiziert nur die dort bereits explizit geschlossene raw/stabilized-Dualität theorematisch sichtbar nach außen; `Step1StrongCore` fasst darauf aufbauend nur bereits geschlossene Teilstränge kompakt zusammen und nimmt weder einen gerichteten Handoff noch Gate-Logik vorweg

Phase	Module / Bereich	Pflichtresultat
P26	`Handoff/Step1MathData`, `Handoff/ABHandoffStrong`, `Handoff/PillarAStep1Closed`; zwischenzeitliche dünne Gate-Stufe; global freigegebene Aliasoberfläche in `PillarA/Notation`	`Step1MathData` bündelt das volle proof-carrying A-Mathematikpaket über `Step1StrongCore`, `ABHandoffStrong` formuliert darauf ausschließlich den gerichteten Handoff `A_to_B`, und `PillarAStep1Closed` schließt die Abgangswelle als einzig legitimes Endobjekt des geared Pfads; die zwischenzeitliche dünne Gate-Stufe konsumiert ausschließlich `PillarAStep1Closed` und enthält keine eigene Produktionslogik; die globale Aliasoberfläche für diese drei Stufen ist freigegeben, während modulnahe Handoff-Notation und konsequente Konsumption als kleiner Nachtrag vor bzw. mit P27 sauber nachzuziehen sind
P27	abgeschlossener Archiv-Cutover; insbesondere `DirichletLaplacianBridge`, `StrongLegacyAdapters`, `RegularizationClosure_old` sowie weitere alte Core-/OQS-/AQFT-/Integration-/Bridge-Pfade und außerdem `Gates`, `Meta`, `Seeds` nach `legacy_sources/CNNA_v0.100_unstable`; `PillarB`–`PillarE` auf `BuildAll`-Stubs reduziert	alte Pfade sind aus dem aktiven Build-Pfad herausgenommen, bleiben außerhalb des Build-Pfads als Regressionsorakel erhalten und dürfen semantisch nicht mehr als aktive Import- oder Ersatzpfade zurückwirken; aktiv verbleibt nur der neue derived-only Pillar-A-Pfad ausgehend vom ToC

Strengthening-Schritt 2. Der abgeschlossene Refactor hat die Importkette bereinigt, die Altpfade archiviert und den aktiven Baum auf den neuen Pillar-A-Pfad reduziert. Der Code-Audit zeigt aber zugleich: Die Abstraktion über `[SubstrateClass Cell]` ist bereits vorhanden, große Teile der `of*`-Builder-Kette stehen schon, und die eigentlichen Lücken sitzen nicht in einer fehlenden Gesamtarchitektur, sondern in wenigen präzisen Generatorengstellen. Für einen produktiv nutzbaren Generator fehlen deshalb nicht nur Referenzfamilie, Variationsfamilie, `WeightPolicy` und die Schließung der freien Builder-Engstellen `DirichletWeight`/`InteriorInverse`/`interfaceInverse`; es fehlt darüber hinaus eine explizite **algebraische A-Wurzel** der endlichen Operatorseite: Adjungierte/Hermitizität, finite Hilbertraumstruktur, endliche Matrix-$*$-Algebra, Kommutator- und Antikommutatorlogik, spektrale Zerlegung, Zustandsraum sowie thermische und unitäre Seeds müssen — soweit sie direkt aus A-generierten endlichen Carriern und Operatoren folgen — bereits in fachlich getrennten A-Modulen bereitgestellt werden und dürfen später nicht in B/C als verdeckter Nachbau wieder auftauchen. Der Legacy-Bestand von REALOQS zeigt gerade in den AQFT-Dateien `StarAlgebra`, `State`, `GNS`, `KMS`, `LocalNet`, `StateNet` und den Derived-Modulen, dass diese Vorstufen später tatsächlich vorausgesetzt werden; Strengthening muss sie daher A-seitig aus der numerischen Pipeline in eine algebraische Generatoroberfläche überführen, ohne volle AQFT-Semantik vorwegzunehmen. Hinzu kommen ein explizites **Computability-Audit** des Generators, eine harte **Fintype/Decidable-Propagationsregel**, eine formale **Directed-Limit**-Infrastruktur für die Lesart $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$ sowie ausdrücklich eine **A-seitige Diagnosevorbereitung** für Spektraldimension sowie für Symmetrie-, Cutoff- und Limesdiagnostik. Ferner ist eine finite Kanal-/Superoperator-Vorstufe als A-seitiger algebraischer Seed bereitzustellen, soweit sie rein algebraischer Consumer der A-generierten Matrixseite bleibt. Diese Werkzeuge gehören architektonisch in Pillar A, dürfen den Generatorhauptpfad nicht zirkular erweitern und dürfen später weder in B noch in anderen Pfeilern ausgelagert werden. Die folgenden Abschnitte listen daher ausschließlich die für Schritt 2 tatsächlich zu ändernden oder hinzuzufügenden Module, Generatorengstellen, algebraischen Seeds und Abarbeitungsphasen.

Semantische Klarstellungen fuer den Strengthening-Pfad

- **Zum Referenzobjekt:** Das konkrete IDEAL_∞ des Plans ist im Referenzfall kein Cayley-Graph und kein frei schwebendes Diagramm, sondern der einfachste robuste rooted ToC-Fall: ein unendlicher regulaerer Verzweigungsbaum in adressierter ToC-Praesentation. Die zweite legale Familie ist nicht unbekannt, sondern der levelabhaengig verzweigende Baum. Alles Weitere ist Zusatzvariation, nicht stillschweigende Voraussetzung.
- **Zur Mathematik von `SpectralDecomposition`:** Der nichttriviale Inhalt liegt nicht in der informellen Tatsache, dass endliche selbstadjungierte Matrizen diagonalisierbar sind, sondern in der proof-carrying und moeglichst berechenbaren Bereitstellung geordneter Eigenwerte, Projektoren, Orthogonalitaet und Exportsaetzen fuer spaetere Seeds und Handoffs.
- **Zur Solver-Frage:** Ziel des Strengthenings sind keine Gleitkomma- oder Blackbox-Inversionen, sondern exakte endliche Operatorobjekte ueber den im Pfad verwendeten algebraischen Grundkoerpern. Wo volle Konstruktivitaet nicht sauber erzielbar ist, bleibt nur ein lokal gekapselter, whitelist-faehiger Solver-Vertrag zulaessig; numerische Diagnostik und Referenzexperimente sind davon getrennt zu halten.
- **Zur Grenze von Pillar A:** A wird durch die endliche Matrix- und Seed-Seite nicht zum beliebigen Mathematik-Framework. Es liefert nur solche algebraischen, spektralen, thermischen, dynamischen, kanalischen und limesseitigen Samen, die direkt aus A-generierten Objekten derived-only entstehen. Haag-Kastler-Axiome, additive Netzlogik, GNS-/KMS-Theorie, offene-System-Interpretation, lokale Kovarianz und Matter/Gauge bleiben Folgearbeit der spaeteren Pfeiler.
- **Zur Raw/Stabilized-Dualitaet:** Der rohe Pfad wird nicht aus nostalgischen Gruenden mitgefuehrt. Er bleibt als Audit-, Regressions-, Regularisierungs- und Diagnoseflaeche erhalten, damit spaetere Aussagen ueber Stabilisierung, Kontraktion oder Closure-Provenienz nicht nur auf dem geglaetteten Endoperator, sondern auf der kontrollierten Transformation desselben Generatorursprungs beruhen.
- **Zur Lean-/mathlib-Problematik der komplexen algebraischen Seeds:** Die oeffentliche algebraische A-Wurzel darf im Strengthening nicht blind auf die analytisch aufgeruestete $\mathbf{C}$-Oberflaeche von Lean/mathlib gelegt werden. Die offiziellen mathlib-Dokumente halten fest, dass `Mathlib/Data/Complex/Basic` die komplexen Zahlen als $\mathbf{R}^2$ modelliert, `Mathlib/Data/Real/Basic` wiederum $\mathbf{R}$ als Cauchy-Vervollstaendigung rationaler Folgen aufbaut und `Mathlib/Analysis/Complex/Basic` $\mathbf{C}$ als normiertes Feld registriert. Fuer die A-seitige Generatorwurzel ist gerade diese analytische Aufruestung die problematische Stelle: sie ist fuer spaetere Analyse- und Topologieschichten nuetzlich, zieht aber frueh eine nichtausfuehrbare Infrastruktur in die oeffentliche Seed-Oberflaeche hinein. Dokumentierte Community-Praxis fuer ausfuehrbare algebraische Zahlobjekte ist daher nicht, die analytische Bibliotheksstruktur zu verbiegen, sondern einen eigenen algebraischen Typ mit expliziten Instanzen zu definieren und erst spaeter strukturtreu in die staerkere Theorie zu ueberfuehren; als Lehrmuster dient in *Mathematics in Lean* gerade der Aufbau der Gaussian Integers als eigener Datentyp mit Ringstruktur.[1]
- **Zur Konsequenz fuer den CNNA-Generator:** Ein frueh oeffentlich `noncomputable` gemachter Hermitian-/Hilbert-/$*$-Seed bringt spaeter vor allem abstrakte Existenz- und Transportaussagen, aber keinen belastbaren produktiven Generatorpfad fuer Referenzlauf, Variationslauf, Matrixwirkung, endlichen Zustandsraum oder rohe Rauchtests. Fuer Schritt 2 ist daher nicht ein kosmetischer Wrapper um $\mathbf{C}$ zulaessig, sondern eine explizit ausfuehrbare algebraische Wurzel, die spaeter per strukturerhaltender Bruecke nach $\mathbf{C}$ und in die analytische mathlib-Welt erweitert werden kann.

[1] Vgl. https://leanprover-community.github.io/mathlib4_docs/Mathlib/Data/Complex/Basic.html, https://leanprover-community.github.io/mathlib4_docs/Mathlib/Data/Real/Basic.html, https://leanprover-community.github.io/mathlib4_docs/Mathlib/Analysis/Complex/Basic.html sowie https://leanprover-community.github.io/mathematics_in_lean/.

12 Vorbereitungsblock V0–V5: oeffentlich computable algebraische A-Wurzeln

Der folgende Zusatzblock ist kein alternativer semantischer Pfad neben dem Strengthening, sondern eine *Vorbereitungs- und Reparaturstufe* fuer S1. Er wird genau dann geoeffnet, wenn die oeffentliche algebraische A-Wurzel ueber **C** in Lean in einen diffusen `noncomputable`-Status kippt oder wenn die Erweiterbarkeit nach **C** zwar fachlich gewuenscht, aber im aktiven Generatorpfad noch nicht computable realisiert ist. Der Block ist bewusst so formuliert, dass er spaeter *nicht* weggeworfen werden muss: sein Ziel ist ein ausfuehrbarer algebraischer Kern mit spaeterer strukturtreuer Bruecke ExecComplex $\to$ **C**, nicht ein provisorischer **Q**-Hack ohne Rueckbindung.

Zielbild des Vorbereitungsblocks

Am Ende des Blocks gilt:

- die oeffentliche algebraische A-Wurzel fuer `HermitianStructure`, `FiniteHilbert` und `MatrixAlgebra` ist **computable**;
- die algebraischen Seeds sind definitorisch **nicht** an die analytische **C**-Schicht gebunden, sondern laufen ueber eine kleine explizite Sternring-/Ringoberflaeche;
- `ExecComplex` ist als eigener ausfuehrbarer Datentyp mit expliziten Instanzen vorhanden und dient als erste konkrete Produktivinstanz;
- eine spaetere strukturtreue Erweiterung ExecComplex $\to$ **C** ist bereits vorbereitet und nicht erst ein spaeterer Architekturumbau;
- die spaeteren Seeds konsumieren dieselbe algebraische Wurzel statt lokaler Hilfskonstruktionen oder stillschweigender **C**-Spezialpfade;
- die algebraische A-Wurzel wird explizit als **paralleler Dualstrang** gefuehrt: ein *operativer Produktivstrang* ueber `ExecComplex` und ein *analytischer Spiegelstrang* ueber **C**, verbunden nur ueber die bewiesene Bruecke; der strikte Generatorpfad darf zu keinem Zeitpunkt versehentlich den **C**-Strang als operativen Ausgabekanal konsumieren.

Praezisierung der Koeffizienten- und Parametergrenze

Der Vorbereitungsblock zieht `ExecComplex` bewusst *zunaechst* auf eine ausfuehrbare rationale Repräsentation, etwa $(a, b) \in \mathbf{Q}^2$. Das ist kein semantischer Ausweichtrick, sondern genau die Grenze, an der der produktive Generatorpfad in Lean heute tragfähig bleibt: fuer diskrete Graph-Laplacians, endliche Matrixcarrier und ganzzahlige bzw. rationale Gewichte reichen endliche algebraische Operationen ueber **Q** aus. Die Grenze dieses Startpunkts ist jedoch ebenfalls explizit festzuhalten: sobald spaetere thermische oder policy-nahe Achsen einen genuin reellen Parameter verlangen, darf dieser im produktiven A-Pfad nicht stillschweigend wieder als analytisches **R**-Objekt in die Wurzel einsickern. Insbesondere bleibt die thermische Achse β auf der Generatorseite bis auf weiteres ein sichtbarer rationaler Provenienz- und Variationsparameter; ihre **R**-Interpretation und spaetere physikalische Schließung erfolgen erst ueber die Bruecke in die komplexanalytische Welt und nicht bereits in der algebraischen A-Wurzel.

Operative Designregel

Die Vorbereitung folgt vier harten Regeln:

1. **Kein oeffentlicher analytischer Kurzschluss.** In den S1-Wurzelmodulen `HermitianStructure`, `FiniteHilbert` und `MatrixAlgebra` werden keine normierten, topologischen oder `RCLike`-artigen Klassen als oeffentliche Voraussetzung eingefuehrt.

2. **Expliziter Parallel-Dualstrang.** Die algebraische Wurzel wird von Anfang an als zwei theorematisch gekoppelte, aber semantisch getrennte Straenge gefuehrt: der *operative Produktivstrang* laeuft ueber `ExecComplex` und bleibt oeffentlich computable; der *analytische Spiegelstrang* laeuft ueber **C**, darf intern `noncomputable` sein und ist nur fuer Transfer, Analyse und spaetere Pfeiler zulaessig. Wie beim rohen/stabilisierten DtN darf der strikte Pfad nicht versehentlich auf den falschen Ausgang gestopft werden: Produktionsconsumer duerfen nur den `ExecComplex`-Strang konsumieren, nie die **C**-Spiegelinstanz als operative Oberflaeche.

3. **ExecComplex nur als erste Instanz, nicht als versteckte Semantik.** Die algebraischen Seeds werden definitorisch generisch ueber einer kleinen Ring-/Sternringoberflaeche formuliert; `ExecComplex` ist die erste konkrete Produktivinstanz, aber nicht die einzige moegliche spaetere Instanz.

4. **Bridge statt Zweitumbau.** Die spaetere Einbettung nach **C** wird von Anfang an als eigener strukturtreuer Brueckenbaustein geplant; spaetere spektrale, thermische oder analytische Pfeiler sollen diese Bruecke konsumieren koennen, ohne die A-Wurzel erneut umzuschreiben.

Vorbereitungsphasen V0–V5

Phase	Betroffene Module	Muss am Phasenende wahr sein
V0	`Foundation/MatrixNorms`, `Foundation/HermitianStructure`, `Foundation/FiniteHilbert`, `Foundation/MatrixAlgebra`, `Foundation/Notation`, `Foundation/BuildAll`, Repo-Audit	die Lean-/mathlib-Problematik der oeffentlichen komplexwertigen Seed-Oberflaeche ist explizit dokumentiert; es ist festgehalten, welche Teile nur algebraisch/exekutiv gebraucht werden und welche spaetere analytische Consumer bleiben; zugleich ist der algebraische Dualstrang als `ExecComplex`-Produktivpfad plus **C**-Spiegelpfad ausdruecklich als parallele Architektur festgehalten; jeder oeffentliche `noncomputable`-Rest in der algebraischen A-Wurzel ist als Planverstoß klassifiziert, sofern er nicht in die Whitelist (lokaler Solver, IDEAL-Transport, genuine Limesselektion oder spaetere **C**-Spiegelinstanz) faellt
V1	`Foundation/ExecComplex`, optional `Foundation/ExecComplexLemmas`	ein eigener ausfuehrbarer Datentyp `ExecComplex` mit expliziten Komponenten (zunaechst rational), `Zero/One/Add/Neg/Sub/Mul/Star`, mindestens `CommRing`- und `StarRing`-Struktur sowie elementaren Rechen- und Injektivitaetssaetzen steht; explizit festgehalten ist dabei, warum $\mathbf{Q}^2$ fuer den Generatorpfad zunaechst reicht und wo die Grenze liegt: diskrete Laplace-/Matrixoperationen laufen algebraisch/exekutiv, waehrend die sichtbare thermische Achse β im produktiven A-Pfad vorerst rationaler Provenienzparameter bleibt und ihre **R**-Lesart erst spaeter ueber die Bruecke erfolgt; `ExecComplex` ist oeffentlich computable und benoetigt keinen Rueckgriff auf die analytische **C**-Schicht
V2	`Foundation/MatrixNorms`, `Foundation/HermitianStructure`, `Foundation/FiniteHilbert`, `Foundation/MatrixAlgebra`	die algebraischen Seeds sind definitorisch generisch ueber einer kleinen Sternring-/Ringoberflaeche formuliert: Adjungierte/Hermitizitaet, finiter Zustandsraum, sesquilineare Seed-Oberflaeche, Matrixwirkung, $*$-Algebra, Kommutator und Antikommutator stehen *ohne* `RCLike`, `NormedField`, `InnerProductSpace` oder andere analytische Klassen als oeffentliche Voraussetzung; zugleich ist `MatrixNorms` als erster expliziter Dualstrang-Consumer so geschaerft, dass Frobenius-Quadrat, Positivitaets- und Shift-Oberflaeche auf der `ExecComplex`-Seite computable liegen, waehrend analytische Normsaetze nur im getrennten **C**-Spiegelstrang verbleiben; `ExecComplex` ist erste konkrete Instanz dieser allgemeinen Schicht
V3	`Foundation/MatrixNorms`, `Foundation/Notation`, `Foundation/Interfaces`, `Foundation/BuildAll`	es gibt eine saubere ausfuehrbare Spezialisierung fuer den Generatorpfad (z.B. `ExecMat`, `ExecState`); die Notation verdeckt weder den Koeffiziententyp noch abhaengige Transporte; `Interfaces` konsumiert die algebraische Wurzel nur ueber explizite Vertragsobjekte und fuehrt keinen zweiten semantischen Pfad neben die generische Seed-Oberflaeche ein; `MatrixNorms` exportiert seine operative Shift- und Vergleichsseite ebenfalls nur als `ExecComplex`-Produktivoberflaeche

Phase	Betroffene Module	Muss am Phasenende wahr sein
V4	`Foundation/ExecComplexBridge`, `Foundation/MatrixNorms`, ggf. kleine lokale Hilfssaetze in `HermitianStructure/FiniteHilbert/MatrixAlgebra`	eine spaetere Erweiterung nach **C** ist strukturell vorbereitet: es gibt einen strukturtreuen Morphismus `ExecComplex` →`**` **C** bzw. eine aequivalente explizite Abbildung, und **Gate-Bedingung dieser Phase ist ihre Injektivitaet**; nur so bleiben Matrixeintraege, Operatoren und spaetere Spektralaussagen sauber transportierbar. Dazu liegen Vertraeglichkeitssaetze fuer `star`, Matrixabbildung, Vektorabbildung, Adjungierte, inneres Produkt, Kommutator und die von `MatrixNorms` exportierten Frobenius-/Shift-Groessen vor; nur dieses Brueckenmodul darf die starke **C**-Schicht direkt importieren, und jede **C**-seitige Spiegelinstanz ist auditierbar als paralleler Nicht-Produktivstrang zu kennzeichnen
V5	alle in V0–V4 betroffenen Module, `StrengtheningPhase`-Audit, Phase S1	der Vorbereitungsblock ist in S1 reintegriert: die oeffentliche algebraische A-Wurzel ist wieder computable, `MatrixNorms` ist als erster expliziter Dualstrang-Consumer legal umgestellt, die spaetere Erweiterbarkeit nach **C** ist proof-carrying vorbereitet, die Audits greifen auf die neue Koeffizienten- und Brueckendisziplin, und Phase S1 darf erst dann als geschlossen gelten, wenn `MatrixNorms/HermitianStructure/FiniteHilbert/MatrixAlgebra` diese Disziplin sichtbar konsumieren

Folge fuer die eigentlichen Strengthening-Phasen

Der Vorbereitungsblock aendert die spaeteren Ziele von S2–S14 nicht, verschiebt aber die Schliessungsbedingung von S1: S1 ist erst dann fachlich grun, wenn die computable Referenzfamilie *und* die oeffentliche algebraische A-Wurzel gemeinsam legal geschlossen sind. Wo der Block geoeffnet wird, ist er daher als S1-Vorstufe bzw. S1a–S1e zu lesen, nicht als spaeter nachziehbarer Komfortblock.

Mit nun gruenem V0-Stand Die Vorbereitungsphase V0 ist auf dem aktiven Pfad geschlossen: `Foundation/HermitianStructure`, `Foundation/FiniteHilbert` und `Foundation/MatrixAlgebra` sind nun als neue algebraische Seed-Module im aktiven Build angelegt, `Foundation/Notation` und `Foundation/BuildAll` sind auf diese Vorstufe fortgeschrieben, und die Importkante nach `Foundation/Interfaces` bleibt dabei konsumdisziplinier, also ohne die neue Algebra-Schicht implizit in jeden spaeteren Consumer einzuschmuggeln.

Inhaltlich schliesst V0 damit genau die beabsichtigte Vorbereitungsstufe und nicht mehr: Die algebraische A-Wurzel ist jetzt explizit als generische Seed-Oberflaeche mit spaeterem `ExecComplex`-Produktivpfad und getrenntem **C**-Spiegelpfad markiert, aber weder `ExecComplex` selbst noch die legale Umstellung von `MatrixNorms` auf den operativen Dualstrang noch die injektive Bruecke ExecComplex → **C** sind dadurch schon geschlossen. Folglich bleiben V1–V5 echte Restphasen; fuer S1 bedeutet der gruene V0-Stand deshalb eine geoeffnete und auditierbar vorbereitete algebraische Wurzel, aber noch nicht ihre endgueltige computable Closure.

Mit nun gruenem V1-Stand Die Vorbereitungsphase V1 ist auf dem aktiven Pfad geschlossen: `Foundation/ExecComplex` ist als eigener ausfuehrbarer Koeffiziententyp im aktiven Build angelegt, `Foundation/ExecComplexLemmas` stellt die elementaren Rechen-, Komponenten- und Injektivitaetssaetze der neuen Wurzel explizit bereit, und `Foundation/Notation` sowie `Foundation/BuildAll` konsumieren diese Vorstufe sichtbar, ohne dadurch bereits die spaeteren analytischen **C**-Consumer in die oeffentliche Produktivschicht hineinzuziehen. Zugleich ist die Importkante von `ExecComplex` bereinigt: der breite Komfortimport `Mathlib` ist entfernt, und der neue Koeffiziententyp haengt nur noch an der fuer `CommRing.ofMinimalAxioms`, rationale Koeffizienten, Sternstruktur sowie `ext/ring` tatsaechlich benoetigten Mathlib-Oberflaeche.

Inhaltlich schliesst V1 damit genau die beabsichtigte computable Wurzelstufe und nicht mehr: Der operative Produktivstrang besitzt nun mit `ExecComplex` eine explizite, oeffentlich computable algebraische Basis mit `Zero/One/Add/Neg/Sub/Mul/Star`, mindestens `CommRing`- und `StarRing`-Struktur sowie den benoetigten elementaren Komponenten- und Rechenfakten. Dabei bleibt die rationale $\mathbf{Q}^2$-Darstellung bewusst ein Startpunkt des Generatorpfads und kein verdeckter semantischer Endzustand: Weder ist `MatrixNorms` damit schon legal auf den operativen Dualstrang umgestellt

noch ist die injektive Bruecke ExecComplex → **C** bereits geschlossen. Folglich bleiben V2–V5 echte Restphasen; fuer S1 bedeutet der gruene V1-Stand deshalb eine nun oeffentlich computable algebraische A-Wurzel, aber noch nicht die vollstaendige Re-Integration des Vorbereitungsblocks in die Strengthening-Hauptkette.

Mit nun gruenem V2-Stand Die Vorbereitungsphase V2 ist auf dem aktiven Pfad geschlossen: `Foundation/MatrixNorms`, `Foundation/HermitianStructure`, `Foundation/FiniteHilbert` und `Foundation/Ma` konsumieren die algebraische A-Wurzel nun sichtbar ueber eine kleine generische Sternring-/Ringoberflaeche statt ueber oeffentliche analytische Klassen, und die Importdisziplin der V2-Module ist auf die tatsaechlich benoetigten Mathlib-Bausteine zurueckgenommen. Insbesondere ist `MatrixNorms` als erster expliziter Dualstrang-Consumer legal geschaerft: die operative Frobenius-, Positivitaets- und Shift-Oberflaeche liegt auf der `ExecComplex`-Seite computable, waehrend der **C**-Spiegelstrang getrennt und auditierbar bleibt.

Inhaltlich schliesst V2 damit genau die beabsichtigte generische Algebra- und Dualstrangstufe und nicht mehr: Die oeffentliche algebraische A-Wurzel ist nun nicht nur computable vorhanden, sondern in ihren ersten Seed-Consumern auch sichtbar legal konsumiert. Noch offen bleiben jedoch die saubere ausfuehrbare Spezialisierung des Generatorpfads ueber explizite Vertragsobjekte in `Foundation/Interfaces`, die proof-carrying Bruecke ExecComplex → **C** sowie die vollstaendige Re-Integration des Vorbereitungsblocks in S1. Folglich bleiben V3–V5 echte Restphasen; fuer S1 bedeutet der gruene V2-Stand deshalb eine nun sichtbar legalisierte algebraische A-Wurzel, aber noch nicht den vollstaendigen Abschluss des Vorbereitungsblocks. Fuer die eigentlichen Strengthening-Phasen oberhalb von V2 ergibt sich daraus keine Umplanung: Die Reihenfolge V3 → V4 → V5 bleibt verbindlich, und insbesondere darf ein etwaiger analytischer **C**-Spiegelpfad bis zur geschlossenen Brueckenphase V4 nicht als operative Produktionsoberflaeche einspringen.

Mit nun gruenem V3-Stand Die Vorbereitungsphase V3 ist auf dem aktiven Pfad geschlossen: `Foundation/MatrixNorms`, `Foundation/Notation`, `Foundation/Interfaces` und `Foundation/BuildAll` tragen nun eine sichtbare ausfuehrbare Spezialisierung des algebraischen Produktivstrangs. Insbesondere exponiert `Interfaces` die operative A-Wurzel ueber explizite Vertragsobjekte und generische Netzhuellen, statt einen zweiten semantischen Pfad neben die bereits in V2 legalisierte Seed-Oberflaeche zu setzen; zugleich bleibt `MatrixNorms` auch auf dieser Spezialisierungsstufe strikt auf die computable `ExecComplex`-Produktivseite beschraenkt. Die in der Implementierung nachgezogene `SeedScalar`-Instanzpropagation fuer `ExecComplex` macht dabei explizit sichtbar, dass V3 nicht nur Alias-/Notationskosmetik schliesst, sondern die algebraische Wurzel als konsumierbare operative Instanz wirklich bis in die Vertrags- und Spezialoberflaeche hinein traegt.

Inhaltlich schliesst V3 damit genau die beabsichtigte ausfuehrbare Spezialisierungs- und Vertragsstufe und nicht mehr: Der Generatorpfad besitzt nun eine lesbare operative Oberflaeche ueber `ExecMat`, `ExecState` und verwandte Spezialisierungen, ohne dabei den Koeffiziententyp oder abhaengige Transporte semantisch zu verdecken, und `Interfaces` bleibt weiterhin strikt Consumer der algebraischen Wurzel statt selbst neue Grundsemantik zu erfinden. Noch offen bleiben jedoch die proof-carrying Bruecke ExecComplex → **C** sowie die vollstaendige Re-Integration des Vorbereitungsblocks in S1. Folglich bleiben V4–V5 echte Restphasen; fuer S1 bedeutet der gruene V3-Stand deshalb eine nun operativ konsumierbare und vertragsseitig sichtbare algebraische A-Wurzel, aber noch nicht den vollstaendigen Abschluss des Vorbereitungsblocks. Fuer die eigentlichen Strengthening-Phasen oberhalb von V3 ergibt sich daraus keine Umplanung: Die Reihenfolge V4 → V5 bleibt verbindlich, und insbesondere darf der analytische **C**-Spiegelpfad bis zur geschlossenen Brueckenphase V4 weiterhin nicht als operative Produktionsoberflaeche einspringen.

13 Strengthening-Modulliste S0–S14

Für flankierende Diagnose- und Limesmodule wird die IDEAL/REAL-Trennung im Feld Muss lokal schliessenëxplizit markiert, damit die Trennungsregel nicht nur in den Phasengates, sondern bereits

in der Modulliste pruefbar bleibt.

Nr.	Modul / Bereich	Aktion	Muss lokal schließen	Muss wie verzahnen
1	`Foundation/SubstrateClass`	ändern	die bereits vorhandene typklassenparametrische Universalität explizit als produktive Generatorbasis ausweisen; kein zweiter Meta-Vertrag neben `SubstrateClass`	bleibt primitive Wurzel für alle späteren Consumer und für jede legale Instanziierung des Produktivpfads
2	`Foundation/MatrixNorms`, `Foundation/Interfaces`, `Foundation/Determinism`	ändern	Matrix-, Norm-, Spur- und Instanzgrundlage so schärfen, dass endliche Operator-, Zustands-, Netz- und Diagnosepfade nicht auf informelle Hilfslogik ausweichen; `Foundation/Interfaces` trägt dabei ausdrücklich generische Netz-, Prägarben-, Koprägarben- und Vertragsobjekte wie `RegionNet`, `PreNet`, `LocalAlgebraNet`, `StateNet`, `ChannelNet` sowie einen A-seitig benannten `StarAlgebraContract`; deterministische Zusätze bleiben flankierende Verfeinerung	versorgt `Contract`, finite Operatorseeds, spätere Handoff-Instantiierungen und Audits, ohne eine versteckte stärkere Importvoraussetzung in P7–P27 einzuschmuggeln
3	`Foundation/HermitianStructure`	hinzufügen	Adjungierte, Hermitizität/Selbstadjungiertheit, unitäre Matrizen und die zugehörigen elementaren Rechenregeln über endlichen Matrixcarriern als *generische* Sternring-Wurzel bereitstellen; keine öffentliche Bindung an **C**-Analyseklassen	liefert die algebraische Vorstufe für Spektralzerlegung, $*$-Algebra, Zeitentwicklung und spätere GNS-/KMS-Pfade, ohne die ausführbare Generatorwurzel an die analytische Bibliothek zu ketten
4	`Foundation/FiniteHilbert`	hinzufügen	endliche Zustands- und Inner-Product-Seed-Oberfläche über demselben algebraischen Koeffizienten bereitstellen; Orthonormalbasis, Matrixwirkung und Spur-/Inner-Product-Zusammenhang derived-only und öffentlich computable, ohne vorzeitig lineartopologische Klassen einzuführen	macht Spektralzerlegung, Zustandsraum und spätere Darstellungsseeds als gemeinsame Consumer derselben Hilbertraumwurzel lesbar und hält die spätere ExecComplex → **C**-Erweiterung offen
5	`Foundation/MatrixAlgebra`	hinzufügen	endliche Matrix-$*$-Algebra über derselben generischen Sternring-Wurzel samt Kommutator, Antikommutator und elementaren Algebra-/Lie-Sätzen bereitstellen; konkrete Produktivinstanz ist `ExecComplex`, nicht blind **C**	verhindert B-/C-seitigen Nachbau der algebraischen Sprache aus bloßen Matrixoperationen und hält eine strukturtreue Brücke in die komplexanalytische Welt explizit vorbereitet
6	`Foundation/ConcreteSubstrate`	ändern	bestehende konkrete Referenzfamilie des regulären konkreten IDEAL-ToC so reparieren, dass die `SubstrateClass`-Instanz ohne `sorry`s und ohne vermeidbare `noncomputable`-Kontamination steht	liefert den ersten robusten Referenzlauf des späteren Generators
7	`Foundation/LevelVariableSubstrate`	hinzufügen	zweite legale Substratfamilie mit levelabhängigem Branching als echter Variationsfall; Determinismus darf hier, falls vorhanden, nur Zusatz und nicht Kernannahme sein	liefert den ersten belastbaren Nicht-Referenzfall für Universalisierungs- und Diskriminanzsätze
8	`Foundation/WeightPolicy`, `Foundation/RegularizationPolicy`	hinzufügen	Gewichte, numerisch-kanonische Regularisierung und die sichtbare thermische Achse β als explizite A-seitige Variations- und Provenienzparameter modellieren; keine versteckte Policy-Logik in `DirichletLaplacian`, `DtN` oder späteren Seeds; Legacy-`Policy`-Felder wie `rStar`, `alpha` oder `alphaDenom` gelten dabei nicht als primitive neue Motorachsen, sondern im Referenzfall als numerische bzw. derived-only Hilfsgroessen, solange keine spaetere mathematische Notwendigkeit fuer eine eigenstaendige Variationsachse nachgewiesen ist	zieht `DirichletWeight` und die Herkunft von ϵ, β aus der Mittelstufe an die Motorseite des Generators, ohne ihre spaetere pfeileruebergreifende Schliessung zu behaupten

Nr.	Modul / Bereich	Aktion	Muss lokal schließen	Muss wie verzahnen
9	`Foundation/SubstrateAnalysis`	hinzufügen	explizit klassifizieren, welche Resultate wirklich nur `SubstrateClass` benötigen und welche stärkere Zusatzannahmen wie Determinismus oder Uniformität verbrauchen	macht Universalität und echte Variationsaussagen formal statt nur proseartig sichtbar
10	`ToC/Contract`, `ToC/Addressing`, `ToC/IdealAddressEquiv`, `ToC/ConcreteIdeal`	ändern/hinzufügen	den konkreten Referenzfall des IDEAL-ToC als Familie schließen und sauber an Contract/Addressing/Equiv rückbinden	liefert den konkreten Idealstartpunkt, aus dem `ToCStrong` produktiv hervorgeht
11	`ToC/Concrete`	ändern	den ersten realen `ToCStrong`-Kern so formulieren, dass Referenzfall und Familieninstanz denselben Downstream-Pfad öffnen	versorgt den Finite-Pfad ohne Smoke-Test-Sonderroute
12	`Finite/CutSpec`, `Finite/RegionCore`, `Finite/BoundaryPorts`, `Finite/Approximant`, optional `Finite/Selection`	ändern	den endlichen Box-Pfad vereinheitlichen, sodass beide Familienfälle ab hier wirklich dieselben Consumer bedienen und als identischer Buildpfad überprüfbar bleiben; zugleich müssen alle exponierten Carrier/Prädikate die nötigen `Fintype`-/`Decidable`-Instanzen mitliefern	überführt alle legalen Einstiege in denselben kanonischen endlichen Box-Inhalt
13	`Finite/DirichletLaplacian`	ändern	den noch freien `DirichletWeight`-Eingang explizit als legitimen A-seitigen Wurzelinput über `WeightPolicy` modellieren; Referenzfälle dürfen zusätzliche kanonische Default-Gewichte tragen, die universelle Theorie aber nicht	bleibt erster operatorischer Consumer des produktiven Generatorpfads und kennt Gewichte nur über die freigegebene Policy-Verzahnung
14	`DtN/DtN`	ändern	die aktuelle nackte `InteriorInverse`-Feldform nicht als Wurzelinput, sondern als zu internalisierende Solver-/Eliminationslogik behandeln; Ziel ist eine algorithmische Ableitung mit öffentlicher Solve-/Korrektheitsoberfläche, wobei ein eventuell verbleibender `noncomputable`-Zeuge nur intern und lokal sichtbar bleiben darf	hält den binären DtN-Kern als echten Folgeconsumer von `DirichletLaplacianStrong` und verhindert eine lose Inversenschraube im Generatorpfad
15	`DtN/DtNStabilized`	ändern	freie `epsilon`-Seite so weit legal möglich in eine kanonische computable Shift-/Regularisierungsprovenienz überführen	bleibt einziger legaler Stabilisierungsconsumer für Closure- und Coupling-Pfade
16	`Closure/ParameterClosure`, `Closure/RegularizationClosure`	ändern	Closure-Seite so verschärfen, dass primitive Restparameter nicht unkommentiert weitergereicht werden und die aus P13 kommende Provenienz explizit bleibt	exportiert eine operative Closure-Oberfläche für spätere Handoffs und Vergleiche
17	`Coupling/GeneralizedDtN`	ändern	raw/stabilized-Dualität, Restriktionsdaten und operatorische Provenienz familiesicher schärfen; keine Reparatur durch spätere Exportmodule	koppelt Operatorpfad und Sektorseite explizit als legalen Consumerblock
18	`Coupling/MultiSchur`	ändern	die aktuelle nackte `interfaceInverse`-Feldform nicht als Wurzelinput, sondern als innerhalb von P22 zu internalisierende Restriktions-/Solver- bzw. Reduktionslogik behandeln; Exportziel sind Schur-/Glueing-/Korrektheitssätze statt einer losen Inversenmatrix	bleibt reiner Folgeconsumer von `GeneralizedDtNStrong` und hält die P22-Fachlogik von freien Schrauben frei
19	`Network/InfiniteCarrier`	ändern	stabile endliche Ausschnitte und gerichtete Übergänge so exportieren, dass Referenzfall und Variation auf derselben Trägeroberfläche vergleichbar bleiben; limesnahe Provenienz darf nicht hinter späteren Exportadaptern verschwinden	liefert die späte Trägeroberfläche für Handoff, Variation und spätere Limesdiagnostik

Nr.	Modul / Bereich	Aktion	Muss lokal schließen	Muss wie verzahnen
20	`Network/RegionNet`, `Network/SectorChannels`	unverändert prüfen	keine fachliche Neuformulierung im Strengthening; explizit festhalten, dass diese Supportmodule in Schritt 2 nur als bereits geschlossene legale Vorgänger konsumiert und nicht stillschweigend mitbearbeitet werden	verhindert Scheinarbeit im Supportpfad und hält die Phasenzuordnung prüfbar
21	`Network/SectorSysEnv`, `Network/RelativeEntropyFlow`, `Geometry/LCPMeasure`, `Geometry/Foliation`, `Geometry/SpacetimePaths`	ändern	IDEAL/REAL getrennt: ja. Flankierende Skalen-, Schicht-, Präfix-, Window- und Environment-Provenienz für spätere Spektraldimensions- sowie Symmetrie-/Cutoff-/Limesdiagnostik derived-only sichtbar halten und dabei die IDEAL-Seite strikt von der REAL-Seite trennen	liefert A-seitige Diagnosewerkzeuge für spätere Pfeiler, ohne den Generatorhauptpfad rückwärts zu verunreinigen
22	`Network/DirectedLimit`	hinzufügen	IDEAL/REAL getrennt: ja. Gerichtete Übergänge, Konvergenzbegriff und Limes-Eindeutigkeit für die Lesart $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$ formalisieren, ohne bereits B- oder AQFT-Limessemantik vorwegzunehmen	macht limesseitige Aussagen aus `InfiniteCarrier` und Exportdaten formal statt nur proseartig
23	`Finite/SpectralDecomposition`	hinzufügen	endliche Spektralzerlegung des symmetrischen/selbstadjungierten Dirichlet-Operators als proof-carrying Daten mit geordneten Eigenwerten, Projektoren, unitärer Diagonalisierbarkeit und Vollständigkeit bereitstellen	liefert B-seitig benötigte spektrale Theoremsamen aus A-generierten Operatoren, ohne AQFT-Semantik nach vorn zu ziehen
24	`Finite/StateSpace`	hinzufügen	positiven Kegel, Zustandsraum, Spurform und Normierungsoberfläche über endlichen Matrixcarriern derived-only und möglichst computable bereitstellen	verhindert, dass spätere Pfeiler den endlichen Zustandsraum aus `MatrixNorms` und eigener Hilfslogik neu aufbauen
25	`Finite/MatrixExponential`, `Finite/GibbsStateSeed`, `Finite/DynamicsAdapter`	hinzufügen	Matrixexponentialfunktion, Partitionsfunktion, diagonalen Gibbs-Zustand mit Positivitaet und Normierung sowie den Heisenberg-Adapter $\alpha_t(A) = U(t)AU(t)^\dagger$ als `StarAlgDynamics`-Vorstufe über endlichen A-seitigen Operatoren bereitstellen, ohne bereits KMS-/AQFT-Semantik zu behaupten	liefert thermische und dynamische Seeds für spätere B-Pfade direkt aus A-generierten Operatoren und verhindert B-seitigen Nachbau roher Matrixarithmetik
26	`Finite/UnitaryEvolution`	hinzufügen	unitäre Einparametergruppe $t \mapsto e^{-itH}$ samt Gruppen-, Unitaritäts- und elementaren Heisenberg-Sätzen über der endlichen Matrix-$*$-Algebra bereitstellen; keine volle Dynamikaxiomatik vorwegzunehmen	liefert die dynamische Wurzel für spätere Heisenberg-, Modular- und OQS-Verbraucher, ohne B/C-Fachlogik
27	`Finite/ChannelSeed`	hinzufügen	Kraus-/Choi-/CPTP-Vorstufe und gegebenenfalls dissipative Halbgruppen-Seeds soweit bereitstellen, wie sie rein algebraischer Consumer der endlichen A-Matrixseite bleiben; keine Vorwegnahme von C-Fachlogik	verhindert verbotenen C-seitigen Nachbau der algebraischen Kanalwurzel und hält die kanalische Vorstufe dennoch strikt unterhalb echter OQS-Fachlogik
28	`Handoff/SectorExport`, `Handoff/Step1StrongCore`	ändern	Export- und Verdichtungsstufe um echte Generator-, algebraische, spektrale, zustandsräumliche, thermische, unitäre und Diagnoseprovenienz erweitern; insbesondere IDEAL-seitige Grenz-, Träger- und Äquivalenzdaten getrennt von REAL-seitigen Cutoff-, Komplement-, Übergangs- und Limesdaten zielneutral nach außen stellen	projiziert nur geschlossene reife A-Daten nach außen und macht spätere Pfeiler unabhängig von informellen Rekonstruktionen oder algebraischem Nachbau

Nr.	Modul / Bereich	Aktion	Muss lokal schließen	Muss wie verzahnen
29	`Handoff/Step1MathData`, `Handoff/ABHandoffStrong`, `Handoff/PillarAStep1Closed`	ändern	Endstufe so verschärfen, dass Referenzlauf und Variationslauf denselben proof-carrying Handoff schließen; `ABHandoffStrong` bleibt nur ausgehender Adapter und wird weder von A selbst konsumiert noch in B durch A-Nachbau ersetzt	bleibt einzig legitimer Ausgang des produktiven Pillar-A-Pfads und schließt Zirkularität zwischen Kern, Export und Adapter aus
30	`PillarA/Generator`	hinzufügen	expliziter Root-to-Handoff-Builder über dem Dreiklang *(Substrat, WeightPolicy, Cutoff)*; zusätzlich ein explizites Computability-Budget der Zahnräder festhalten; öffentliche Exportoberfläche soll vollständig computable sein, soweit nicht eine explizit whitelistete Restgrenze greift	macht den produktiven Smoke-Test zum echten Generatorlauf desselben aktiven Pfads und verhindert schleichende `noncomputable`-Erosion
31	`PillarA/VariationAnalysis`	hinzufügen	Vergleichsoberfläche für substratnahe, policy-, regularisierungs- und beta-nahe, cutoffnahe und genuinely universelle Invarianten; Theoreme sind explizit nach Universalitätstyp zu klassifizieren	erlaubt Aussagen darüber, welche Resultate universell, substratabhängig, policyabhängig, ϵ-/β-abhängig oder cutoffabhängig sind
32	`PillarA/Diagnostics/SpectralDimension`	hinzufügen	IDEAL/REAL getrennt: ja. Flankierende Diagnosevorstufe mit explizit getrennter IDEAL-Seite und REAL-Seite; keine Vorwegnahme eines physikalischen Spektraldimensionssatzes	bereitet spätere RG-, Backreaction- und Spektraldimensionspfade innerhalb von A vor, damit B und folgende Pfeiler keinen A-Code nachtragen müssen
33	`PillarA/Diagnostics/SymmetryCutoffLimit`	hinzufügen	IDEAL/REAL getrennt: ja. Flankierende Diagnosevorstufe mit explizit getrennter IDEAL-Seite und REAL-Seite; keine fertigen physikalischen Brechungssätze	stellt spätere Symmetrie-, Cutoff- und Limeswerkzeuge A-seitig bereit und hält ihre Herkunft strikt von späterer AQFT-/Modularsemantik getrennt
34	`Foundation/Notation`, `ToC/Notation`, `Finite/Notation`, `DtN/Notation`, `Closure/Notation`, `Coupling/Notation`, `Network/Notation`, `Geometry/Notation`, `Handoff/Notation`, `PillarA/Notation`	ändern	neue Lesbarkeitsoberflächen für Referenzfamilie, Variationsfamilien, `WeightPolicy`, Hermitian-/Hilbert-/Algebra-Seeds, Spektral-/Zustands-/Thermal-/Dynamikseeds, Generator, Directed-Limit und Diagnosemodule einziehen, ohne Parameter oder abhängige Transporte zu verdecken	hält den Strengthening-Pfad reviewbar und bleibt strikt scoped sowie semantisch neutral
35	`Foundation/BuildAll`, `ToC/BuildAll`, `Finite/BuildAll`, `DtN/BuildAll`, `Network/BuildAll`, `Geometry/BuildAll`, `Handoff/BuildAll`, `PillarA/BuildAll`, `CNNA/BuildAll`	ändern	neue Module legal in den aktiven Build einhängen; keine zweite Schatten-Topologie und kein Rückgriff auf Archivpfade	macht den Strengthening-Pfad zum einzigen aktiven Produktions-, Diagnose- und Smoke-Test-Baum

13.1 Generatorengstellen und ihre Sollklassifikation

Objekt	Sollklassifikation	Zielmodul	Öffentliche Verzahnung	Migrationsschritt
`DirichletWeight`	legitimer A-seitiger Wurzelinput des universellen Pfads; auf Referenzpfaden zusätzlich kanonischer Default möglich	`Foundation/WeightPolicy`, konsumiert in `Finite/DirichletLaplacian`	`WeightPolicy.assign` bzw. ein `ofApproximant`-Builder; spätere Consumer sehen nur das freigegebene Policy-Gesetz, nicht freie Gewichtsfelder	freie Mittelstufen-Eingabe beseitigen; Gewichte an die Motorseite ziehen und optional kanonische Referenzpolicy für IDEAL/REAL-Referenzlauf ergänzen

Objekt	Sollklassifikation	Zielmodul	Öffentliche Verzahnung	Migrationsschritt
`InteriorInverse`	kein Wurzelinput; Ziel ist algorithmische Ableitung als Solver-/Eliminationslogik des P12-Zahnrads	`DtN/DtN`	öffentliche Solve-, Randoperator- und Korrektheitssätze; ein eventuell verbleibender Inversionszeuge bleibt intern und lokal	nacktes Inversenfeld aus der öffentlichen Generatorverzahnung entfernen oder streng internalisieren; `noncomputable` nur am lokalen Solve-Rand akzeptieren
`interfaceInverse`	kein Wurzelinput; Ziel ist algorithmische Ableitung als Restriktions-/Solver- bzw. Reduktionslogik der P22-Schur-Stufe	`Coupling/MultiSchur`	öffentliche Schur-, Glueing-, Restriktions- und Korrektheitssätze; keine lose Inversenmatrix als Exportoberfläche	nacktes Inversenfeld aus der öffentlichen Generatorverzahnung entfernen oder streng internalisieren; Schur-Reduktion als echtes Zahnrad der P22-Fachlogik formulieren

13.2 Computability-Audit des Generators

Generatorzahnrad	Sollstatus	Akzeptanzkriterium	Bemerkung
Wurzel- und ToC-Seite bis `ToCStrong`	`computable`	alle öffentlichen `def`s kompilieren ohne `noncomputable`; Referenzfamilie und Variationsfamilie instanziieren den Pfad ohne `sorry`, ohne vermeidbare `noncomputable`-Kontamination und mit expliziten Instanzen für endliche Carrier dort, wo sie exponiert werden	hier darf keine schleichende `classical`-Abkürzung stehen bleiben
Finite Box-Seite bis `ApproximantStrong`	`computable`	alle exponierten Carrier, Teilmengen und Prädikate liefern `Fintype`-, `DecidableEq`- und `Decidable`-Instanzen; die öffentliche Oberfläche bleibt auswertbar; `grep` auf öffentliche `noncomputable def`s dieses Zahnrads muss leer sein	dies ist die Wurzel fast aller späteren Berechenbarkeitsfragen
MatrixNorms-Dualstrang `MatrixNorms`	operative Seite `computable`, analytischer Spiegelstrang nur nach Whitelist (d)	`frobeniusSq`, Nulltest, Positivitaet bei $\neq 0$ und die vom Generator konsumierte Shift-Oberflaeche laufen oeffentlich computable ueber `ExecComplex`; jede volle Frobenius-/Normspiegelung ueber $\mathbf{C}$ ist als parallele Nicht-Produktivinstanz gekennzeichnet und nur ueber die injektive Bruecke konsumierbar	`MatrixNorms` ist der erste Normalfall der Whitelist-Klasse (d) und der einzige legale Zulieferer der computablen Shift-/Vergleichsseite fuer `DtNStabilized`

Generatorzahnrad	Sollstatus	Akzeptanzkriterium	Bemerkung
Algebraische A-Wurzeln `HermitianStructure/FiniteHilbert/MatrixAlgebra`	`computable`	Adjungierte, innere Produkte, $*$-Algebra, Kommutator und elementare algebraische Seeds stehen öffentlich ohne `noncomputable`; die öffentliche Produktivinstanz läuft über einen explizit ausführbaren Koeffiziententyp (Vorbereitungsblock `ExecComplex`), während die spätere **C**-Erweiterung nur über ein eigenes Brückenmodul als paralleler Spiegelstrang erfolgt; alle späteren finite Seeds konsumieren operativ den `ExecComplex`-Strang statt eigener Hilfslogik oder stillschweigender **C**-Spezialpfade	verhindert versteckte algebraische Basisarbeit in `SpectralDecomposition`, `StateSpace` oder B/C und schützt den Generator vor analytischer Frühkontamination
Operatorseite `DirichletLaplacian/DtNOnStable`, `DtNStabilized`	öffentlich `DirichletWeight` läuft über `WeightPolicy`; eventuell verbleibende Solve-/Inversionszeugen bleiben intern und theorematisch abgeschirmt; jede öffentliche `def` kompiliert ohne `noncomputable`; `DtNStabilized` konsumiert Shift- und Vergleichsgrößen ausschließlich aus der computablen `ExecComplex`-Seite von `MatrixNorms`	akzeptiert werden nur klar lokalisierte Restgrenzen, kein diffuser Globalstatus des Generators und kein versehentlicher Rueckgriff auf den **C**-Spiegelstrang
Spektral-, Zustands-, thermische und dynamische Seeds	öffentlich `computable`, intern höchstens lokal `noncomputable`	`SpectralDecomposition`, `StateSpace`, `MatrixExponential`, `GibbsStateSeed`, `UnitaryEvolution` und `DynamicsAdapter` exponieren nur kanonische, auswertbare Oberflächen; etwaige interne Nichtberechenbarkeit muss aus der Whitelist begründet werden. Fuer `MatrixExponential` ist die Designentscheidung explizit zu dokumentieren: im `ExecComplex`-Pfad ist zunächst nur eine endliche, computable Partialsummen- bzw. Approximationsoberfläche zulässig; der Zusammenhang zum analytischen Exponential über **C** wird erst über die Brücke theoremiert und nicht stillschweigend vorausgesetzt	diese Seeds dürfen nicht erst in B/C nachgebaut werden
Coupling- und Handoff-Seite bis `PillarAStep1Closed`	öffentlich `computable`, intern höchstens lokal `noncomputable`	Schur-/Glueing-/Handoff-Oberflächen sowie IDEAL/REAL-Provenienz bleiben von internen Solverdetails getrennt; spätere Pfeiler konsumieren eine kanonische, möglichst computable Exportfläche; `ABHandoffStrong` bleibt reiner Auswärtsadapter	falls lokale Nichtberechenbarkeit unvermeidbar bleibt, ist sie im Modul und im Plan explizit zu benennen

Whitelist für verbleibende `noncomputable`-Reste im aktiven Pfad. Jede `noncomputable def`

in einem Strengthening-Modul muss explizit genau einer der folgenden Klassen zugeordnet werden:

- **(a) lokale Matrixinversion / Solve-Logik:** ein interner Inversions- oder Solverzeuge, der öffentlich nur über Korrektheits- und Eliminationssätze exponiert wird;
- **(b) IdealAddressEquiv- oder vergleichbarer Transport über genuinely unendlichem Träger:** ein nicht-algorithmischer Äquivalenz- bzw. Transportrest der IDEAL-Seite;
- **(c) genuinely unendliche Filter-/Limesselektion:** ein unvermeidbarer `classical`-Rest auf der IDEAL- oder Directed-Limit-Seite, der nicht in öffentliche finite Seeds oder Handoffs einsickern darf;
- **(d) C-seitige Parallelinstanz mit Bridge-Transfer:** eine `noncomputable def`, die ausschliesslich als mathlib-**C**-Spiegelung einer bereits computable geschlossenen `ExecComplex`-Definition existiert, nur ueber die bewiesene injektive Bruecke konsumiert wird und **nicht** als eigenstaendige oeffentliche Produktionsoberflaeche des Generators dienen darf.

Alles andere gilt im Strengthening als Planverstoß und ist vor Phasenschluss zu beseitigen. Die Whitelist-Klasse (d) ist kein Freibrief fuer einen zweiten Generatorpfad, sondern dient ausschliesslich dazu, den analytischen Spiegelstrang auditierbar vom operativen Produktivstrang zu trennen.

14 Strengthening-Phasen S0–S14 mit Unterphasen S7a/S7b, S10a/S10b und S11a/S11b

Phase	Betroffene Module	Muss am Phasenende wahr sein
S0	Plan-Neuschnitt, betroffene `BuildAll`- und `Notation`-Dateien, Repo-Audit	Strengthening als Schritt 2 ist terminologisch und architektonisch sauber vom abgeschlossenen Refactor getrennt; zugleich ist durch Code-Audit explizit festgehalten, welche Builder bereits stehen, wo die realen Generatorengstellen sitzen, welche algebraischen Seeds spätere Pfeiler wirklich voraussetzen und dass Export/Handoff als reine Kupplung späterer Pfeiler zu lesen sind
S1	`Foundation/SubstrateClass`, `Foundation/MatrixNorms`, `Foundation/Interfaces`, `Foundation/Determinism`, `Foundation/HermitianStructure`, `Foundation/FiniteHilbert`, `Foundation/MatrixAlgebra`, `Foundation/ConcreteSubstrate`; gestarteter Vorbereitungsblock V0–V5 (nach gruenem V3 verbleiben V4–V5 offen)	die typklassenparametrische Universalität bleibt Wurzel des Produktivpfads; zugleich steht eine computable konkrete Referenzfamilie ohne `sorry` und ohne vermeidbare `noncomputable`-Kontamination, `MatrixNorms` ist als erster expliziter Dualstrang-Consumer legal umgestellt, die algebraische Matrix-/Hilbert-/$*$-Grundlage ist über eine öffentlich computable Wurzel mit vorbereiteter strukturtreuer Brücke ExecComplex → **C** ausgerichtet, und `Foundation/Interfaces` exponiert bereits die generischen Netz- und Vertragsobjekte, die spätere Pfeiler nur noch instantiieren dürfen
S2	`Foundation/LevelVariableSubstrate`, `Foundation/WeightPolicy`, `Foundation/RegularizationPolicy`, `Foundation/SubstrateAnalysis`	eine zweite legale Substratfamilie, eine explizite `WeightPolicy`-Achse und eine numerisch-kanonische `RegularizationPolicy`-Achse stehen; zugleich bleibt β als sichtbare thermische Provenienz- und Variationsachse erhalten und ist auf dem produktiven Generatorpfad zunaechst als rationaler Parameter modelliert, waehrend die spaetere **R**-/**C**-Interpretation nur ueber die Bruecke erfolgt; ausserdem ist auditierbar gemacht, welche Aussagen universell über `SubstrateClass` laufen und welche stärkere Voraussetzungen benötigen
S3	`ToC/Contract`, `ToC/Addressing`, `ToC/IdealAddressEquiv`, `ToC/ConcreteIdeal`	der konkrete Referenzfall des IDEAL-ToC ist als Familie geschlossen und legal an Contract/Addressing/Equiv rückgebunden; deterministische Zusätze bleiben flankierend und werden nicht heimlich zum Generatorvertrag erklärt
S4	`ToC/Concrete`, `Finite/CutSpec`, `Finite/RegionCore`, `Finite/BoundaryPorts`, `Finite/Approximant`, optional `Finite/Selection`	Referenzlauf und Variationslauf laufen ab dem ersten realen `ToCStrong`-/Approximantenkern in denselben endlichen Box-Pfad ein; alle exponierten Carrier und Prädikate liefern die nötigen `Fintype`/`Decidable`-Instanzen; mindestens ein harter Typcheck für beide Familien belegt, dass kein Smoke-Test-Sonderpfad entsteht
S5	`Finite/DirichletLaplacian`, `DtN/DtN`	die beiden ersten Generatorengstellen sind geschlossen: `DirichletWeight` ist als legitimer Wurzelinput an eine explizite Policy-/Builderoberfläche gebunden, und `InteriorInverse` ist als zu internalisierende Solver-/Eliminationslogik klassifiziert, statt als lose externe Eingabe mitzulaufen

Phase	Betroffene Module	Muss am Phasenende wahr sein
S6	`DtN/DtNStabilized`, `Closure/ParameterClosure`, `Closure/RegularizationClosure`	die Shift-, Stabilisations- und Closure-Seite ist soweit legal möglich internalisiert; primitive Restparameter werden nicht mehr unkommentiert durch den Produktionspfad getragen, und `DtNStabilized` konsumiert die noetigen Shift-/Vergleichsgroessen ausschließlich aus der computablen Produktivseite von `MatrixNorms` statt aus einer analytischen **C**-Spiegelinstanz
S7a	`Coupling/GeneralizedDtN`, `Coupling/MultiSchur`	die Mehrsektor-Kopplung ist familiesicher geschärft; raw/stabilized bleibt explizit getrennt, und die operative Schur-Seite läuft ausschließlich auf legal stabilisierter Provenienz; insbesondere ist `interfaceInverse` als zu internalisierende P22-Logik klassifiziert und nicht als Wurzelinput
S7b	`Network/InfiniteCarrier`	stabile Ausschnitte und gerichtete Übergänge bleiben für Vergleiche sowie spätere Limesdiagnostik exportierbar; die späte Trägerstufe bleibt dabei strikt Consumer des bereits geschlossenen Netzpfads und nicht der Coupling-Phase selbst
S8	`Finite/SpectralDecomposition`	die endliche Spektralzerlegung des A-seitig erzeugten Dirichlet-Operators steht als proof-carrying Datenobjekt; spätere Pfeiler müssen weder Eigenwerte noch Projektoren als A-Nachtrag neu aufbauen, und der Generator hat eine explizite algebraische Spektralwurzel
S9	`Finite/StateSpace`, `Finite/MatrixExponential`, `Finite/GibbsStateSeed`, `Finite/UnitaryEvolution`, `Finite/DynamicsAdapter`, `Finite/ChannelSeed`	aus der endlichen Matrixseite sind positiver Kegel, Zustandsraum, Spur-/Normierungsoberfläche sowie thermische, Gibbs- und unitäre Seeds einschliesslich der Heisenberg-Konjugation als `StarAlgDynamics`-Vorstufe und eine algebraische kanalische Vorstufe bereitgestellt; spätere Pfeiler erhalten damit theoremisierte Axiomsamen statt roher Matrixarithmetik und muessen keine Kraus-/Choi-/CPTP-Basis neu aufbauen
S10a	`Network/SectorSysEnv`, `Network/RelativeEntropyFlow`, `Geometry/LCPMeasure`, `Geometry/Foliation`, `Geometry/SpacetimePaths`, `Network/DirectedLimit`; `Network/RegionNet`, `Network/SectorChannels` nur unverändert als legale Vorgänger	die nicht-kernblockierende Netzwerk-, Geometrie- und Limesvorstufe steht; IDEAL- und REAL-Provenienz bleiben getrennt sichtbar, und `DirectedLimit` exportiert eine PreNet-kompatible Übergangsfamilie, ohne bereits AQFT-Limessemantik zu behaupten
S10b	`PillarA/VariationAnalysis`, `PillarA/Diagnostics/SpectralDimension`, `PillarA/Diagnostics/SymmetryCutoffLimits`, `PillarA/Diagnostics/ScaleBreaking`, `PillarA/Diagnostics/ApproxSymmetry`, `PillarA/Update/MismatchSeed`, `PillarA/Update/TailEliminationSeed`, `PillarA/Update/UpdateStep`	die nicht-kernblockierende Diagnose-, Variations-, Scale-Breaking-, Symmetrie- und Update-Seed-Schicht steht; Spektraldimensionsfluss sowie Symmetrie-, Cutoff- und Limesdiagnostik sind jeweils explizit in IDEAL- und REAL-Seite getrennt A-seitig vorbereitet, und `VariationAnalysis` klassifiziert Theoreme nach Universalitäts-, Substrat-, Policy-, ϵ-/β- und Cutoff-Abhängigkeit
S11a	`Handoff/BInterfaceSeeds`, `Handoff/SectorExport`, `Handoff/Step1StrongCore`	die zielneutrale Kupplungs- und Verdichtungsstufe exportiert Generator-, algebraische, spektrale, zustandsräumliche, thermische, dynamische, kanalische und IDEAL/REAL-Diagnoseprovenienz einmalig nach außen; zugleich stehen konstruktive BInterface-Seeds aus den reifen A-Daten bereit, und `SectorExport` wird nicht doppelt in späteren Phasen angefasst
S11b	`Handoff/Step1MathData`, `Handoff/ABHandoffStrong`, `Handoff/PillarAStep1Closed`, `PillarA/Generator`	der produktive Root-to-Handoff-Generator steht; Referenzfall und beliebige legale Kombination aus Substratfamilie, `WeightPolicy`, `RegularizationPolicy`, sichtbarer β-Achse und Cutoff erzeugen über denselben Pfad ein echtes `PillarAStep1Closed`; zugleich ist das Computability-Budget nach Whitelist auditiert, die öffentliche Generatoroberfläche bleibt soweit wie möglich computable, und `ABHandoffStrong` exponiert den A-seitigen `StarAlgebraContract` nur als ausgehenden Instantiierungssatz
S12	alle betroffenen `Notation`-Dateien	der gesamte Strengthening-Pfad ist lesbar notiert: Referenzfamilien, algebraische Seeds, Gibbs-/Dynamik-/Update-Seeds, BInterface-Seeds, Generator, Directed-Limit und Diagnosemodule sind scoped und parameter-sichtbar, ohne Transporte oder Provenienz zu verdecken
S13	alle betroffenen `BuildAll`-Dateien, Root-Build	der Strengthening-Pfad ist vollständig in den aktiven Build eingehängt; keine zweite Schatten-Topologie und kein Rückgriff auf Archivpfade verbleiben
S14	Root-Build, Handoff-Audit zwischen den Pillars, finaler Generator- und Exportaudit	der Strengthening-Pfad ist produktiv rauchtesttauglich und zugleich streng pfeilerisoliert: spätere Pfeiler konsumieren nur die bereitgestellten Export- und Handoff-Flächen, ohne rückwirkend frühere Pfeiler zu reparieren oder algebraische A-Vorstufen neu zu implementieren

15 Handoff-Architektur und Pillar-Isolation zwischen den Pillars

- **Kein globales Mega-Handoff pro Pillar.**
- Pro Pillar eine stabile `Export`-Fläche als zielneutrale öffentliche Oberfläche; sie ist die Kupplung nach außen und nicht die interne Arbeitsoberfläche desselben Pfeilers.
- Konkrete gerichtete Handoffs nur on demand: `A_to_B`, `A_to_C`, `D_to_A` usw.
- Pillar A kennt Pillar B nicht als Consumerlogik; Pillar B kennt die interne Herstellungslogik von A nicht, sondern nur die kanonisch bereitgestellte Export- bzw. Handoff-Oberfläche. Dieselbe Asymmetrie gilt für alle späteren Pfeiler.
- Kein Pfeiler darf seine eigene Export- oder Handoff-Fläche intern als Ersatzpfad konsumieren. Dieselben Daten bleiben intern nur über die vorgelagerten Kernmodule erreichbar; dadurch wird Zirkularität zwischen Motor, Kupplung und nachfolgenden Getrieben verhindert.
- Nachfolgende Pfeiler dürfen fehlende Werkzeuge eines früheren Pfeilers nicht in ihrem eigenen Code nachbauen. Wenn B, C, D oder E A-seitige Diagnose-, Skalen-, Spektral-, Zustands-, thermische oder limesseitige Werkzeuge benötigen, gehören diese in den Strengthening-Pfad von A und nicht in spätere Consumer.
- Pillar A soll daher nicht nur rohe Daten, sondern – soweit aus seinen eigenen generierten Größen möglich – bereits theoremisierte Axiomsamen späterer Pfeiler liefern. B, C, D und E konsumieren diese Samen als öffentliche Kupplungsoberfläche, ohne die A-seitige Herkunftslogik neu zu implementieren.
- Backreaction ist eine eigene Gegenrichtung und nicht im Vorwärtshandoff zu verstecken. Eine Gegenrichtung darf frühere Pfeiler öffnen, aber nur kontrolliert über eigene Rückkanäle und nie durch impliziten Missbrauch des Vorwärtshandoffs.
- Struktur: `PillarX/Export/...`, `PillarY/Input/FromX/...`, `Handoffs/X_to_Y/...`.
- Das verbreiterte `ABHandoffStrong` ist nur der gerichtete Adapter `A_to_B`, nicht die globale Exportfläche von A. Die Exportfläche von A soll dabei so reichhaltig wie nötig bleiben, damit spätere Pfeiler keine A-seitigen Hilfen nacherfinden müssen, ohne dass A dadurch semantisch auf B zugeschnitten wird.
- Für flankierende Diagnosepfade sind IDEAL-seitige Referenzdaten und REAL-seitige Approximanten-/Cutoffdaten im Export ausdrücklich getrennt zu führen. Spätere Pfeiler dürfen diese beiden Provenienzschienen konsumieren und vergleichen, aber weder in A zurückspeisen noch durch eigene A-Rekonstruktionen ersetzen.
- Die in A sichtbare thermische Achse β wird im Handoff nur als offener Provenienz- und Variationsparameter exportiert. Ihre physikalische Schließung ist keine A-interne Aufgabe, sondern entsteht erst über die nachgelagerte Kette $B \to C \to D$ (KMS/Modularitaet, Backreaction, Fixpunkt-Closure).

16 Fortschreibung nach dem Rauchtest: Architekturfix P21/P22

- Der Rauchtest ist kein optionaler Nebentest, sondern ein harter Architekturprüfer des geared Hauptpfads. Sein Befund wird nicht durch Seitenimporte geheilt, sondern durch Präzisierung der bereits legalen Provenienzketten.
- Der zulässige Fix für die Kopplungsseite lautet daher nicht `MultiSchur` → `RegularizationClosureStrong`, sondern: `GeneralizedDtNStrong` bleibt strikt Consumer von `DtNStrong` und `SectorSplitStrong`,

trägt aber raw/stabilized als explizit getrennte, theorematisch gebundene Operatoroberflächen derselben legalen P21-Provenienz.

- `MultiSchurStrong` bleibt strikt Consumer von `GeneralizedDtNStrong`. Die operative Inversions- und Schur-Seite läuft ausschließlich auf dem stabilisierten Restriktionspfad; der rohe Pfad bleibt als Audit-/Diagnostikfläche mitgeführt und darf weder gelöscht noch semantisch als operative Ersatzoberfläche benutzt werden.
- Rückwirkend in P21 nachzutragende Hilfsstrukturen sind auf sektorielle Randträger, Restriktions-/Projektionsdaten sowie Konsistenzsätze zwischen raw und stabilized beschränkt. Interface-Inverse, Gluing und reduzierte Schur-Komposition bleiben ausschließlich P22-Fachlogik.
- `SectorExportStrong`, `Step1StrongCore` und `Step1MathDataStrong` sind kein Reparaturort für einen Coupling-Fehler. Sie dürfen die in P21/P22 bereits geschlossene raw/stabilized-Dualität nur theorematisch sichtbar projizieren und keinesfalls durch rückwirkende Auswahl eines "passenden" Operators neu semantisieren.
- Notation bleibt nach diesem Fix strikt explizit: parameter-sichtbar, rein lesbar und ohne semantisch mehrdeutige Einheitsnotation für den operativen Kopplungsoperator. Insbesondere darf die Aliasoberfläche nicht wieder verdecken, welcher Pfad raw und welcher stabilized ist.

17 Nicht verwenden / vermeiden

- Bool-/Audit-Logik als Pseudobeweis für mathematische Schließung.
- große `decide`/`native_decide`-Gates über freie oder halbfreie Ziele.
- lange `simp`/`simpa`-Verkettungen als Ersatz für definitorische oder termweise Beweise.
- möglichst kein `@[simp]` im aktiven Pfad; insbesondere keine pauschalen Wrapper- oder Projektions-Definitionen, deren automatische Aufklappung spätere Consumer-Beweise aufbläht.
- verallgemeinerte Entscheidungsziele mit freien Variablen im kritischen Pfad.
- bridge-lokale Produktionsbeweise, die fachlich in Builder-, Kern- oder Handoff-Module gehören.
- Placeholder-/Label-Ersatz für echte Objekte.
- rohe Gleichsetzung von abstraktem und adressiertem IDEAL_∞ ohne explizite strukturtreue Äquivalenz.
- frühe AQFT-/Modular-Stubs als aktive Träger des Schritt-1-Pfads.
- dauerhafte Doppelspur alt/neu.
- neue `noncomputable`-Produktionsdefinitionen im aktiven geared Hauptpfad ohne expliziten Unvermeidbarkeitsnachweis, ohne Prüfung einer computable Ersatzformulierung oder mit nicht-computable Consumer-/Exportoberfläche.
- selector-lokale Rekonstruktion von Branching aus rohen Sektorcarriern oder bloßen Aktivitätszählern, wenn `BranchingWitness`/`SelectedBranching` die kanonische proof-carrying Oberfläche bereits bereitstellen.

18 Gate-, Cutover- und Archivregel

- Die zwischenzeitliche Step-1-Gate-Stufe konsumierte nur `PillarAStep1Closed`; sie war nie eigener Produktionsort und liegt nach grünem P27 selbst in der Archivspur.

- Zu prüfen ist nicht nur, ob Einzelmodule stark sind, sondern ob die gesamte Komposition bis zur Abgangswelle kanonisch und by-pass-frei ist.
- Cutover-Regel: neues Modul bauen, Downstream umstellen, alten Pfad abhängen und anschließend als Regressionsorakel archivieren statt physisch zu löschen.
- `PillarAStep1Bridge.lean` blieb nur solange als Regressionsorakel nötig, bis `PillarAStep1Closed` samt zwischenzeitlicher dünner Gate-Stufe den aktiven Pfad vollständig ersetzt hatte; seit grünem P27 liegt die Brücke in `legacy_sources/CNNA_v0.100_unstable`.

19 Informative Überlegungen zur Spektraldimension (nicht Teil des abzuarbeitenden Plans)

Diese Überlegungen gehören *nicht* zum abzuarbeitenden Refactor-Plan. Im Strengthening werden sie lediglich als **flankierende A-seitige Diagnosevorbereitung** berücksichtigt, damit spätere Pfeiler keine Spektraldimensionswerkzeuge für A nachtragen müssen. Dabei ist die Analyse ausdrücklich in eine **IDEAL-Seite** und eine **REAL-Seite** zu trennen; gerade diese Trennung ist fachlich wertvoll, weil sonst Cutoff- und Fensterartefakte mit idealen Grenzaussagen vermischt würden.

- **IDEAL-Seite:** Ein späterer Lauf der Spektraldimension d_s benötigt eine cutoff-freie Skalenreferenz. Dafür ist es architektonisch günstig, dass das ToC-Substrat zuerst als idealer unendlicher Träger formuliert wird und die adressierte Präsentation über eine strikt strukturtreue Äquivalenz an den abstrakten Idealvertrag rückgebunden bleibt.
- **IDEAL-Seite:** Die adressierte Präsentation des Ideals soll nicht roh mit dem abstrakten IDEAL_∞ identifiziert werden; für spätere Analysen ist eine bidirektionale Transferbrücke wertvoll, weil manche Aussagen abstrakt und andere adress- bzw. frontiernah natürlicher formulierbar sind. Diese Äquivalenz ist zugleich die Stelle, an der spätere kanonische Identifikationen des "Box-Inhalts" gegen bloß präsentationslokale Benennungen abgesichert werden müssen.
- **REAL-Seite:** Endliche REAL-Approximanten mit Parametern wie Branching-Stärke, Fensterlage und $L_{\max}$ tragen die spätere operative Diagnose. Auf dieser Seite sind numerische oder halbformale Diagnostiken des UV-Cutoff-Bruchs und des Flows von d_s plausibel, gerade weil hier Cutoff- und Approximantenabhängigkeit explizit sichtbar bleiben.
- **REAL-seitige Vorstufe:** Die in S10b bereitgestellte Kontraktionsdiagnostik (`ContractionExperiment`) liefert eine natürliche REAL-seitige Vorstufe zur Spektraldimensionsanalyse: die Kontraktionsrate bestimmt das Skalierungsverhalten einer Zelle unter wachsender Umgebung und ist direkt mit der effektiven Dimensionalität verknüpft.
- **Verbindungsregel IDEAL/REAL:** Die intendierte Fixpunktintuition lautet nicht, dass zwei bereits gegebene Typen roh gleichgesetzt werden, sondern dass der gerichtete Grenzpfad der REAL-Approximanten das ideale ToC rekonstruiert. Erst über diese gerichtete Rekonstruktionslesart werden Aussagen wie $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$ sinnvoll; eine rohe Identifikation von Diagnosewerten beider Seiten wäre gerade kein zulässiger Schritt.
- Diese Spektraldimensionsanalyse ist ausdrücklich **kein** Bestandteil der hier abzuarbeitenden Refactor-Phasen. Sie darf den Kernpfad nicht erweitern, bevor P1–P6 ontisch sauber stehen.
- Die jetzige Planaufnahme hat nur den informativen Zweck, spätere Backreaction- oder Analysepfade vorzubereiten, ohne den strikten Refactor-Hauptstrang zu vermischen.

20 Informative Überlegungen zu Symmetrie-, Cutoff- und Limesdiagnostik (nicht Teil des abzuarbeitenden Plans)

Diese Überlegungen gehören ebenfalls *nicht* zum abzuarbeitenden Refactor-Plan. Im Strengthening werden sie nur als **flankierende A-seitige Diagnose- und Provenienzvorbereitung** mitgeführt, damit spätere Pfeiler auf kanonische Werkzeuge zugreifen können, ohne frühere Pfeiler nachträglich zu reparieren. Auch hier ist die Analyse ausdrücklich in **IDEAL-Seite** und **REAL-Seite** zu trennen; nur so bleibt klar, welche Symmetrie oder welcher Limes ein idealer Referenzbegriff ist und welche Effekte erst aus endlichen Ausschnitten entstehen.

- **IDEAL-Seite:** Für spätere Diagnostiken ist es sinnvoll, die Symmetrie- und Grenzreferenz des idealen ToC separat zu halten. Auf dieser Seite stehen cutoff-freie Selbstähnlichkeit, idealisierte rooted Referenzsymmetrien und der reine Grenzträger als Vergleichsfolie; dies soll im Refactor als Daten- und Provenienzfrage vorbereitet werden, nicht schon als fertiger physikalischer Satz.
- **REAL-Seite:** Zwischen root-zentrierten Ausschnitten ohne äußere Umgebung und Fenstern mit echter Umgebung/Komplementseite ist strikt zu unterscheiden. Auf dieser Seite sind Cutoff, Fensterlage und Umgebungspräsenz operative Daten endlicher Approximanten und keine bloßen Darstellungsartefakte.
- **REAL-Seite:** Es ist fachlich plausibel, dass ein endlicher REAL-Ausschnitt die exakte Selbstähnlichkeit des idealen ToC durch den Cutoff bricht, während eine vorhandene Umgebung/Komplementseite zusätzlich rooted Symmetrien eines zentrierten Ausschnitts brechen kann. Ob und in welcher Form dies als Satz gilt, ist jedoch eine spätere Analysefrage und kein gegenwärtiges Pflichtresultat des Kernpfads.
- **Verbindungsregel IDEAL/REAL:** Die im Plan verwendete Fixpunktintuition bleibt auch hier gerichtet zu lesen: ein wachsender REAL-Approximant soll nicht roh mit dem idealen ToC identifiziert werden, sondern über gerichtete Übergänge und stabile endliche Ausschnitte eine Rekonstruktionslesart von $\mathrm{REAL}_\infty \to \mathrm{IDEAL}_\infty$ tragen.
- Für spätere Analysepfade ist es daher wichtig, dass Export- und Trägerstufen IDEAL-seitige Grenzreferenz und REAL-seitige Cutoff-, Komplement- und Übergangsdaten explizit getrennt sichtbar halten. Nur so lässt sich später unterscheiden, ob eine beobachtete Asymmetrie oder Skalenbrechung aus der idealen Referenz, aus der Fensterlage, aus der Umgebung oder aus dem gewählten Cutoff stammt.
- Ob aus der diskreten Brechungsfreiheit des IDEAL kontinuierliche Raumzeitsymmetrien wie Lorentz- oder CPT-nahe Strukturen emergieren, ist eine separate Frage für Pillar D/E. A-seitig beweisbar ist nur die schwache, aber harte Aussage: im IDEAL existiert kein Brechungsmechanismus.
- Eine weitergehende Brücke zu quasilokalem Limes, globaler geschlossener Dynamik, Liouville-von-Neumann-Formulierungen oder operatoralgebraischen Typ-III-Fragen ist fachlich plausibel, gehört aber ausdrücklich nicht in die hier abzuarbeitenden Refactor-Phasen. Der frühe Pillar-A-Pfad soll diese Brücke lediglich vorbereiten, nicht schon schließen.

21 Zukünftige Arbeiten jenseits von Pillar A: Semantische Schärfung der Folgepfeiler

Die folgenden Punkte gehören *nicht* mehr in den abzuarbeitenden Strengthening-Hauptpfad von Pillar A. Sie sind als präzise Folgearbeiten jenseits von A einzuplanen, weil sie echte AQFT-, OQS-, Geometrie- oder Matter-Fachlogik tragen. Zugleich werden sie jetzt semantisch geschärft, damit der Übergang von A in spätere Pfeiler weder auf alte boundary-first Muster zurückfällt noch A-seitige Vorstufen später heimlich nachbaut. Maßgeblich ist dabei die Roadmap-Regel, dass A nur solche

Samen liefert, die aus A-generierten Größen derived-only erzeugbar sind, während spätere Pfeiler diese Samen auf ihrer eigenen Fachschicht weiterverarbeiten. Vgl. die Roadmap zum A→B-Handoff, zum modularen B-Kern, zur Kanal- und Backreaction-Lesart in C sowie zu den späteren Seeds für D/E.

21.1 Zukünftige Arbeiten für Pillar B: AQFT aus A-Samen statt Bright-Recovery

- Die generische AQFT-Basis – `StarAlgebra`, `State`, `LocalNet`, `StateNet`, `GNS`, `KMS`, `RelativeEntropy`, `QuasiLocal/*` und die Haag–Kastler-nahe Grundschicht – bleibt eigenständige zukünftige Arbeit in Pillar B. Sie wird nicht nach A vorgezogen, muss aber die in A bereitgestellten algebraischen, thermischen, dynamischen und BInterface-Seeds direkt konsumieren.
- Das Legacy-Material aus `BoundaryMatrix*` bleibt für B ausdrücklich Extraktions-, Recovery- und Gate-Quelle, aber nie wieder Kernsemantik. Generische algebraische Lemmata, C*- und Zustandsaussagen dürfen extrahiert werden; ToC-gebundene Bright-Semantik ist in einen späteren Recovery-/Vergleichsstrang zu verschieben.
- Die B-seitige Konsumption der A-Exportfläche muss über explizite Brückenmodule laufen: insbesondere ein späteres `Handoffs/A_to_B/StarAlgebraBridge` und analoge Adapter für `LocalAlgebraNet`, `StateNet`, `ChannelNet` und quasilokale Vervollständigung. Diese Brücken liegen absichtlich außerhalb von A und B, damit weder A B importiert noch B A-interne Logik rekonstruiert.
- Der modulare B-Kern – `ComplementGNS`, `SeparatingProperty`, `TomitaTakesakiData`, `ModularConjugation`, `ComplementKMS`, `BisognanoWichmannSeed` – bleibt zukünftige B-Fachlogik. A liefert dafür nur die endliche Operatoralgebra, Gibbs-/Dynamikseeds, gerichtete Limesvorstufen und die sektorisierte Exportfläche.
- Besonders zu vermeiden ist jeder Rückfall in die alte Richtung "BoundaryMatrix zuerst, Komplementnetz spaeter". Die Roadmap fordert explizit `ComplementLocalNet` und `InterfaceLocalNet` vor jeder Recovery. B hat daher aus A-Sektorstruktur und BInterface-Seeds ein echtes Komplementnetz zu bauen, statt den Bright-Rand erneut zum impliziten Zentrum zu machen.

21.2 Zukünftige Arbeiten für Pillar C: Kanäle, Tail-Elimination und Backreaction

- Pillar C bleibt die Fachschicht für sektorielle Kanäle, Stinespring-, Lindblad-, Semigruppen- und Nichtmarkov-Lagen. A liefert dafür nur die algebraischen Seeds, `RelativeEntropyFlow`, `SectorChannels`, `SectorSysEnv` sowie die neuen Update-/Mismatch-/Tail-Elimination-Seeds.
- Die Roadmap verlangt ausdrücklich, Tail-Elimination als Kanal neu zu lesen und Backreaction als abgeleitete, nicht frei postulierte Zusatzdynamik zu behandeln. Genau deshalb werden `MismatchSeed`, `TailEliminationSeed` und `UpdateStep` in A vorbereitet, waehrend ihre offene-System-Interpretation, semigruppenartige Spezialisierung und Nichtmarkov-Verallgemeinerung zukünftige C-Arbeit bleiben.
- C darf `DerivedSpacetime` nicht als fertige Ontologie abschließen. Die Roadmap ordnet diese Richtung explizit als Vorbereitungsobjekt für D ein: aus Kanalinfluss wird emergente Kausal- und Geometriestruktur. C bleibt daher Kanal-, Entropie- und Rückwirkungspfeiler; die echte Geometrieschließung ist D vorbehalten.
- Zu vermeiden ist jeder semantische Rückfall, bei dem der dunkle Sektor wieder nur als eliminierte Restumgebung erscheint. C hat die Interface-vermittelte Wechselwirkung kanalisch zu explizieren und aus A nur diejenigen Seeds zu konsumieren, die diesen Schritt vorbereiten.

21.3 Zukünftige Arbeiten für Pillar D und Pillar E: Closure, Geometrie, Symmetrie und Matter

- Die eigentliche physikalische Schließung der Parameter b, $L_{\max}$, β und sekundärer Regularisierungsterme liegt nach Roadmap nicht in A, sondern am D-Eingang. A macht deren Provenienz sichtbar; B/C liefern modulare und kanalische Daten; D schließt den Fixpunktkreis aus Backreaction, Zustandsdaten, Skalenwahl und effektiver IR-Dynamik.
- Seeds wie `BisognanoWichmannSeed`, `NuclearitySeed`, `DHRSectorsSeed` und `FieldAlgebraReconstructionSe` bleiben ausdruecklich zukünftige B/D/E-Arbeit. Dasselbe gilt für `RelativeCauchyEvolutionSeed`, `HadamardStateSeed` und `LocalWickPolynomials`. Im Strengthening-Plan dürfen sie nur als Folgearbeit semantisch vorbereitet, aber nicht als scheinbar A-seitig schon geschlossen behauptet werden.
- Für E gilt dieselbe Semantikregel: A liefert Träger-, Algebra-, Zustands-, Dynamik-, Diagnose- und Limesvorstufen; Charge-, Gauge-, Matter-, CPT-, Kramers- und Spin-Statistik-Schichten entstehen erst auf stabiler modularer und kanalischer Unterlage in späteren Pfeilern.
- Der Legacy-Bestand darf auch hier nur als Extraktions- und Vergleichsquelle dienen. Jede spätere Portierung hat das Fachmodulprinzip des Refactors beizubehalten: pro Modul eine zentrale Aussage bzw. Fachlogik, keine kreuz und quer vermischten Derived-/Gate-/Example-Sammelbäume.

21.4 Semantische Leitregel für den Legacy-Einsatz

Der Legacy-Pfad von REALOQS ist für CNNA wertvoll, aber semantisch zu entzentrieren. Aus ihm dürfen allgemeine Mathematik, Gates, Bright-Recovery-Vergleichsobjekte und spätere Seed-Kandidaten systematisch extrahiert werden. Nicht zulässig ist dagegen, Legacy-Strukturen wegen ihrer faktischen Verfügbarkeit erneut zum impliziten Zentrum der neuen Stammtheorie zu machen. Dies betrifft besonders alte AQFT-Derived-Dateien, Bright-spezifische Dynamikableitungen und A→B-Zirkularitäten wie `AQFTSubstrate`. Die zukünftigen Folgepfeiler haben den Legacy-Bestand deshalb nur noch unter den Disziplinen *Extraktion*, *Recovery*, *Gatevergleich* und *Regression* zu nutzen.