Complement-Netzarchitektur (CNNA) aus dem ToC

Ziel

Neues Lean-Projekt, in dem der **Tree-of-Cliques (ToC)** nicht mehr primär als Quelle eines hellen Rand-/Blocknetzes gelesen wird, sondern als **Generator einer abgeleiteten Komplement-Netzarchitektur** mit drei gekoppelten Sektoren:

- **Bright**: expliziter REAL-Approximant / beobachtbarer Patch
- **Dark**: organisierte Komplementfamilie (Trunk, Seitenäste, eigene UV-Tails)
- **Interface**: effektive Kopplungs-, Rückwirkungs-, Entropie- und Modularitätsstruktur zwischen Bright und Dark

Das Projekt ist **kein Fork mit minimalen Umbenennungen**, sondern eine **neue Architektur**, die nur die belastbaren Daten, Strukturen und Beweise des alten REAL-OQS-Blocks übernimmt.

Leitidee

Ausgangspunkt ist nicht mehr der binäre Split

- `V ≃ B ⊕ I`

sondern ein mehrsektoriger, branch-anchored Split des endlichen sichtbaren Approximanten gegen eine organisierte Komplementfamilie. Der ToC bleibt das kanonische Substrat, aber er wird semantisch neu gelesen:

- nicht nur als Quelle des `BoundaryMatrixLocalNet`
- sondern als Quelle von **Komplementfamilien**, **Generalized DtN**, **Komplementnetzen**, **Komplementzuständen**, **Interface-Kanälen** und daraus emergenter kinematischer/geometrischer Strukturen.

Architekturprinzipien

1. **Derived-only**
2. keine exogenen Geometrien
3. keine exogenen Feldvariablen
4. keine exogene Kausalität

5. spätere Strukturen müssen aus ToC + Deterministik + Komplementkopplung emergieren

6. **Sector-first statt Boundary-first**

7. heller Sektor bleibt wichtig, ist aber nicht mehr privilegiertes Endprodukt
8. dunkler Sektor wird als erste Klasse formalisiert

9. Interface wird eigener mathematischer Träger

10. **Komplementfamilie statt einzelnes Komplement**

11. nicht nur `S \ R`, sondern eine organisierte Familie komplementärer Sektoren

12. verschiedene Anker, IR/UV-Schnitte, Seitenprofilierungen, Observer-/Selection-bedingte Varianten

13. **Neue Minimalbasis**

14. das Projekt startet mit den kleinstmöglichen übernehmbaren ToC-Daten

15. alles AQFT-/OQS-/Spacetime-artige wird darauf neu aufgesetzt

16. **Parameter-Closure statt fixer Policy-Parameter**

17. freie Strukturparameter sollen soweit möglich in **abgeleitete Selektoren** überführt werden
18. insbesondere sind `b`, die Nichttrivialität `1 < b`, und die effektive Horizont-/IR-Größe nicht als bloße Eingaben zu behandeln
19. `L_max` soll im neuen Stamm nicht mehr als ontischer Grundparameter erscheinen, sondern als **dynamische Horizontgröße** bzw. Backreaction-Fixpunkt

20. numerische Stabilisierungen der Dirichlet-/DtN-Schicht sind methodisch **sekundär** und sollen idealerweise eliminiert oder als abgeleitete Restgrößen interpretiert werden

21. **Mehrfachbeschreibung statt monolithischer Oberflächenform**

22. verschiedene emergente QFT-Sektoren können prinzipiell **dieselbe tiefere CNNA-Struktur** in unterschiedlichen Regimen beschreiben
23. der methodische Fokus liegt daher nicht auf „Strings als Fundament“, sondern auf **Dualitäts-Dictionaries**, **Glueing/Funktorialität**, **Defekten/Interfaces** und möglichen **kategorialen Äquivalenzen** zwischen emergenten Sektoren

24. stringtheorie-nahe Mathematik ist für CNNA vor allem dort relevant, wo sie präzise Beziehungen zwischen verschiedenen Oberflächenbeschreibungen derselben Physik organisiert

25. **Keine generischen Platzhalter auf dem aktiven Kernpfad**

26. generische Signaturen (`Region`, `Alg`, `State`, freie Skalare, freie Cutoffs, freie Hintergrunddaten, freie Boundary-State-Spaces, freie Dictionaries) sind nur als **Scaffold-Hüllen** erlaubt
27. auf dem **aktiven derived-only-Pfad** muss jedes Objekt eine explizite Herkunft aus ToC, Bright/Dark/Interface, DtN-/Schur-Elimination, Zustandsrestriktion oder Backreaction besitzen
28. was diese Herkunft nicht besitzt, bleibt als `Seed`, `Hook`, `Window` oder `Meta` **außerhalb** des kritischen Pfads

---

Neues Projekt: vorgeschlagener Name

- `CNNA` = Complement Net Architecture

- alternativ: `REALOQS_Complement`, `ToCComplementAQFT`, `DarkBrightInterface`

Empfehlung: `CNNA` als bewusst neuer Stamm.

I. Zu übernehmende Altbausteine

Nur belastbare, strukturstabile Module werden übernommen oder inhaltlich portiert.

A. Aus Pillar A

übernehmen / inhaltlich portieren

- `Core/Approximant`
- `Core/RegionCore`
- `Core/RegionNet`
- `Core/InfiniteCarrier`
- `Core/Selection`
- `Core/BInterfaces/LocalAlgebraNet`
- `Core/BInterfaces/StateNet`
- `Core/BInterfaces/ChannelNet`
- `OQS/DtN`
- `OQS/DtNStabilized`
- `Update/TailEliminationDtN`
- `Update/TailEliminationDtNStabilized`
- `OQS/SysEnv`
- `Core/TailEliminationCoherence`
- `Update/TwoStageApprox`
- ToC-spezifische Datenstrukturen aus `Core/ToC` und `Ideal/TreeOfCliques`

nur als Referenz, nicht als direkte Zielarchitektur

- rein boundary-zentrierte Exportpfade
- reine Harness-/Example-Pfade
- legacy hooks ohne tragende neue Funktion

B. Aus Pillar B

übernehmen / inhaltlich portieren

- `AQFT/StarAlgebra`
- `AQFT/State`
- `AQFT/StateNet`
- `AQFT/LocalNet`
- `AQFT/GNS`
- `AQFT/KMS`
- `AQFT/RelativeEntropy`
- `AQFT/QuasiLocal/*`
- relevante Haag-Kastler-Basis

- relevante Gates für Isotonie, Lokalität, Split, StateRestriction

nur teilweise portieren

- `BoundaryMatrix*`-Dateien nur soweit sie echte allgemeine Algebra-/Netztechnik enthalten
- alles ToC-gebundene an bright boundary wird in CNNA **neu als Spezialfall** aufgebaut

C. Aus Pillar C

übernehmen / inhaltlich portieren

- `OQS/Channel`
- `OQS/Stinespring`
- `OQS/NonMarkov`
- `Integration/TailElimAsChannel`
- `Integration/DerivedSpacetime`
- `Integration/Locality`

II. Was ausdrücklich nicht direkt übernommen werden soll

1. boundary-first Dateinamen als neues Zentrum
2. Meta-Dateien, die rein den alten Closure-Pfad auditieren
3. ad hoc entstandene Beispielpfade als Architekturgrundlage
4. alte Begriffsführung, in der Komplement/Umgebung nur „eliminiert“ und nicht strukturell organisiert wird

---

Derived-only-Audit: generische Inhalte und Rückbindungsregeln

Der aktuelle Canvas enthält bewusst einige **generische Gerüstbegriffe**, damit der Projektstamm aufgebaut werden kann. Für den **strikten derived-only-Pfad** müssen diese Begriffe jedoch schrittweise entgenerisiert werden.

A. Was im Canvas derzeit noch generisch ist

1. **Scaffold-Signaturen**
2. `Region`, `Alg`, `State`, `HasGeneralizedDtN`, freie `Prop`-Felder, freie Bool-/Scalar-Träger
3. diese sind als frühe Lean-Hüllen legitim, aber nicht als aktive Kerntheorie
4. **numerische Ergebniscontainer**
5. `EffectiveLambda`, `HorizonLevel`, `SpectralDimensionFlow` sind im Scaffold noch reine Container
6. derived-only verlangt hier eine Definition **aus** operatoriellen, spektralen oder zustandsbasierten Daten

7. **späte Übersetzungsbegriffe**

8. `BoundaryStates`, `FunctorialGlue`, `DualityDictionary`, `BulkBoundaryDictionary`, `ConformalWindow`

9. diese dürfen nicht axiomatisch frei beginnen, sondern nur aus schon ausgerichteten Observablen-, Regionen- und Zustandsstrukturen erwachsen

10. **AS-nahe Seeds**

11. `BackgroundIndependentCoarseGraining`, `RunningBoundaryData`, `TadpoleSeed`, `SplitSymmetrySeed`, `SectorRGSeed`, `TheorySpaceSeed`
12. diese sind methodische Explikationen und keine primitives des Kerns

B. Derived-only-Rückbindung pro Begriffsfamilie

Begriffsfamilie	im Scaffold generisch	auf strict derived-only zu ersetzen durch
Regionen	freier `Region`-Typ	aus `Approximant`, `BranchPatch`, `ComplementSectorFamily`, ToC-Pfad- und Zellstruktur abgeleitete Regionen
Algebren / Netze	freier `Alg`-Träger	aus blockweisen Operator-/Matrix-/C*-Daten der sektoriellen DtN-/Komplementstruktur abgeleitete Algebrazuweisung
Zustände	freier `State`-Träger	Restriktionen, Pushforwards oder Balance-Zustände aus Bright/Dark/Interface
Geometrie	freie Raumzeit-/Kausalbegriffe	aus `ComplementInfluence`, `DerivedSpacetime`, `InterfaceCausality`, Backreaction
Skalare (`Λ`, Horizon)	freie numerische Container	aus Interface-Balance, Spektraldaten, Entropiefluss und Fixpunktbedingungen abgeleitete Größen
Dualitätsbegriffe	freie Dictionaries / Boundary States	nur nach Observablenabgleich, Glueing und Zustandskompatibilität
RG-/AS-Begriffe	freie coarse-graining-/RG-Seeds	nur als spätere Explikation bereits vorhandener derived Strukturen

C. Strikte Regel für aktive Dateien

Eine Datei darf nur dann auf dem **kritischen Pfad** stehen, wenn ihre zentralen Definitionen einen expliziten Herkunftspfad besitzen, der auf eine der folgenden Quellen zurückführt:

- ToC-Zell-/Adress-/Elternpfadstruktur
- `Approximant` / `BranchPatch`
- `ComplementSectorFamily` / `SectorSplit`
- DtN-/Schur-/Tail-Elimination
- Bright/Dark/Interface-Zustandsrestriktion
- Backreaction-/Fixpunktkonstruktion

- daraus abgeleitete spektrale oder entropische Größen

Alles andere ist als `Seed`, `Hook`, `Window`, `Meta` oder `Example` zu markieren.

D. Neue Derived-only-Meta-Regeln

1. **kein freier Beobachter**
2. Anker/Selection müssen aus ToC und definierter Auswahlstruktur kommen, nicht aus exogenem Beobachterbegriff
3. **kein freier Cutoff**
4. `L_max` darf auf dem aktiven Pfad nur noch als temporärer Scaffold-Name vorkommen; die echte Zielgröße ist `HorizonLevel`
5. **kein freier Zustand**
6. Zustände müssen aus Bright/Dark/Interface-Relationen, Restriktionen oder Balancen kommen
7. **kein freies Dictionary**
8. Dualität/Dictionary erst nach konkretem Observablen- und Zustandsabgleich
9. **kein freier RG-Fluss**
10. AS-Begriffe dürfen nicht stärker sein als die tatsächlich abgeleitete Closure-Struktur

E. Dateinamenkonvention für nicht-abgeleitete Module

Damit der Modulbaum selbst den derived-only-Status sichtbar macht, gilt:

- `...Scaffold.lean`
- rein strukturelle Lean-Hülle
- darf auf dem Build-Pfad stehen, aber nicht als fertige Kerndefinition gelesen werden
- `...Seed.lean`
- methodischer oder später auszudefinierender Begriff
- darf Orientierung geben, aber keinen unverdienten ontischen Status bekommen
- `...Window.lean`
- spätes Spezialfenster / spätes Regime
- bewusst nicht Fundament
- **unsuffixed Modulnamen**

- nur für Begriffe, die im aktiven Pfad bereits als genuinely derived Kernobjekte gedacht sind

F. De-seeding-Regel

Ein Modul darf ein `Seed`-/`Scaffold`-Suffix erst dann verlieren, wenn:

1. eine `SourceMap` vorliegt,
2. keine freien Cutoffs/Zustände/Dictionaries mehr im zentralen Definitionskern verbleiben,
3. das Modul ein Derived-only-Gate auf dem kritischen Pfad besteht.

G. Seed → Zielmodul-Übersicht

Die folgende Tabelle ist die operative De-seeding-Landkarte. Sie legt fest, welche derzeit generischen Module später in echte derived Kernmodule überführt werden sollen und welche bewusst Seeds/Windows bleiben.

aktuelles Modul	Zielmodul	Bedingung für De-seeding
`GeneralizedDtNScaffold`	`GeneralizedDtN`	blockweise DtN-/Schur-Definition aus `ComplementSectorFamily` + `SectorSplit`
`ComplementLocalNetScaffold`	`ComplementLocalNet`	Regionen und Algebrazuweisung aus ToC/Komplement/DtN explizit abgeleitet
`ComplementStateNetScaffold`	`ComplementStateNet`	Zustandstransfer/Restriktion aus Bright/Dark/Interface konkret definiert
`EffectiveLambdaSeed`	`EffectiveLambda`	aus Interface-Balance / Entropie / Spektral- oder Fixpunktdaten konstruiert
`HorizonLevelSeed`	`HorizonLevel`	nicht mehr primitiver Cutoff, sondern aus Backreaction/Fixpunkt abgeleitet
`SpectralDimensionFlowSeed`	`SpectralDimensionFlow`	aus abgeleiteten Heat-/Semigruppen-/Spektraldaten gewonnen
`DerivedStressTensorSeed`	`DerivedStressTensor`	aus lokaler Zustands-/Entropie-/Interface-Balance gewonnen
`EntanglementEquilibriumSeed`	`EntanglementEquilibrium`	nur nach lokaler Horizon-/Entropie-/Volumenbilanz auf D

aktuelles Modul	Zielmodul	Bedingung für De-seeding
`EinsteinLimitSeed`	`EinsteinLimit`	nur nach geschlossener Jacobson-Route in der IR-Grenze
`InducedGravitySeed`	`InducedGravity`	nur nach präziser Materiesektor-Integration/ Elimination
`DualityDictionarySeed`	`DualityDictionary`	Observablen-, Zustands- und Dynamikerhalt explizit nachgewiesen
`BoundaryStateSpaceSeed`	`BoundaryStateSpace`	Randzustände aus derived Netz-/Zustandsstruktur komponierbar
`Defects/BasicSeed`	`Defects/Basic`	Defektbegriff aus konkreten Interface-/Sektorübergängen gewonnen
`Defects/FusionSeed`	`Defects/Fusion`	nur nach wohldefinierter Defektkomposition/Fusion
`FunctorialGlueSeed`	`FunctorialGlue`	Regionen-/Zustands-/ Randkomposition derived definiert
`BulkBoundaryDictionarySeed`	`BulkBoundaryDictionary`	bulk/boundary-Zuordnung nicht mehr heuristisch, sondern datenverträglich
`BackgroundIndependentCoarseGrainingSeed`	kein unmittelbares unsuffixed Pflichtziel	erst nach Closure und klarer skalenabhängiger Geometrieselektion
`RunningBoundaryDataSeed`	kein unmittelbares unsuffixed Pflichtziel	erst nach expliziter Unterscheidung Bulk vs. Interface-Running
`TadpoleSeed`	optional `SelfConsistentGeometry` oder analog	erst wenn `BackreactionFixedPoint` zu echter Tadpole-/ Selbstkonsistenzgleichung erweitert ist
`SplitSymmetrySeed`	kein unmittelbares unsuffixed Pflichtziel	erst nach präziser Hintergrund-/Interface-Rollentrennung
`SectorRGSeed`	kein unmittelbares unsuffixed Pflichtziel	erst nach expliziter running-effective-data-Formulierung

aktuelles Modul	Zielmodul	Bedingung für De-seeding
`TheorySpaceSeed`	kein unmittelbares unsuffixed Pflichtziel	erst nach Begriffsbildung relevanter/irrelevanter Richtungen
`CutoffModeCountingSeed`	optional `CutoffModeCounting`	nur nach derived Spektral-/Modenzählstruktur
`SpectralCutoffMapSeed`	optional `SpectralCutoffMap`	nur nach expliziter Verträglichkeit mit `SpectralDimensionFlow` und `HorizonLevel`
`FLRWSectorWindow`	bleibt `Window`	spätes kosmologisches Spezialregime, kein Kernmodul
`ConformalWindow`	bleibt `Window`	spätes Spezialregime, kein Kernmodul
`MinkowskiWindow`	bleibt `Window`	spätes Spezialregime, kein Kernmodul

H. Operative De-seeding-Reihenfolge

Die bevorzugte Reihenfolge ist:

1. `GeneralizedDtNScaffold` → `GeneralizedDtN`
2. `ComplementLocalNetScaffold` → `ComplementLocalNet`
3. `ComplementStateNetScaffold` → `ComplementStateNet`
4. `HorizonLevelSeed` / `EffectiveLambdaSeed` / `SpectralDimensionFlowSeed`
5. `DerivedStressTensorSeed` / `EntanglementEquilibriumSeed` / `EinsteinLimitSeed`
6. erst danach Glueing-/Boundary-/Dualitäts-Seeds
7. AS-nahe Seeds bleiben bis nach Closure grundsätzlich Seeds

I. Harte Regel für Windows

`...Window`-Module verlieren ihr Suffix grundsätzlich **nicht**, außer es wird explizit beschlossen, dass ein bisher spätes Spezialfenster selbst Teil des derived Kerns geworden ist. Die Default-Annahme ist also:

- `ConformalWindow` bleibt `Window`
- `MinkowskiWindow` bleibt `Window`

III. Neuer Kernformalismus

1. BranchPatch

Ein endlicher sichtbarer Patch relativ zu einem ToC-Anker.

```
structure BranchPatch (Ω : Type u) where
  anchor      : Ω
  bright      : Finset Ω
  uvCut       : Nat
  irCut       : Nat
  sideProfile : Finset Ω
```

Bedeutung

- `anchor` : branch-/observer-bezogener Anker
- `bright` : sichtbarer, explizit approximierter Bereich
- `uvCut` : Tiefe des lokalen sichtbaren Bereichs
- `irCut` : coarse/trunk-Abschneidung
- `sideProfile` : kontrollierte seitliche Sektorwahl

2. ComplementSectorFamily

Organisierte Beschreibung des dunklen Sektors relativ zu einem `BranchPatch`.

```
structure ComplementSectorFamily (Ω : Type u) where
  patch        : BranchPatch Ω
  commonTrunk  : Finset Ω
  privateIR    : Finset Ω
  sideBranches : Finset (Finset Ω)
  ownUVTails   : Finset (Finset Ω)
```

Semantik

- `commonTrunk` : gemeinsam genutzter Vorfahrenstamm
- `privateIR` : patch-exklusive trunk-/coarse-Komponente
- `sideBranches` : exklusive Seitenäste der Umgebung
- `ownUVTails` : eigene UV-Tails dieser Seitenäste

3. SectorSplit

Mehrsektorige Zerlegung des Gesamtraums.

```
structure SectorSplit (V : Type u) where
  Bright      : Type u
  Trunk       : Type u
  Side        : Type u
  Tail        : Type u
  totalEquiv  : V ≃ (((Bright ⊕ Trunk) ⊕ Side) ⊕ Tail)
```

4. InterfaceKernel

Kodiert Rückwirkung, Schur-/DtN-Selbstenergie und Kopplungsdaten.

```
structure InterfaceKernel (𝕜 : Type u) (n : Nat) where
  selfEnergy      : Matrix (Fin n) (Fin n) 𝕜
  couplingWitness : Prop
  stabilized      : Bool
```

5. GeneralizedDtN

Elimination des organisierten dunklen Sektors gegen den hellen Sektor.

```
def generalizedDtN := ...
```

Zentrale Sätze: - `generalizedDtN_wellDefined` - `iteratedSchur_eq_globalSchur` - `interface_selfEnergy_respects_sector_split` - `blockDiagonal_dark_implies_additive_interface`

6. ComplementLocalNet

Lokales Netz auf dunklen oder gemischten Regionen.

```
structure ComplementLocalNet (Region Alg : Type u) where
  toAlg     : Region → Alg
  isotony   : ...
  locality  : ...
```

7. InterfaceStateNet / ComplementStateNet

Zustandsfamilien auf Bright, Dark und Interface.

8. ComplementGNS / ComplementKMS

Repräsentations- und Thermostruktur des dunklen/complementären Netzes.

IV. Neuer Modulbaum

Top-Level

- `CNNA/Basic.lean`
- `CNNA/BuildAll.lean`
- `CNNA/ImportPolicy.md`

- `CNNA/CStarFoundationPolicy.md`
- `CNNA/ArchitecturePolicy.md`
- `CNNA/Interface.lean`
- `CNNA/PillarA.lean`
- `CNNA/PillarB.lean`
- `CNNA/PillarC.lean`
- `CNNA/PillarD.lean`
- `CNNA/All.lean`

Pillar-Subroots

- `CNNA/PillarA/Interface.lean`
- `CNNA/PillarA/All.lean`
- `CNNA/PillarA/BuildAll.lean`
- `CNNA/PillarB/Interface.lean`
- `CNNA/PillarB/BuildAll.lean`
- `CNNA/PillarC/Interface.lean`
- `CNNA/PillarC/BuildAll.lean`
- `CNNA/PillarC/OQS.lean`
- `CNNA/PillarC/Integration.lean`
- `CNNA/PillarD/Interface.lean`
- `CNNA/PillarD/BuildAll.lean`

Audit-Korrektur zur Vollständigkeit

Die erste CNNA-Fassung war **konzeptionell richtig**, aber **nicht vollständig explizit** in Bezug auf den tatsächlichen REAL-OQS-Bestand. Für Vollständigkeit müssen im neuen Projekt ausdrücklich geplant sein:

1. **Top-Level-Pillar-Aggregatoren**
2. analog zu `REALOQS/PillarA.lean`, `REALOQS/PillarB.lean`, `REALOQS/PillarC.lean`
3. in CNNA zusätzlich ein expliziter `CNNA/PillarD.lean`
4. **Pillar-spezifische Build-/All-Dateien**
5. `CNNA/PillarA/All.lean`
6. `CNNA/PillarA/BuildAll.lean`
7. `CNNA/PillarB/BuildAll.lean`
8. `CNNA/PillarC/BuildAll.lean`
9. `CNNA/PillarD/BuildAll.lean`
10. **explizite Pillar-D-Struktur**
11. die erste Fassung hatte D nur implizit unter `Integration/*`
12. für die Matrix ist ein echter D-Strang nötig: emergente Raumzeit, Kausalstruktur, lokale Kovarianz, spätere Spezialfenster

13. **Grunddateien aus Pillar A, die nicht stillschweigend verloren gehen dürfen**

14. `Interfaces`, `AQFTHandoff`, `Determinism`, `TailInterface`, `BoundaryPorts`, `DirichletLaplacian`, `DirichletLaplacianBridge`

15. außerdem Hooks/Seeds für Kinematik und Symmetrie, soweit sie auf dem kritischen Pfad bleiben

16. **Grunddateien aus Pillar B, die über reine AQFT-Basis hinaus relevant sind**

17. `CStarNorm`, `CStarState`, `CStarKMSState`
18. `GlobalCone`, `TimeOrientation`, `TimeReversal`
19. `HaagKastler/Support`, `HaagKastler/SupportFull`, `HaagKastler/SplitProperty`

20. Derived-Boundary-Matrix-Dateien als zu emulierender Bright-Spezialfall

21. **Grunddateien aus Pillar C, die explizit im Modulbaum erscheinen sollten**

22. `Finite`, `Lindblad`, `Semigroup`

23. nicht nur `Channel`, `Stinespring`, `NonMarkov`

24. **Recoverability-Grenze**

25. Bright-/Boundary-first wird nicht Architekturzentrum
26. aber die vollständige Rückgewinnung des alten aktiven Bright-Pfads muss als geplante Beweiskette explizit verankert sein

Die folgende Struktur ist daher als **korrigierte Vollversion** zu lesen.

1. Core

- `CNNA/Core/Approximant.lean`
- `CNNA/Core/RegionCore.lean`
- `CNNA/Core/RegionNet.lean`
- `CNNA/Core/InfiniteCarrier.lean`
- `CNNA/Core/Selection.lean`
- `CNNA/Core/Interfaces.lean`
- `CNNA/Core/AQFTHandoff.lean`
- `CNNA/Core/Determinism.lean`
- `CNNA/Core/TailInterface.lean`
- `CNNA/Core/BoundaryPorts.lean`
- `CNNA/Core/DirichletLaplacian.lean`
- `CNNA/Core/DirichletLaplacianBridge.lean`
- `CNNA/Core/BranchPatch.lean`
- `CNNA/Core/ComplementSectorFamily.lean`
- `CNNA/Core/SectorSplit.lean`
- `CNNA/Core/InterfaceKernel.lean`
- `CNNA/Core/BranchingWitness.lean`
- `CNNA/Core/ParameterClosure.lean`
- `CNNA/Core/Exports.lean`

2. ToC

- `CNNA/ToC/Substrate.lean`
- `CNNA/ToC/Addresses.lean`
- `CNNA/ToC/ParentPath.lean`
- `CNNA/ToC/Cells.lean`
- `CNNA/ToC/BranchPatchInstances.lean`
- `CNNA/ToC/ComplementSectorInstances.lean`
- `CNNA/ToC/Examples.lean`

3. OQS

- `CNNA/OQS/DtN.lean`
- `CNNA/OQS/DtNStabilized.lean`
- `CNNA/OQS/GeneralizedDtN.lean`
- `CNNA/OQS/MultiSchur.lean`
- `CNNA/OQS/SectorChannels.lean`
- `CNNA/OQS/SectorSysEnv.lean`
- `CNNA/OQS/Finite.lean`
- `CNNA/OQS/Lindblad.lean`
- `CNNA/OQS/Semigroup.lean`
- `CNNA/OQS/RelativeEntropyFlow.lean`
- `CNNA/OQS/Backreaction.lean`
- `CNNA/OQS/BackreactionFixedPoint.lean`
- `CNNA/OQS/RegularizationClosure.lean`

4. AQFT

- `CNNA/AQFT/StarAlgebra.lean`
- `CNNA/AQFT/CStarNorm.lean`
- `CNNA/AQFT/CStarState.lean`
- `CNNA/AQFT/CStarKMSState.lean`
- `CNNA/AQFT/State.lean`
- `CNNA/AQFT/StateNet.lean`
- `CNNA/AQFT/LocalNet.lean`
- `CNNA/AQFT/ComplementLocalNet.lean`
- `CNNA/AQFT/ComplementStateNet.lean`
- `CNNA/AQFT/InterfaceLocalNet.lean`
- `CNNA/AQFT/GNS.lean`
- `CNNA/AQFT/ComplementGNS.lean`
- `CNNA/AQFT/KMS.lean`
- `CNNA/AQFT/ComplementKMS.lean`
- `CNNA/AQFT/RelativeEntropy.lean`
- `CNNA/AQFT/Exports.lean`
- `CNNA/AQFT/GlobalCone.lean`
- `CNNA/AQFT/TimeOrientation.lean`
- `CNNA/AQFT/TimeReversal.lean`
- `CNNA/AQFT/DualityDictionarySeed.lean`
- `CNNA/AQFT/BoundaryStateSpaceSeed.lean`

- `CNNA/AQFT/Defects/BasicSeed.lean`
- `CNNA/AQFT/Defects/FusionSeed.lean`
- `CNNA/AQFT/QuasiLocal/Basic.lean`
- `CNNA/AQFT/QuasiLocal/ComplementClosure.lean`
- `CNNA/AQFT/HaagKastler/Basic.lean`
- `CNNA/AQFT/HaagKastler/Complement.lean`
- `CNNA/AQFT/HaagKastler/Support.lean`
- `CNNA/AQFT/HaagKastler/SupportFull.lean`
- `CNNA/AQFT/HaagKastler/SplitProperty.lean`

5. Integration

- `CNNA/Integration/TailElimAsChannel.lean`
- `CNNA/Integration/DerivedSpacetime.lean`
- `CNNA/Integration/ComplementInfluence.lean`
- `CNNA/Integration/SpacetimeOfComplementFamily.lean`
- `CNNA/Integration/LocalCovarianceSeed.lean`
- `CNNA/Integration/EffectiveLambdaSeed.lean`
- `CNNA/Integration/HorizonLevelSeed.lean`
- `CNNA/Integration/FunctorialGlueSeed.lean`
- `CNNA/Integration/BulkBoundaryDictionarySeed.lean`

5a. PillarD

- `CNNA/PillarD/BuildAll.lean`
- `CNNA/PillarD/DerivedSpacetime.lean`
- `CNNA/PillarD/ComplementGeometry.lean`
- `CNNA/PillarD/InterfaceCausality.lean`
- `CNNA/PillarD/LocalCovariance.lean`
- `CNNA/PillarD/SpectralDimensionFlowSeed.lean`
- `CNNA/PillarD/EntanglementEquilibriumSeed.lean`
- `CNNA/PillarD/DerivedStressTensorSeed.lean`
- `CNNA/PillarD/EinsteinLimitSeed.lean`
- `CNNA/PillarD/InducedGravitySeed.lean`
- `CNNA/PillarD/HorizonThermodynamicsSeed.lean`
- `CNNA/PillarD/FLRWSectorWindow.lean`
- `CNNA/PillarD/ConformalWindow.lean`
- `CNNA/PillarD/MinkowskiWindow.lean`

6. Gates

- `CNNA/Gates/BranchPatch.lean`
- `CNNA/Gates/ComplementSectorFamily.lean`
- `CNNA/Gates/GeneralizedDtN.lean`
- `CNNA/Gates/ComplementNet.lean`
- `CNNA/Gates/ComplementState.lean`
- `CNNA/Gates/ComplementKMS.lean`
- `CNNA/Gates/InterfaceEntropy.lean`
- `CNNA/Gates/DerivedSpacetime.lean`

- `CNNA/Gates/ParameterClosure.lean`
- `CNNA/Gates/DerivedOnlyNaming.lean`
- `CNNA/Gates/BrightRecovery.lean`
- `CNNA/Gates/BuildAll.lean`

7. Meta

- `CNNA/Meta/OpenPoints.lean`
- `CNNA/Meta/VacuityAudit.lean`
- `CNNA/Meta/CriticalPath.lean`
- `CNNA/Meta/NoLegacyInCriticalPath.lean`
- `CNNA/Meta/BrightDarkInterfaceSurface.lean`
- `CNNA/Meta/DerivedOnlyAudit.lean`
- `CNNA/Meta/SourceMap.lean`
- `CNNA/Meta/NoFreeParametersOnCriticalPath.lean`

8. Examples

- `CNNA/Examples/BuildAll.lean`
- `CNNA/Examples/ToC_BranchPatch.lean`
- `CNNA/Examples/ToC_ComplementFamily.lean`
- `CNNA/Examples/ToC_GeneralizedDtN.lean`
- `CNNA/Examples/ToC_ComplementNet.lean`
- `CNNA/Examples/ToC_InterfaceEntropy.lean`
- `CNNA/Examples/ToC_DerivedSpacetime.lean`

V. Alt → Neu: Migrationsabbildung

direkte Portierung

Alt	Neu
`PillarA/Core/Approximant`	`CNNA/Core/Approximant`
`PillarA/Core/RegionCore`	`CNNA/Core/RegionCore`
`PillarA/Core/RegionNet`	`CNNA/Core/RegionNet`
`PillarA/Core/InfiniteCarrier`	`CNNA/Core/InfiniteCarrier`
`PillarA/Core/Selection`	`CNNA/Core/Selection`
`PillarA/OQS/DtN`	`CNNA/OQS/DtN`
`PillarA/OQS/DtNStabilized`	`CNNA/OQS/DtNStabilized`
`PillarA/Update/TailEliminationDtN`	`CNNA/Integration/TailElimAsChannel` + `CNNA/OQS/Backreaction`
`PillarA/Core/BInterfaces/*`	`CNNA/Core/*` oder `CNNA/AQFT/*` Interfaces

Alt	Neu
`PillarB/AQFT/LocalNet`	`CNNA/AQFT/LocalNet`
`PillarB/AQFT/StateNet`	`CNNA/AQFT/StateNet`
`PillarB/AQFT/GNS`	`CNNA/AQFT/GNS`
`PillarB/AQFT/KMS`	`CNNA/AQFT/KMS`
`PillarB/AQFT/RelativeEntropy`	`CNNA/AQFT/RelativeEntropy`
`PillarC/OQS/Channel`	`CNNA/OQS/SectorChannels`
`PillarC/OQS/Stinespring`	`CNNA/OQS/SectorChannels` / `CNNA/AQFT/ComplementGNS` flankierend
`PillarC/Integration/DerivedSpacetime`	`CNNA/Integration/DerivedSpacetime`

gezielte Neuinterpretation statt 1:1-Port

Alt	Neu
`BoundaryMatrixLocalNet`	`ComplementLocalNet` + `InterfaceLocalNet` + Bright-Spezialfall
`BoundaryMatrixStateNet`	`ComplementStateNet` + Bright-Spezialfall
`BoundaryMatrixGNS`	`ComplementGNS` + Bright-Spezialfall
`BoundaryMatrixKMSFromL`	`ComplementKMS` + Interface-induzierte KMS-Pfade
`SysEnv`	`SectorSysEnv`
`SplitInvariants`	`SectorSplit` + `ComplementSectorFamily`

explizit nicht portieren

- alte Closure-/Audit-Meta-Dateien mit boundary-first Fokus
- example-only Harness-Strukturen ohne neue Kernfunktion
- veraltete triviale hooks auf kritischem Pfad

VI. Kritischer Pfad der Implementierung

Phasenprinzip

Die erste Fassung des kritischen Pfads war als **inhaltliche Makrostruktur** gut, aber mehrere Phasen waren für eine reale Lean-Umsetzung zu groß. Für die tatsächliche Arbeit gilt daher jetzt:

- jede Phase sollte **einen klaren Build-Zustand** hinterlassen,
- jede Phase sollte möglichst **nur eine neue begriffliche Last** einführen,

- große Theorieblöcke werden in **Mikrophasen** zerlegt,
- Recovery-/Spezialfenster werden **später** und klar getrennt angeschlossen.

Faustregel:

- **Makrophase** = fachlicher Themenblock
- **Mikrophase** = realistischer Lean-Arbeitsschritt mit engem Importkegel und klaren Gates

Prioritätenmarkierung

Zusätzlich zur zeitlichen Reihenfolge erhalten die Mikrophasen jetzt eine **Arbeitspriorität**:

- **[K1] kritisch zuerst**
- blockiert den weiteren Stamm direkt
- sollte vor parallelisierbaren Zweigen abgeschlossen sein
- betrifft Grundbegriffe, Importstabilität, Kernsignaturen und die ersten nichttrivialen mathematischen Kopplungen
- **[K2] früh, aber nach Stammstabilisierung**
- sinnvoll direkt nach den K1-Phasen
- baut auf stabilem Kern auf, erweitert aber noch nicht in Spezialfenster
- **[P] parallelisierbar**
- kann nach Erreichen klarer Gates in getrennten Arbeitssträngen laufen
- sollte nur auf bereits stabilen Signaturen aufsetzen
- **[S] bewusst spät / Spezialfenster**
- nicht blockierend für den Kern
- Recovery-, Conformal-, Wightman- oder ähnliche Spezialfenster
- erst sinnvoll, wenn die Kernarchitektur bereits trägt

Priorisierte Übersicht

K1 — kritisch zuerst

- 0a, 0b, 0c
- 1a, 1b
- 2a, 2b, 2c
- 3a, 3b, 3c
- 3.5a, 3.5b, 3.5c
- 4a, 4b
- 5a, 5b, 5c
- 7a
- 8a, 8b
- 8.5a

K2 — früh nach Stammstabilisierung

- 1c
- 4c, 4d
- 5d
- 6a, 6b
- 7b, 7c, 7d
- 8c, 8d, 8e, 8f
- 8.5b, 8.5c, 8.5d
- 8.6a, 8.6b

P — parallelisierbar nach klaren Gates

- 6c
- 8g
- 8.6c, 8.6d
- 8.75a, 8.75b, 8.75c, 8.75d
- Teile der Beispielpfade
- Teile der zusätzlichen Kinematik-/Symmetrie-Seeds aus Pillar A

S — bewusst spät / Spezialfenster

- 9a, 9b, 9c
- `ConformalWindow`
- spätere Wightman-/Vakuum-/Spektral-Fenster jenseits des Kernpfads
- vollständige Replay-Portierung tiefer Altbeispiele

Abhängigkeitsregeln

1. **Phase 3c ist die erste echte Kernschwelle.** Vor `3c` sollte nichts gestartet werden, das auf nichttrivialer DtN-/Schur-Kopplung beruht.

2. **Phase 5c ist die erste Netz/Zustands-Schwelle.** Erst nach `5c` sind dunkler Sektor, Interface und Zustandstransfer stabil genug, damit GNS/KMS oder Dualitätsstränge sinnvoll anschließen.

3. **Phase 7d und 8b bilden die OQS/Geometrie-Schwelle.** Erst danach ist Parameter-Closure mehr als reine Symbolik.

4. **Phase 8f ist die erste D-GR-Schwelle.** Erst nach `8f` ist in D mehr als bloße emergente Kinematik vorhanden; erst dann darf von einer Einstein-/Jacobson-artigen IR-Richtung gesprochen werden.

5. **Phase 3.5 ist die Derived-only-Signaturschwelle.** Vor `3.5c` dürfen generische Scaffold-Signaturen nicht in den aktiven AQFT-/Zustands-/Geometriepfad hineinwachsen.

6. **Phase 8.5d ist die Closure-Schwelle.** Erst nach `8.5d` ist der Weg frei für anspruchsvollere Mehrfachbeschreibungen, Bulk/Boundary-Dictionaries, AS-Kompatibilitätsstränge und spätere Spezialfenster.

7. **Phase 8.6 ist ein methodischer Explikationsstrang, kein Ersatz für Closure.** Sie darf erst nach `8.5d` beginnen, weil sonst `HorizonLevel`, `SpectralDimensionFlow` und `BackreactionFixedPoint` noch zu unbestimmt wären.

8. **Phase 9 bleibt bewusst spät.** Bright-Recovery darf die neue Architektur nicht dominieren. Bright-Recovery darf die neue Architektur nicht dominieren.

Empfohlene Arbeitsbündel

Bündel A — Stamm bis erster echter mathematischer Kern

- 0a → 0c
- 1a → 1b
- 2a → 2c
- 3a → 3c

Bündel B0 — Derived-only-Signaturbereinigung

- 3.5a → 3.5c

Bündel B — algebraischer und netztheoretischer Aufbau

- 4a → 4d
- 5a → 5c
- danach Gate
- 5d, 6a, 6b

Bündel C — Dynamik, Geometrie, emergente GR, Closure

- 7a → 7d
- 8a → 8f
- 8.5a → 8.5d

Bündel D — parallele spätere Erweiterungen

- 6c
- 8g
- 8.6a → 8.6d
- 8.75a → 8.75d
- 9a → 9c

Phase 0 — neues Skelett

Ziel: neues Projekt baut mit leerem, aber sauberem Stamm.

Phase 0a — Repo-Grundgerüst [K1]

- Basisstruktur `CNNA/*`
- `lakefile` / Toolchain / Top-Level-Dateien
- Policies / Tools

Phase 0b — Top-Level-Imports und kritischer Leerpfad [K1]

- `BuildAll`
- `All`
- Pillar-Top-Level-Dateien
- leerer kritischer Pfad

Phase 0c — Meta-Minimum [K1]

- `Meta/CriticalPath`
- `Meta/OpenPoints` als Stub
- erste No-Legacy-Grenze

Phase 1 — ToC-Daten portieren

Ziel: ToC als neue Basissprache verfügbar.

Phase 1a — Kernträger [K1]

- `Approximant`
- `RegionCore`
- `RegionNet`
- `InfiniteCarrier`

Phase 1b — ToC-Adress- und Elternpfadkern [K1]

- `Addresses`
- `ParentPath`
- `Cells`

Phase 1c — ToC-Substratanschluss [K2]

- `Substrate`
- erste Portierung der alten ToC-Basisbegriffe
- noch ohne neue Komplementarchitektur

Phase 2 — neue Sektorobjekte

Ziel: Bright/Dark/Interface formalisieren.

Phase 2a — Bright-Patch [K1]

- `BranchPatch`
- `toApproximant`
- erste Wellformedness-Sätze

Phase 2b — dunkler Sektor [K1]

- `ComplementSectorFamily`
- `darkSupport`
- Basissätze über Disjunktheit und Zerlegung

Phase 2c — Mehrsektor-Split [K1]

- `SectorSplit`
- `InterfaceKernel`
- erste Verträglichkeitsaussagen Split ↔ Komplementfamilie

Phase 3 — verallgemeinerte DtN-/Schur-Architektur

Ziel: mathematischer Kopplungskern.

Phase 3a — Signatur und Blockstruktur [K1]

- `GeneralizedDtN`
- Blocksicht des dunklen Sektors
- Signaturstabilisierung gegenüber dem alten binären Fall

Phase 3b — MultiSchur-Kern [K1]

- `MultiSchur`
- globale vs. iterierte Eliminierung

Phase 3c — erste nichttriviale Sätze [K1]

- `generalizedDtN_wellDefined`
- `iteratedSchur_eq_globalSchur`
- `blockDiagonal_dark_implies_additive_interface`

Phase 3.5 — Derived-only-Signaturbereinigung

Ziel: generische Scaffold-Begriffe aus dem aktiven Kernpfad herausdrängen.

Phase 3.5a — Source maps [K1]

- `Meta/SourceMap`
- jeder aktive Kernbegriff erhält explizite Herkunft: ToC / Bright-Dark / DtN / Backreaction

Phase 3.5b — No-free-parameter audit [K1]

- `Meta/NoFreeParametersOnCriticalPath`
- freie Cutoffs, freie Zustände, freie Dictionaries, freie Skalare aus aktiven Signaturen entfernen oder als Seeds markieren

Phase 3.5c — Derived-only audit [K1]

- `Meta/DerivedOnlyAudit`
- aktiver Pfad nur noch mit abgeleiteten Signaturen; generische Stubs bleiben außerhalb des Kernpfads

Phase 4 — AQFT-Basis portieren

Ziel: algebraisches Rückgrat.

Phase 4a — algebraische Grundschicht [K1]

- `StarAlgebra`
- `CStarNorm`
- `CStarState`
- `CStarKMSState`

Phase 4b — Zustands- und Netzgrundschicht [K1]

- `State`
- `StateNet`
- `LocalNet`

Phase 4c — Repräsentationsgrundschicht [K2]

- `GNS`
- `KMS`
- `RelativeEntropy`

Phase 4d — HK-/Quasilokal-Minimum [K2]

- `QuasiLocal/Basic`
- `HaagKastler/Basic`
- nur das Minimum, noch ohne Complement-Spezialisierung

Phase 5 — ComplementNet aufbauen

Ziel: dunkler Sektor als echtes lokales Netz.

Phase 5a — dunkles lokales Netz [K1]

- `ComplementLocalNet`
- Isotonie-/Lokalitäts-Grundsignatur

Phase 5b — Interface-Netz [K1]

- `InterfaceLocalNet`
- Bright/Dark-Übergänge auf Netzebene

Phase 5c — Zustandsanschluss [K1]

- `ComplementStateNet`
- Restriktion / Inklusion / Zustandstransfer-Minimum

Phase 5d — quasilokale Ergänzung [K2]

- `QuasiLocal/ComplementClosure`
- `HaagKastler/Complement`

Phase 6 — GNS/KMS für dunklen und gemischten Sektor

Ziel: Repräsentations- und Thermostruktur.

Phase 6a — ComplementGNS [K2]

- `ComplementGNS`
- Existenz- und Anschlussfragen

Phase 6b — ComplementKMS [K2]

- `ComplementKMS`
- Interface-induzierte thermische Pfade

Phase 6c — HK-Zusatz für dunklen Sektor [P]

- `HaagKastler/Support`
- `HaagKastler/SupportFull`
- `HaagKastler/SplitProperty`

Phase 7 — OQS-/Kanäle neu anschließen

Ziel: Stage-1/2-Dynamik als sektorielle offene Dynamik.

Phase 7a — Kanalgrundschicht [K1]

- `SectorChannels`
- `SectorSysEnv`

Phase 7b — endliche / semigruppenartige Dynamik [K2]

- `Finite`
- `Lindblad`
- `Semigroup`

Phase 7c — Entropie- und Flussschicht [K2]

- `RelativeEntropyFlow`
- erste Monotonie-/Dissipationspfade

Phase 7d — Backreaction-Anschluss [K2]

- `Backreaction`
- Kopplung OQS ↔ Interface ↔ Geometrie

Phase 8 — emergente Kinematik/Geometrie

Ziel: Interface-induzierte Spacetime-Struktur.

Phase 8a — Einflussrelationen [K1]

- `ComplementInfluence`
- Einfluss als Primärstruktur

Phase 8b — diskrete Spacetime-Ableitung [K1]

- `DerivedSpacetime`
- `SpacetimeOfComplementFamily`

Phase 8c — lokale Kovarianzkeime [K2]

- `LocalCovarianceSeed`
- `PillarD/DerivedSpacetime`
- `PillarD/InterfaceCausality`

Phase 8d — Geometrische Zusatzstruktur [K2]

- `PillarD/ComplementGeometry`
- `PillarD/LocalCovariance`

Phase 8e — D-Materiesektor auf emergenter Geometrie [K2]

- `DerivedStressTensorSeed`
- Materie-/Zustandsgröße auf D aus Bright/Dark/Interface ableiten

Phase 8f — Jacobson-Route / Einstein-Limit-Seed [K2]

- `HorizonThermodynamicsSeed`
- `EntanglementEquilibriumSeed`
- `EinsteinLimitSeed`

Phase 8g — Alternative induzierte Gravitation [P]

- `InducedGravitySeed`
- materiesektorinduzierte IR-Gravitation als Alternativroute

Phase 8.5 — Parameter Closure / Backreaction

Ziel: primitive Altparameter in selbstkonsistente Selektoren überführen.

Phase 8.5a — Verzweigungs-Nichttrivialität [K1]

- `BranchingWitness`
- `nontrivialSideBranching_iff_one_lt_b`

Phase 8.5b — Dimensionsfluss [K2]

- `SpectralDimensionFlow`
- `UVSpectralSelector`

Phase 8.5c — effektive kosmologische Größen [K2]

- `EffectiveLambda`
- `HorizonLevel`

Phase 8.5d — Fixpunkt- und Closure-Schicht [K2]

- `BackreactionFixedPoint`
- `RegularizationClosure`
- Regularisierungsherabstufung oder -eliminierung

Phase 8.6 — Asymptotic-Safety-Kompatibilitätsstrang

Ziel: die bereits implizit vorhandenen AS-nahen Strategien explizit machen und ihre Grenzen markieren.

Phase 8.6a — coarse-graining-Saat [K2]

- `BackgroundIndependentCoarseGraining`
- `horizonLevel` als cutoff-/Skalenselektor lesen

Phase 8.6b — spektrale Modenzählung [K2]

- `CutoffModeCounting`
- `SpectralCutoffMap`

Phase 8.6c — laufende Boundary-/Interface-Daten [P]

- `RunningBoundaryData`
- Unterscheidung Bulk vs. Boundary/Interface-Daten

Phase 8.6d — RG-Samen im Sektorbild [P]

- `SectorRGSeed`
- erste formale Grenze zwischen CNNA-Closure und vollwertigem RG-Fluss

Phase 8.75 — Dualitäts- / Glueing-Strang

Ziel: präzise Mehrfachbeschreibungen emergenter Sektoren mathematisch organisieren.

Phase 8.75a — Boundary- und Glueing-Minimum [P]

- `BoundaryStates`
- `FunctorialGlue`

Phase 8.75b — bulk/boundary-artiges Dictionary [P]

- `BulkBoundaryDictionary`
- erste Konsistenzsätze

Phase 8.75c — Defektstrang [P]

- `Defects.Basic`
- `Defects.Fusion`

Phase 8.75d — Dualitätsdictionary [P]

- `DualityDictionary`

- optional später `ConformalWindow`

Phase 9 — Bright-Spezialfall rückgewinnen

Ziel: zeigen, dass der alte Boundary-Matrix-Zweig ein Spezialisierungsfenster ist.

Phase 9a — Bright-only Reduktion [S]

- Bright-only Restriktion der neuen Architektur

Phase 9b — Recovery des alten Randnetzes [S]

- Recovery des alten Randnetzes als Spezialfall

Phase 9c — Alt↔Neu-Kompatibilität [S]

- Kompatibilitätsbeweise Alt ↔ Neu
- Replay ausgewählter alter Beispiele

Empfohlene Build-Grenzen

Nach folgenden Mikrophasen sollte jeweils ein harter Build-Gate stehen:

- 0c
- 1b
- 2c
- 3c
- 3.5c
- 4d
- 5c
- 6b
- 7d
- 8d
- 8.5d
- 8.75b
- 9c

VII. Erste harte Theoremliste

Core

- `branchPatch_bright_finite`
- `complementSectorFamily_disjoint_union`
- `commonTrunk_disjoint_privateIR`
- `sideBranches_pairwise_disjoint`
- `ownUVTails_disjoint_from_bright`
- `nontrivialSideBranching_iff_one_lt_b`
- `branchingWitness_exists_of_one_lt_b`
- `parameterClosure_classifies_ontic_dynamic_numeric`

OQS

- `generalizedDtN_wellDefined`
- `generalizedDtN_reduces_to_old_DtN_on_binary_split`
- `iteratedSchur_eq_globalSchur`
- `dark_blockDiagonal_implies_interface_additivity`
- `interface_backreaction_monotone_under_coarsening`
- `regularization_secondary_to_dirichlet_closure`
- `backreaction_fixedPoint_wellPosed`
- `horizonLevel_consistent_with_backreaction`

AQFT

- `derivedRegion_assignment_used_by_complementNet`
- `derivedState_assignment_used_by_complementStateNet`
- `complementNet_isotone`
- `complementNet_local`
- `interfaceNet_isotone`
- `state_restricts_bright_to_interface`
- `state_restricts_dark_to_interface`
- `complementGNS_exists`
- `complementKMS_from_interface_balance`
- `boundaryStateSpace_composes_under_gluing`
- `defectFusion_wellDefined`
- `dualityDictionary_preserves_selected_observables`

Integration

- `tailElim_channel_factors_through_generalizedDtN`
- `effectiveLambda_not_primitive_scalar`
- `horizonLevel_not_primitive_cutoff`
- `spectralDimensionFlow_from_derived_heat_data`
- `conformalCausalStructure_from_complementInfluence`
- `derivedStressTensorSeed_from_interface_balance`
- `entanglementEquilibriumSeed_defined`
- `einsteinLimitSeed_from_local_equilibrium`
- `inducedGravitySeed_from_matter_sector`
- `relativeEntropy_monotone_on_sector_channels`
- `derivedSpacetime_from_interface_influence`
- `effectiveLambda_from_interface_balance`
- `spectralDimensionFlow_defined`
- `uvSpectralSelector_selects_branching`
- `approximant_realizes_effective_truncation_strategy`
- `horizonLevel_can_be_read_as_background_independent_cutoff_seed`
- `spectralCutoffMap_compatible_with_dimension_flow`
- `functorialGlue_composes_regions`
- `bulkBoundaryDictionary_consistent`
- `bright_boundary_net_recovered_as_special_case`

VIII. Gates

Gate A — Sektorgrundlage

- `gate_branchPatch_wellformed`
- `gate_complementFamily_wellformed`
- `gate_sectorSplit_total`

Gate A1 — Derived-only-Audit

- `gate_every_active_definition_has_source_map`
- `gate_no_free_cutoff_on_critical_path`
- `gate_no_free_state_on_critical_path`
- `gate_no_free_dictionary_on_critical_path`
- `gate_seed_modules_outside_critical_path`
- `gate_no_unsuffixed_generic_module_on_critical_path`

Gate B — DtN-Kern

- `gate_generalizedDtN_exists`
- `gate_multischur_consistent`
- `gate_oldDtN_embeds_into_new`

Gate C — Komplementnetz

- `gate_complement_isotony`
- `gate_complement_locality`
- `gate_interface_state_restriction`

Gate D — Repräsentation/Thermik

- `gate_complement_GNS`
- `gate_complement_KMS`
- `gate_entropy_flow_monotone`

Gate D1 — Dualitäts- / Glueing-Strang

- `gate_boundary_state_space_defined`
- `gate_functorial_glue_defined`
- `gate_defect_fusion_defined`
- `gate_duality_dictionary_preserves_core_data`

Gate D1.5 — Emergent-GR-Strang in D

- `gate_conformal_causal_structure_defined`
- `gate_metric_scale_seed_defined`
- `gate_derived_stress_tensor_seed_defined`
- `gate_entanglement_equilibrium_seed_defined`

- `gate_einstein_limit_seed_defined`
- `gate_induced_gravity_seed_defined`
- `gate_flrw_window_defined`

Gate D2 — Parameter-Closure / Backreaction

- `gate_one_lt_b_from_branching_nontriviality`
- `gate_uv_selector_defined`
- `gate_backreaction_fixedPoint_defined`
- `gate_horizonLevel_replaces_primitive_Lmax`
- `gate_regularization_secondary_or_eliminated`

Gate D3 — Asymptotic-Safety-Kompatibilität

- `gate_effective_truncation_interpretable`
- `gate_backreaction_not_purely_discarded`
- `gate_cutoff_mode_counting_defined`
- `gate_running_boundary_data_defined`
- `gate_self_consistent_geometry_seed_defined`

Gate E — Rückgewinnung des alten Pfads

- `gate_boundary_path_is_bright_special_case`
- `gate_old_examples_replay_under_bright_restriction`

IX. Minimaler Startsatz von Dateien

Wenn der Projektstart maximal fokussiert erfolgen soll, zuerst nur:

- `CNNA/Core/BranchPatch.lean`
- `CNNA/Core/ComplementSectorFamily.lean`
- `CNNA/Core/SectorSplit.lean`
- `CNNA/OQS/GeneralizedDtNScaffold.lean`
- `CNNA/AQFT/ComplementLocalNetScaffold.lean`
- `CNNA/AQFT/ComplementStateNetScaffold.lean`
- `CNNA/Integration/ComplementInfluence.lean`
- `CNNA/Meta/CriticalPath.lean`

und dann Build-Gate.

X. Strategische Zielaussage

CNNA soll zeigen:

1. dass der alte bright/boundary-first AQFT-Pfad nur ein sichtbarer Spezialfall ist,
2. dass der ToC natürlicherweise eine helldunkle Komplementarchitektur trägt,

3. dass OQS, AQFT und emergente Kinematik über den Interface-Sektor zusammenhängen,
4. dass spätere QFT-Gesichter als sektorielle Emergenzregime derselben tieferen Struktur lesbar werden,
5. dass die noch primitiven Strukturgrößen des Altpfads in einen **Parameter-Closure-Pfad** überführt werden können,
6. dass in Pillar D eine **effektive GR-/Einstein-Grenze** aus der ToC/QFT-Stammtheorie emergiert, mit Hauptziel **Jacobson-/Entanglement-Gleichgewicht** und alternativer Route **induzierte Gravitation aus dem Materiesektor**,
7. dass das heutige **Standardmodell-Patchwork** in zerlegbare derived Teilprobleme auseinandergezogen und als kohärente späte IR-Effektive Theorie in die Stammtheorie reintegriert werden kann.

Pillar D: strikter Pfad zur emergenten GR

Wenn Raumzeit in Pillar D verortet wird, genügt dort nicht bloß ein kinematisches Spacetime-Labeling. Pillar D muss vielmehr den **GR-spezifischen Kreis** schließen:

- Kausal-/Konformstruktur,
- Maß-/Skalenfixierung,
- lokaler Materie-/Zustandssektor auf der emergenten Geometrie,
- Backreaction,
- effektive Einstein-/Friedmann-artige IR-Dynamik.

Der derived-only-Pfad in D ist daher zweistufig organisiert:

Primärroute — Jacobson / Entanglement-Equilibrium

Hauptziel ist nicht, die Einstein-Gleichung als Fundament einzusetzen, sondern sie als **lokale Zustandsgleichung** aus Horizon-/Entanglement-/Entropiebalance zu gewinnen.

Strikter Pfad:

1. `ComplementInfluence`, `DerivedSpacetime`, `InterfaceCausality`
2. liefern die emergente Kausal-/Konformstruktur.
3. `HorizonLevel`, `SpectralDimensionFlow`, Volumen-/Skaleninformation
4. fixieren die fehlende Maß-/Skalenseite der Geometrie.
5. `ComplementLocalNet`, `ComplementStateNet`, `ComplementGNS`, `ComplementKMS`
6. liefern den Materie-/Zustandssektor auf der emergenten Geometrie.
7. `RelativeEntropyFlow`, `InterfaceEntropy`, `EffectiveLambda`
8. liefern den thermodynamisch-/entanglementbasierten Input.
9. `EntanglementEquilibriumSeed`, `DerivedStressTensorSeed`

10. formulieren den lokalen Gleichgewichts- und Energiefluss-Seed.

11. `EinsteinLimitSeed`

12. zielt auf die IR-Effektive Gleichung ab, in der eine Einstein-/Jacobson-artige Dynamik als emergente Zustandsgleichung erscheint.

Alternativroute — induzierte Gravitation aus dem Materiesektor

Falls der Jacobson-Pfad nicht den ganzen dynamischen Gehalt liefert, bleibt als klarer Alternativpfad:

1. Materiesektor / Komplementnetz / Zustandssektor auf D präzise aufbauen,
2. dunkle/komplementäre Freiheitsgrade geeignet integrieren / eliminieren,
3. daraus eine effektive gravitative Dynamik auf D gewinnen,
4. `InducedGravitySeed` als Platzhalter für diese Route führen.

Diese Route behandelt Gravitation als **durch den Materiesektor induzierte IR-Dynamik**, nicht als primitiven ToC-Baustein.

Strikte derived-only-Regel für Pillar D

Folgende Begriffe dürfen in D **nicht** frei eingeführt werden:

- freie Lorentzmetrik,
- freier Stress-Energy-Tensor,
- freie Einstein-Gleichung,
- freier bevorzugter Vakuumzustand,
- freie kosmologische Konstante.

Sie müssen aus den bereits vorhandenen CNNA-Strukturen abgeleitet werden:

- Kausalstruktur aus `ComplementInfluence` / `DerivedSpacetime`,
- Maß-/Skalenfixierung aus `HorizonLevel` / spektralen Daten,
- Materiesektor aus `ComplementLocalNet` / `ComplementStateNet` / GNS / KMS,
- Dynamik aus Entropie-/Backreaction-/Eliminationsdaten.

Neue Modulidee für emergente GR in D

Zusätzliche bzw. verstärkte Dateien/Begriffe:

- `CNNA/PillarD/EntanglementEquilibriumSeed.lean`
- `CNNA/PillarD/DerivedStressTensorSeed.lean`
- `CNNA/PillarD/EinsteinLimitSeed.lean`
- `CNNA/PillarD/InducedGravitySeed.lean`
- `CNNA/PillarD/FLRWSectorWindow.lean`
- `CNNA/PillarD/HorizonThermodynamicsSeed.lean`

Neue Theoremziele / Diagnosen aus dem D-GR-Strang

- `conformalCausalStructure_from_complementInfluence`
- `volumeScaleFixesMetricSeed`
- `derivedStressTensorSeed_from_interface_balance`

- `entanglementEquilibriumSeed_defined`
- `einsteinLimitSeed_from_local_equilibrium`
- `inducedGravitySeed_from_matter_sector`
- `effectiveLambda_enters_ir_geometry_equation`
- `flrwSectorWindow_consistent_with_horizonLevel`

Neue Gates aus dem D-GR-Strang

- `gate_conformal_causal_structure_defined`
- `gate_metric_scale_seed_defined`
- `gate_derived_stress_tensor_seed_defined`
- `gate_entanglement_equilibrium_seed_defined`
- `gate_einstein_limit_seed_defined`
- `gate_induced_gravity_seed_defined`
- `gate_flrw_window_defined`

Pillar E (spät): Standardmodell aus der Stammtheorie

Für CNNA darf das Standardmodell **nicht** als fertiger Block angeklebt werden. Der sinnvolle Weg ist, das heutige SM zunächst als **Patchwork aus mehreren logisch verschiedenen Teilproblemen** zu zerlegen und erst danach kohärent wieder in die Stammtheorie einzufügen.

Die relevante Primärliteratur legt genau diese Zerlegung nahe: Das Standardmodell ist eine renormierbare lokale QFT mit Eichgruppe `SU(3)×SU(2)×U(1)`, chiralem Fermionspektrum, Higgs-Sektor, Yukawa-Struktur und nichttrivialen Anomalie-/Ladungsbedingungen. Diese Bausteine sind konsistent miteinander, aber nicht bereits aus einem allgemein akzeptierten tieferen Prinzip vollständig hergeleitet. Ein besonderer Engpass für jede diskrete/lattice-nahe Stammtheorie ist die Realisierung **chiraler Fermionen**; genau hier greift der Nielsen–Ninomiya-Komplex. Gleichzeitig zeigen Spektral-/NCG-Ansätze, dass zumindest ein Teil des SM-Patchworks geometrisch/algebraisch vereinheitlicht organisiert werden kann. (arxiv.org)

Das Standardmodell als zu zerlegendes Patchwork

Für den CNNA-Pfad sollte das SM in mindestens sechs Teilprobleme zerlegt werden:

1. **Eichsektoren / lokale Symmetrie**
2. Woher kommen die effektiven lokalen Eichfreiheitsgrade?
3. Was entspricht in CNNA später `SU(3)`, `SU(2)`, `U(1)` oder deren Vorstufen?
4. **chirales Materiespektrum**
5. linke/rechte Asymmetrie der Fermionen
6. Generationenstruktur
7. entscheidende Konsistenzfrage wegen chiralem Lattice-/diskretem Ursprung
8. **Anomalien / Ladungsquantisierung**
9. Anomaliefreiheit ist keine Nebensache, sondern Konsistenzbedingung

10. Ladungsquantisierung und Hyperladungsstruktur müssen auf emergenter Seite verträglich rekonstruiert werden

11. **Higgs-/Symmetriebrechungssektor**

12. EWSB nicht als freier Zusatz, sondern als emergenter Ordnungss-/Kopplungssektor

13. Masseerzeugung / Goldstone-/Anderson-Higgs-artige Mechanismen

14. **Flavor-/Yukawa-Patchwork**

15. Yukawa-Matrizen, Massenhierarchien, CKM/PMNS, mögliche Familienreplikation

16. dies ist klar von der bloßen Existenz der Eichgruppe zu trennen

17. **QCD-/IR-Hadronfenster**

18. asymptotisch freie UV-Beschreibung vs. konfinierendes IR-Regime
19. Hadronenwelt und chirale EFT als spätes Spezialfenster

Strikte CNNA-Leitfrage für das SM

Nicht:

- „Wie setzen wir das Standardmodell auf den ToC?“

sondern:

- „Welche derived Bright/Dark/Interface-Strukturen liefern erstens emergente Eichsektoren, zweitens chirales/anomaliefreies Materiespektrum, drittens einen Symmetriebrechungs-/Flavor-Sektor und viertens ein IR-Matching auf das Standardmodell oder SMEFT?“

Hauptprobleme und direkte Konsequenzen für CNNA

1. Eichsektoren emergent machen

Die Literatur zu emergenten Eichfeldern zeigt, dass lokale Eichstrukturen als effektive Beschreibungen aus tieferer Quantenstruktur emergieren können. Für CNNA ist das der natürlichste Einstieg: Eichsymmetrien zunächst als **emergente Redundanz / effektive lokale Organisationsform** aus Komplementnetz, Defekten, Glueing und Interface-Struktur lesen – nicht als primitives ToC-Postulat. (arxiv.org)

2. Chiralität ist der eigentliche harte Engpass

Für eine diskrete/lattice-nahe Stammtheorie ist die Realisierung chiraler Fermionen der kritischste Punkt. Genau hier greifen Nielsen–Ninomiya-artige Obstruktionen; moderne Arbeiten betonen weiterhin, dass eine direkte lokale chiral-gauge-theoretische Realisierung auf Gitter-/diskretem Ursprung hochgradig nichttrivial ist und oft über Boundary-/Anomaly-inflow-/höherdimensionale Konstruktionen gedacht wird. Für CNNA bedeutet das: **Ein später Standardmodell-Pfad muss explizit einen Chiralitäts-/Anomaly-Inflow-Strang tragen.** (arxiv.org)

3. Anomalien und Ladungsstruktur müssen als Gates kommen

Anomaliefreiheit und Ladungsquantisierung sind integrale Konsistenzbedingungen des SM und kein Feintuning am Ende. Für CNNA heißt das: Ein emergenter Eich-/Materiesektor darf erst dann als SM-kompatibel gelten, wenn die entsprechenden Anomalie- und Ladungsgates bestanden sind. (arxiv.org)

4. Higgs/Yukawa sind ein eigener Sektor, nicht automatisch mit der Eichgruppe gegeben

Die Existenz einer Eichgruppe allein liefert noch weder Massen noch Flavor. Der Higgs-/Yukawa-Sektor ist ein eigener organisationsbedürftiger Teil des Patchworks. NCG-/Spektral-Ansätze sind hier lehrreich, weil sie zeigen, dass Gauge- und Higgs-Strukturen gemeinsam aus einer reicheren algebraisch-geometrischen Datenstruktur organisiert werden können – ohne dass das schon die CNNA-Lösung wäre. (arxiv.org)

5. „Das Standardmodell" ist nur die erste IR-Stufe, nicht das Ende

Selbst bei Erfolg sollte CNNA nicht zuerst auf die volle beobachtete Teilchenphänomenologie zielen, sondern auf ein sauberes **IR-Matching** zu

- Standardmodell-Gauge-/Matter-Sektor,
- danach gebrochenem elektroschwachem Fenster,
- und schließlich, noch später, SMEFT-/Flavor-/Hadronfenstern.

Strikter derived-only-Pfad für Pillar E

E0 — Voraussetzung

Pillar E darf erst beginnen, wenn A–D bereits stehen:

- lokales Komplementnetz,
- Zustände/GNS/KMS,
- emergente Geometrie / D,
- Parameter-Closure,
- abgeleitete spektrale/entropische Größen.

E1 — emergente Eichsektoren [K2/spät]

Ziel: - lokale effektive Eichfreiheitsgrade aus Bright/Dark/Interface ableiten, - nichtfreie `GaugeSectorSeed`-Begriffe einführen, - zunächst ohne Anspruch auf genau `SU(3)×SU(2)×U(1)`.

E2 — chirales Materiespektrum [K2/spät]

Ziel: - einen derived Chiralitätsmechanismus formulieren, - Boundary-/Interface-/Inflow-Routen prüfen, - `ChiralMatterSeed` nur dann de-seeden, wenn Doubling-/Mirror-Probleme sauber adressiert sind.

E3 — Anomalie- und Ladungsgates [K1/spät]

Ziel: - emergente Ladungszuweisung, - Anomaliefreiheit, - Ladungsquantisierung, - erst danach SM-kompatible Redeweise.

E4 — Higgs-/Symmetriebrechungssektor [K2/spät]

Ziel: - emergente Ordnung/Instabilität/Interface-Kopplung, die Higgs-/Anderson-Higgs-artige Rolle spielen kann, - `HiggsSectorSeed`, `EWSBSeed`.

E5 — Flavor-/Yukawa-Strang [P/spät]

Ziel: - Familienreplikation, Mischungsmatrizen, Hierarchien als eigener Strang, - **nicht** auf den Kernpfad vorziehen.

E6 — QCD-/IR-Fenster [P/spät]

Ziel: - asymptotisch freie UV-Beschreibung vs. konfinierender IR-Sektor, - spätere `HadronWindow`, `ChiralEFTWindow`.

E7 — IR-Matching auf SM / SMEFT [S]

Ziel: - prüfen, ob die emergente IR-Theorie tatsächlich auf das bekannte SM oder eine SM-EFT-artige Beschreibung matched.

Neue Modulidee für den Standardmodell-Strang

Zusätzliche bzw. verstärkte Dateien/Begriffe:

- `CNNA/PillarE/GaugeSectorSeed.lean`
- `CNNA/PillarE/ChiralMatterSeed.lean`
- `CNNA/PillarE/AnomalyCancellationGate.lean`
- `CNNA/PillarE/ChargeQuantizationGate.lean`
- `CNNA/PillarE/HiggsSectorSeed.lean`
- `CNNA/PillarE/EWSBSeed.lean`
- `CNNA/PillarE/YukawaFlavorSeed.lean`
- `CNNA/PillarE/GenerationSeed.lean`
- `CNNA/PillarE/SMMatchingWindow.lean`
- `CNNA/PillarE/SMEFTWindow.lean`
- `CNNA/PillarE/HadronWindow.lean`
- `CNNA/PillarE/ChiralEFTWindow.lean`
- optional später `CNNA/PillarE/AnomalyInflowSeed.lean`

Neue Theoremziele / Diagnosen aus dem SM-Strang

- `gaugeSectorSeed_from_interface_defects`
- `chiralMatterSeed_not_vectorlike`
- `anomalyCancellationGate_passed`
- `chargeQuantizationGate_passed`
- `higgsSectorSeed_from_interface_instability`
- `ewsbSeed_generates_mass_splitting`
- `generationSeed_from_branching_or_internal_sector`
- `yukawaFlavorSeed_not_primitive`
- `smMatchingWindow_consistent`
- `smeftWindow_defined`

Neue Gates aus dem SM-Strang

- `gate_gauge_sector_seed_defined`
- `gate_chiral_matter_seed_defined`
- `gate_anomaly_cancellation_passed`
- `gate_charge_quantization_passed`
- `gate_higgs_sector_seed_defined`
- `gate_ewsb_seed_defined`
- `gate_sm_matching_window_defined`

Derived-only-Regel für Pillar E

Folgende Dinge dürfen im SM-Strang nicht frei gesetzt werden:

- freie Wahl der SM-Eichgruppe als Postulat,
- freies chirales Fermionspektrum,
- freie Hyperladungen,
- freier Higgs-Dublettsektor,
- freie Yukawa-Matrizen,
- freie Familienzahl.

Falls solche Daten im frühen E-Strang nur als organisatorische Platzhalter auftauchen, müssen sie als `Seed` / `Window` markiert bleiben.

Konsequenz für die Stammtheorie

Das heutige Standardmodell wird in CNNA **nicht** als Block importiert. Es wird in logisch getrennte derived Teilprobleme zerlegt, die nacheinander wieder zusammengesetzt werden:

- emergente Eichsektoren,
- chirale/anomaliefreie Materie,
- Higgs-/Symmetriebrechung,
- Flavor/Familien,
- QCD-/Hadronfenster,
- IR-Matching.

Erst wenn diese Gates getragen werden, darf man von einem kohärent in die Stammtheorie eingefügten Standardmodell-Sektor sprechen.

Asymptotic-Safety-nahe Strategien: Einordnung für CNNA

CNNA ist **nicht** asymptotic safety. Dennoch tragen der Bright-Approximant und der bisherige REAL-OQS/CNNA-Pfad bereits mehrere Strategien in sich, die stark an den methodischen Kern asymptotic-safety-naher Forschung erinnern.

Was die Primärliteratur in AS wirklich als Kern hervorhebt

Aus den zentralen AS-Arbeiten ergibt sich ein relativ klarer Methodenkanon:

1. **effektive Average Action / FRGE statt nackter Bare-Ansatz**
2. die nichtperturbative Dynamik wird auf einer skalenabhängigen effektiven Aktion formuliert
3. der RG-Fluss lebt auf einem Theorieraum von Funktionalen und wird durch eine Wilson-/FRG-artige Gleichung gesteuert
4. **background-independent coarse graining**
5. das coarse graining darf in der Gravitation nicht auf einem starren, extern fixierten Hintergrund beruhen
6. die Skalenwahl muss vielmehr relativ zu einer dynamischen bzw. variablen Geometrie formuliert werden
7. **bimetrische / split-symmetry-sensitive Sicht**
8. der Hintergrund ist nicht bloß Hilfsgröße, sondern erscheint in der EAA als eigener struktureller Träger
9. daraus folgt die Relevanz von Split-Symmetrie, Hintergrundabhängigkeit und bimetric-truncation-artigen Fragen
10. **self-consistent backgrounds / tadpole equation**
11. selbstkonsistente Hintergründe sind nicht einfach eingesetzt, sondern erfüllen eine running tadpole equation
12. dies liefert ein k-abhängiges Gegenstück zur klassischen Feldgleichung
13. **laufende Boundary-Daten**
14. AS-nahe Analysen zeigen explizit, dass Rand-/Boundary-Aktionen eigenständig laufen können und nicht nur Bulk-Daten wiederholen
15. **Spektraldimension als Diagnosegröße**
16. der Lauf der Spektraldimension ist in AS keine Dekoration, sondern ein wichtiger Hinweis auf skalenabhängige Quantengeometrie
17. typisch ist ein UV-Wert von 2 in vierdimensionalen QEG-Regimen, bei zugleich möglichen Zwischenplateaus und truncation-sensitiven Übergangsregimen

Was im Bright-Approximanten bereits AS-nah angelegt ist

1. **endliche effektive Beschreibung bei expliziter Skale**
2. der Bright-Approximant ist eine endliche, kontrollierte Beschreibung aktiver Freiheitsgrade

3. das entspricht strukturell einer truncation / effektiven Beschreibung auf einer skalenmarkierten Stufe

4. **coarse graining durch aktiven vs. eliminierten Sektor**

5. Bright trägt die explizit behaltenen Freiheitsgrade
6. Dark/Komplement wird nicht einfach vergessen, sondern beeinflusst Bright über DtN / Schur / Tail-Elimination

7. dies ist deutlich näher an einer Wilson-/FRG-Logik als ein bloß abgeschnittener harter UV-Cut ohne Rückwirkung

8. **nur in Summe unendliche Komplementfamilien als RG-ähnlicher Turm**

9. jede einzelne Komplementfamilie kann endlich kontrollierbar gewählt sein, während die Gesamtheit der möglichen Komplementfamilien eine offene/unendliche Struktur repräsentiert
10. das ähnelt eher einem **Turm effektiver Beschreibungen** als einem einzigen absoluten Komplement

11. genau hierin liegt die stärkste AS-Analogie: nicht ein einzelner Cutoff, sondern eine Familie skalen- und sichtspezifischer effektiver Beschreibungen

12. **skalenabhängige effektive Geometrie statt fixer Hintergrund**

13. in CNNA soll `L_max` gerade nicht fundamental bleiben, sondern als `HorizonLevel` / effektive IR-Größe rückgekoppelt werden

14. damit liegt ein Keim für eine selbstkonsistente skalenabhängige Geometrie vor

15. **Spektraldimension als Probe statt nur als Dekoration**

16. der geplante `SpectralDimensionFlow` ist direkt kompatibel mit einer der wichtigsten Diagnosegrößen asymptotic-safety-naher Quantengeometrie

17. **effektive kosmologische Größe aus Rückwirkung**

18. `EffectiveLambda` als aus Interface-Balance entstehende Größe passt strukturell zu der Idee, dass makroskopische Geometrie aus einer skalenabhängigen effektiven Dynamik mitbestimmt wird

Was im Komplement / Dark-Sektor derzeit gerade nicht AS-artig ausgebaut ist

1. **kein expliziter RG-Fluss auf dem Komplement**
2. der Dark-Sektor ist bislang primär organisierte Komplementfamilie und Eliminationspartner

3. er ist noch keine explizite skalenlaufende effektive Aktion / kein Theory-Space-Punkt mit Beta-Funktionen

4. **kein expliziter FRGE-/EAA-Strang**

5. CNNA besitzt noch keine genaue Entsprechung einer effektiven Average Action auf einem Raum sektorabhängiger Funktionale
6. solange dies fehlt, bleibt die AS-Nähe methodisch, nicht identitär
7. **keine explizite split-symmetry-/bimetric-Kontrolle**
8. Bright/Dark/Interface ist zwar strukturell zweifeld- bzw. mehrfeldartig reich genug, aber noch nicht als genauer Hintergrund-/Fluktuations-Split mit eigener Ward-Identität formuliert
9. **keine selbstkonsistente Hintergrundgleichung für den Komplementsektor**
10. es gibt noch keinen expliziten Tadpole-/Self-consistent-background-Schritt für Bright/Dark/Interface zusammen
11. `BackreactionFixedPoint` ist geplant, aber noch nicht der volle EAA-artige Selbstkonsistenzmechanismus
12. **kein expliziter UV-critical-surface-Begriff**
13. CNNA besitzt bislang Selektoren und Gates, aber noch keinen begrifflich sauberen Raum relevanter/irrelevanter Richtungen analog zum UV-kritischen Mannigfaltigkeitsbild
14. **laufende Rand-/Interface-Kopplungen fehlen noch als eigener Strang**
15. da Bright/Dark/Interface eine rand- und komplementgetriebene Architektur bilden, liegt die AS-nahe Lehre nahe, dass Bulk- und Boundary-/Interface-Daten unterschiedlich laufen können
16. genau dies ist im bisherigen CNNA-Baum noch nicht explizit ausgebaut
17. **Modenzählung im skalenabhängigen Hintergrund fehlt noch**
18. asymptotic-safety-nahe coarse graining arbeitet mit spektraler Modenselektion relativ zu einer skalenabhängigen Geometrie
19. CNNA hat zwar `SpectralDimensionFlow` und `HorizonLevel`, aber noch keinen expliziten `ModeCounting`-/`CutoffMode`-Strang

Präzise Diagnose

Daher ist der passende Satz:

- Der **Bright-Approximant** enthält bereits mehrere **AS-nahe Strategien** implizit: skalenmarkierte effektive Beschreibung, Rückwirkung eliminierter Freiheitsgrade, Spektraldiagnostik, emergente Geometrieskala.
- Die **Komplementfamilien in ihrer Gesamtheit** tragen bereits die Idee eines **familienweisen coarse grainings** und sprechen gegen die Lesart eines einzigen absoluten UV-Rests.
- Das **Komplement** ist aber derzeit noch **nicht** als vollwertiger background-independent RG-Sektor ausgebaut, sondern primär als strukturierter Eliminations- und Rückwirkungssektor.

CNNA steht also näher an einer **AS-kompatiblen, aber noch nicht RG-vollständig formulierten** Architektur als an einem völlig fremden Paradigma.

Konsequenzen für die CNNA-Architektur

Wenn die AS-Nähe systematisch nutzbar gemacht werden soll, sind sechs zusätzliche Arbeitsstränge sinnvoll:

1. **Background-Independent Coarse Graining explizit machen**
2. `HorizonLevel` nicht nur als Resultat, sondern als skalenabhängigen cut-off-bezogenen geometrischen Selektor lesen
3. **selbstkonsistente Hintergrund-/Geometriegleichung**
4. `BackreactionFixedPoint` zu einem echten Bright/Dark/Interface-Tadpole-Prinzip ausbauen
5. **Running Boundary / Interface Data**
6. eigene scale dependent Rand-/Interface-Kopplungen für die Bright/Dark-Schnittfläche einführen
7. **Spektrale Modenzählung und UV/IR-Selektion**
8. explizit formal machen, welche Freiheitsgrade bei welcher effektiven Skale „aktiv“ sind und wie sich dies mit `SpectralDimensionFlow` und `EffectiveLambda` verträgt
9. **Split-Symmetrie-/Bimetric-Samen**
10. Bright/Dark/Interface nicht nur als Sektorzerlegung, sondern auch als Kandidat für eine präzisere Hintergrund-/Fluktuations-Trennung lesen
11. **Theory-Space-/Relevanz-Samen**
12. ein erster Begriff von laufenden effektiven Daten, Theorieparametern und möglichen relevanten/irrelevanten Richtungen muss explizit gemacht werden, ohne vorschnell eine volle FRGE zu behaupten

Neue Modulidee aus dem AS-Strang

Zusätzliche bzw. verstärkte Dateien/Begriffe:

- `CNNA/Integration/BackgroundIndependentCoarseGrainingSeed.lean`
- `CNNA/Integration/RunningBoundaryDataSeed.lean`
- `CNNA/Integration/TadpoleSeed.lean`
- `CNNA/Integration/SplitSymmetrySeed.lean`
- `CNNA/PillarD/CutoffModeCountingSeed.lean`
- `CNNA/PillarD/SpectralCutoffMapSeed.lean`
- `CNNA/OQS/SectorRGSeed.lean`
- `CNNA/OQS/TheorySpaceSeed.lean`

Neue Theoremziele / Diagnosen aus dem AS-Strang

- `approximant_realizes_effective_truncation_strategy`
- `generalizedDtN_retains_backreaction_of_eliminated_sector`
- `complementFamily_realizes_scale_ladder_seed`
- `horizonLevel_can_be_read_as_background_independent_cutoff_seed`
- `runningBoundaryData_distinct_from_bulk_data`
- `spectralCutoffMap_compatible_with_dimension_flow`
- `tadpoleSeed_extends_backreactionFixedPoint`
- `splitSymmetrySeed_distinguishes_background_and_interface_roles`
- `sectorRGSeed_identifies_running_effective_data`
- `backreactionFixedPoint_extends_to_selfConsistent_geometry_seed`

Neue Gates aus dem AS-Strang

- `gate_effective_truncation_interpretable`
- `gate_backreaction_not_purely_discarded`
- `gate_scale_ladder_seed_defined`
- `gate_cutoff_mode_counting_defined`
- `gate_running_boundary_data_defined`
- `gate_tadpole_seed_defined`
- `gate_split_symmetry_seed_defined`
- `gate_self_consistent_geometry_seed_defined`

Neue Phase im kritischen Pfad

Zwischen Phase 8.5 und dem Dualitäts-/Glueing-Strang ist ein zusätzlicher optional-früher Strang sinnvoll:

Phase 8.6 — Asymptotic-Safety-Kompatibilitätsstrang

Ziel: die bereits implizit vorhandenen AS-nahen Strategien explizit machen und ihre Grenzen markieren.

- `BackgroundIndependentCoarseGraining`
- `CutoffModeCounting`
- `SpectralCutoffMap`
- `RunningBoundaryData`
- `TadpoleSeed`
- `SplitSymmetrySeed`
- `SectorRGSeed`
- `TheorySpaceSeed`
- Erweiterung von `BackreactionFixedPoint` in Richtung self-consistent geometry seed

Diese Phase dient **nicht** dazu, CNNA in asymptotic safety aufzulösen. Sie dient dazu, die bereits vorhandene strukturelle Nähe präzise zu nutzen, ohne mehr zu behaupten als tatsächlich formalisiert ist.

Stringtheorie-nahe Mathematik: Relevanz für CNNA

CNNA übernimmt **nicht** die Stringtheorie als Fundament. Der ToC ist kein „Netzwerk aus Strings“ im üblichen Sinn, und die stringtypischen Grundannahmen (Weltblatt-Sigma-Modelle, kritische Dimensionen, Supersymmetriezwang, Calabi-Yau-Kompaktifizierung) gehören **nicht** in die frühe Kernarchitektur.

Der relevante Gewinn aus dem stringtheorie-nahen Umfeld liegt vielmehr in der Mathematik von:

1. **Glueing / Funktorialität**
2. Zustandsräume an Rändern
3. Komposition von Regionen
4. Verträglichkeit von Bright/Dark/Interface-Zerlegungen mit Kompositionen
5. **Defekten / Interfaces / Boundary States**
6. Übergänge zwischen verschiedenen emergenten Sektoren
7. strukturierte Kopplungen statt bloßer ad-hoc-Übergänge
8. mögliche spätere defect-fusion-artige Strukturen
9. **Dualitäts-Dictionaries**
10. wenn mehrere emergente QFT-Sektoren dieselbe tiefere CNNA-Struktur unterschiedlich beschreiben, müssen präzise Übersetzungsregeln formuliert werden
11. Ziel ist nicht bloße Analogie, sondern Erhalt von Observablen, Zuständen, Entropiedaten und effektiver Dynamik
12. **späte konforme Spezialfenster**
13. falls ein effektiver 2D-, randdominierter oder konformer Sektor emergiert, können später CFT-/VOA-/modulare Werkzeuge relevant werden
14. dies ist jedoch **kein Fundament**, sondern ein spätes Spezialregime

Methodische Abgrenzung

Nicht früh zu übernehmen sind:

- Weltblatt-Formalismus als Fundament
- kritische 10/11 Dimensionen als Grundannahme
- stringtypische Spektren als Startpunkt
- supersymmetrische oder Calabi-Yau-basierte Grundarchitektur

Früh und direkt relevant sind dagegen:

- funktorielle Kompositionen
- bulk/boundary-artige Dictionaries
- kategoriale oder semikategoriale Äquivalenzen zwischen emergenten Sektoren

- Defekt- und Interface-Strukturen als Träger von Übersetzungen zwischen Oberflächenbeschreibungen

Neue Leitfrage aus diesem Strang

Nicht:

- „Ist CNNA eigentlich eine Stringtheorie?“

sondern:

- „Welche emergenten Sektoren von CNNA beschreiben dieselbe Physik, und welche funktoriellen, kategorialen oder defect-vermittelten Übersetzungen realisieren diese Mehrfachbeschreibung?“

Neue Modulidee für den Dualitäts-/Glueing-Strang

Zusätzliche bzw. verstärkte Dateien/Begriffe:

- `CNNA/AQFT/DualityDictionary.lean`
- `CNNA/AQFT/BoundaryStates.lean`
- `CNNA/AQFT/Defects/Basic.lean`
- `CNNA/AQFT/Defects/Fusion.lean`
- `CNNA/Integration/FunctorialGlue.lean`
- `CNNA/Integration/BulkBoundaryDictionary.lean`
- `CNNA/PillarD/ConformalWindow.lean`

Neue Theoremziele aus diesem Strang

- `boundaryStateSpace_composes_under_gluing`
- `functorialGlue_composes_regions`
- `defectFusion_wellDefined`
- `bulkBoundaryDictionary_consistent`
- `dualityDictionary_preserves_selected_observables`

Konsequenz für die Gesamtarchitektur

Der stringtheorie-nahe Mehrwert für CNNA liegt nicht in stringtypischen Mikroobjekten, sondern in der **Mathematik präziser Mehrfachbeschreibungen**. Genau diese Mathematik passt strukturell zu Bright/Dark/Interface, Komplementfamilien und späteren emergenten QFT-Fenstern.

Parameter-Closure: neue Leitlinie

Die Parameterfrage wird im neuen Projekt **dreischichtig** gelesen:

1. **ontische Mikrostrukturparameter**
2. Kandidat: `b`
3. `b` ist kein bloßer Komfortparameter, sondern steuert echte Verzweigung, Sektornichttrivialität und dimensionsrelevante Mikrostruktur.

4. **dynamische / zustandsabhängige Emergenzparameter**

5. Kandidat: `L_max`

6. im neuen Stamm ist `L_max` nicht als ontische Eingabe zu behandeln, sondern als **effektive Horizont-/IR-Größe** `HorizonLevel`, die aus Backreaction, Zustand und emergenter Geometrie selbstkonsistent bestimmt wird.

7. **sekundäre numerische Regularisierer**

8. Dirichlet-/DtN-Stabilisierungsgrößen dürfen nicht vorschnell zur Physik erhoben werden
9. Ziel ist entweder ihre Eliminierung durch stärkere strukturelle Sätze oder ihre Herabstufung zu abgeleiteten Restgrößen.

Konsequenz für den Altpfad

Der bestehende REAL-OQS-Pfad hat bereits mehrere abgeleitete Größen (`β`, dimensions-/break-bezogene Exportgrößen, `φ`-artige Profile). Der neue CNNA-Stamm soll diese Richtung radikalisieren:

- **nicht** mehr nur thermische / modulare Größen ableiten,
- sondern auch die verbleibenden Strukturgrößen in einen geschlossenen Selektionspfad überführen.

Primärer Zielpfad für `1 < b`

Die Bedingung `1 < b` wird nicht als bloße Nichtleereingabe verstanden, sondern als **erste echte Nichttrivialitätsbedingung** der Komplement-Netzarchitektur.

Interpretation:

- `0 < b` sichert nur Nichtleerheit,
- `1 < b` sichert die Existenz echter Verzweigung,
- damit die Existenz exklusiver Seitenäste,
- damit die Existenz eines nichttrivialen dunklen Sektors,
- damit die Existenz eines echten Interface-Sektors.

Der neue Zielbegriff lautet daher:

```
structure BranchingWitness (b : Nat) where
  left  : Fin b
  right : Fin b
  neq   : left ≠ right
```

und der Kernsatz:

- `nontrivialSideBranching_iff_one_lt_b`

soll die strukturelle Äquivalenz von Verzweigungszeugnis und `1 < b` liefern.

Primärer Zielpfad für `b`

`b` soll nicht willkürlich fixiert, sondern über einen **Selektor** aus mehreren Schichten bestimmt werden:

1. **Strukturselektor**
2. nichttrivialer heller/dunkler/interface-Sektor,
3. nichttriviale Komplementfamilie,
4. nichtdegenerierte Mehrsektor-DtN-Rückwirkung.
5. **UV-Selektor**
6. aus einem Spektraldimensionsfluss bzw. Heat-Kernel-artigen Größen soll ein mikroskopisches Regime selektiert werden, das mit der tieferen diskreten Struktur kompatibel ist.
7. **IR-/Backreaction-Selektor**
8. derselbe `b` muss mit einer selbstkonsistenten emergenten Geometrie / Horizontgröße vereinbar sein.

Arbeitsidee:

- `UVSpectralSelector`
- `BranchingSelector`
- `BackreactionSelector`

und daraus ein kombinierter Kandidat:

- `selectedBranching : Nat`

Primärer Zielpfad für `L_max`

Im alten Stamm ist `L_max` ein truncation regulator. Im neuen Stamm soll daraus werden:

- `HorizonLevel`
- `EffectiveIRScale`
- oder ein äquivalenter dynamischer Zustandsschnitt.

Die Leitidee ist:

- ToC + `b` + Komplementnetz → effektive L-/DtN-Daten
- daraus Zustand / Entropie / Einfluss / emergente Geometrie
- daraus effektive makroskopische Größe (`EffectiveLambda` , `HorizonLevel` , `EffectiveExpansion`)
- daraus Rückkopplung auf die zulässige/selbstkonsistente IR-Tiefe.

Das Ziel ist also **nicht** eine nackte Zahl `L_max` , sondern ein Fixpunktproblem:

- `BackreactionFixedPoint`
- `horizonLevel_consistent_with_backreaction`

Sekundärer Zielpfad für Dirichlet-/DtN-Regularisierung

Die Stabilisierung der Dirichlet-/DtN-Schicht ist im neuen Projekt methodisch **sekundär**.

Zielhierarchie:

1. **starker Zielpfad:**
2. Regularisierer verschwinden aus der Physik,
3. weil die relevante Positivität / Koerzivität / Quotientenstruktur intrinsisch bewiesen wird.
4. **schwächerer Zielpfad:**
5. wenn ein Restterm unvermeidbar ist, wird er als abgeleitete Größe aus Spektralgap, Nullmodenstruktur, Offset oder Interface-Balance interpretiert.

Er darf aber nicht als primärer freier Modellparameter übrig bleiben.

Neue Modulidee für Parameter-Closure

Zusätzliche Dateien/Begriffe:

- `CNNA/Core/BranchingWitness.lean`
- `CNNA/Core/ParameterClosure.lean`
- `CNNA/OQS/BackreactionFixedPoint.lean`
- `CNNA/OQS/RegularizationClosure.lean`
- `CNNA/Integration/EffectiveLambdaSeed.lean`
- `CNNA/Integration/HorizonLevelSeed.lean`
- `CNNA/PillarD/SpectralDimensionFlowSeed.lean`

Neue Theoremziele für Parameter-Closure

- `nontrivialSideBranching_iff_one_lt_b`
- `branchingWitness_exists_of_one_lt_b`
- `uvSpectralSelector_selects_branching`
- `effectiveLambda_from_interface_balance`
- `backreaction_fixedPoint_wellPosed`
- `horizonLevel_consistent_with_backreaction`
- `regularization_secondary_to_dirichlet_closure`

Neue Phase im kritischen Pfad

Zwischen bisheriger emergenter Geometrie und Bright-Recovery wird ein zusätzlicher Schritt eingefügt:

Phase 8.5 — Parameter Closure / Backreaction

Ziel: primitive Altparameter in selbstkonsistente Selektoren überführen.

- `BranchingWitness`
- `nontrivialSideBranching_iff_one_lt_b`

- `SpectralDimensionFlow`
- `UVSpectralSelector`
- `EffectiveLambda`
- `HorizonLevel`
- `BackreactionFixedPoint`
- Regularisierungsherabstufung oder -eliminierung

Erst danach ist die Theorie im strengen Sinn **selbstschließend**.

XI. Sofort umsetzbarer Arbeitsauftrag

Als erster echter Implementationsblock:

1. neues Projekt `CNNA` anlegen
2. `Approximant`, `RegionCore`, `InfiniteCarrier`, ToC-Adressierung portieren
3. `BranchPatch`, `ComplementSectorFamily`, `SectorSplit` definieren
4. `GeneralizedDtN` als zentrale neue Datei anlegen
5. beweisen, dass der alte binäre DtN-Fall Spezialfall des neuen mehrsektorigen Falls ist
6. `ComplementLocalNet` als erste echte neue AQFT-Datei bauen
7. Bright-only Recovery als Gate formulieren

Das ist die kleinste saubere Route, mit der das neue Projekt sofort einen echten inhaltlichen Kern besitzt.

XII. Pflicht vs. stark empfohlen vs. optional/ später

Die folgende Tabelle ist an der **tatsächlichen Struktur von** `REALOQS_v0.0604` gegengeprüft. Sie beantwortet nicht die Frage, was philosophisch wünschenswert wäre, sondern was für einen sauberen neuen Stamm **wirklich explizit geplant** sein sollte.

A. Pflicht (ohne diese Dateien/Schichten ist CNNA strukturell unvollständig)

Bereich	Pflichtbestand
Top-Level	`CNNA/Basic.lean`, `CNNA/BuildAll.lean`, `CNNA/All.lean`, `CNNA/PillarA.lean`, `CNNA/PillarB.lean`, `CNNA/PillarC.lean`, `CNNA/PillarD.lean`
Top-Level-Interfaces	`CNNA/Interface.lean`, `CNNA/PillarA/Interface.lean`, `CNNA/PillarB/Interface.lean`, `CNNA/PillarC/Interface.lean`, `CNNA/PillarD/Interface.lean`
Pillar-Aggregatoren	`CNNA/PillarA/All.lean`, `CNNA/PillarA/BuildAll.lean`, `CNNA/PillarB/BuildAll.lean`, `CNNA/PillarC/BuildAll.lean`, `CNNA/PillarD/BuildAll.lean`

Bereich	Pflichtbestand
Policy/Tooling	`CNNA/ImportPolicy.md`, `CNNA/CStarFoundationPolicy.md`, `CNNA/tools/check_import_policy.py`, `CNNA/tools/check_cstar_foundation.py`
Examples-Aggregation	`CNNA/Examples/BuildAll.lean`
Core-Basis A	`Approximant`, `RegionCore`, `RegionNet`, `InfiniteCarrier`, `Selection`, `Interfaces`, `AQFTHandoff`, `Determinism`, `TailInterface`, `BoundaryPorts`, `DirichletLaplacian`, `DirichletLaplacianBridge`, `Exports`
ToC-Basis	`Addresses`, `ParentPath`, `Cells`, `Substrate`, `BranchPatchInstances`, `ComplementSectorInstances`
Neue Kernobjekte	`BranchPatch`, `ComplementSectorFamily`, `SectorSplit`, `InterfaceKernel`, `BranchingWitness`, `ParameterClosure`
OQS-Kern	`DtN`, `DtNStabilized`, `GeneralizedDtN`, `MultiSchur`, `SectorSysEnv`, `SectorChannels`, `Backreaction`, `BackreactionFixedPoint`, `RegularizationClosure`
AQFT-Kern	`StarAlgebra`, `CStarNorm`, `CStarState`, `CStarKMSState`, `State`, `StateNet`, `LocalNet`, `ComplementLocalNet`, `ComplementStateNet`, `InterfaceLocalNet`, `GNS`, `ComplementGNS`, `KMS`, `ComplementKMS`, `RelativeEntropy`
HK-/QuasiLocal-Kern	`QuasiLocal/Basic`, `QuasiLocal/ComplementClosure`, `HaagKastler/Basic`, `HaagKastler/Complement`, `HaagKastler/Support`, `HaagKastler/SupportFull`, `HaagKastler/SplitProperty`
Pillar C Integration	`TailElimAsChannel`, `DerivedSpacetime`, `ComplementInfluence`, `SpacetimeOfComplementFamily`, `LocalCovarianceSeed`, `EffectiveLambda`, `HorizonLevel`
Pillar D	`DerivedSpacetime`, `ComplementGeometry`, `InterfaceCausality`, `LocalCovariance`, `SpectralDimensionFlow`, `DerivedStressTensor`, `EntanglementEquilibrium`, `EinsteinLimit`, `InducedGravity`, `FLRWSectorWindow`, `MinkowskiWindow`
Meta/Gates	`Meta/OpenPoints`, `Meta/VacuityAudit`, `Meta/CriticalPath`, `Meta/NoLegacyInCriticalPath`, `Meta/BrightDarkInterfaceSurface`, sowie Gates für BranchPatch, ComplementSectorFamily, GeneralizedDtN, ComplementNet, ComplementState, ComplementKMS, InterfaceEntropy, DerivedSpacetime

B. Stark empfohlen (sollte sehr früh eingeplant werden, sonst verliert CNNA wichtige Altstärke)

Bereich	Stark empfohlen
Pillar A zusätzliche Basis	`Core/BInterfaces/CovariantNets`, `Core/BInterfaces/GlobalCone`, `Core/BInterfaces/RelativeEntropyHook`, `Core/CylinderSets`, `Core/Derived/BridgeToSubstrate`, `Core/Derived/GNS`, `Core/Derived/GNS_CStar`, `Core/Derived/ModularKMS`
Symmetrie/ Kinematik in A	`Kinematics/Causality`, `Kinematics/Worldline`, `Kinematics/ProperTime`, `Kinematics/DiscreteSpacetime`, `Kinematics/FrameChange`, `Kinematics/SpacetimeRegions`, `Symmetry/ApproxPoincare`, `Symmetry/PoincareLorentzSplit`
Update-/ Dynamikpfad A	`Update/ApproximationPipeline`, `Update/DeterministicExport`, `Update/MismatchDriver`, `Update/MismatchFunctional`, `Update/Step`, `Update/Seed`, `Update/TwoStageApprox`, `Update/TailEliminationDtN`, `Update/TailEliminationDtNStabilized`
Physik-/RG-Seeds A	`Physics/ScaleBreaking*`, `RG/ScaleWindow`, `OQS/LiebRobinsonHook`, `OQS/DtN_TailHook`, `OQS/SplitInvariants`
AQFT-Zusatz B	`GlobalCone`, `TimeOrientation`, `TimeReversal`, `Exports`, `DualityDictionary`, `BoundaryStates`, `Defects/*`
Boundary-Recovery-Zweig	gezielte Neuinterpretation der `BoundaryMatrix*`-Derived-Dateien als Bright-Spezialfall mit formalen Recovery-Sätzen
Pillar C Zusatzbasis	`OQS/Finite`, `OQS/Finite/Kraus`, `OQS/Finite/RectKraus`, `OQS/Lindblad`, `OQS/NonMarkov`, `OQS/Semigroup`, `OQS/Stinespring`, `Integration/Locality`, `Integration/FunctorialGlue`, `Integration/BulkBoundaryDictionary`, `Integration/BackgroundIndependentCoarseGraining`, `Integration/RunningBoundaryData`
Beispielpfade	ToC-Beispiele, End-to-End-Szenarien, Kraus-/Channel-Beispiele zur Regression

C. Optional / später (nützlich, aber nicht blockierend für den neuen Stamm)

Bereich	Optional / später
Legacy/Harness	alte Closure-Metas mit boundary-first Fokus, example-only Harness-Pfade, `.bak`-Dateien
Spezialfenster spät	explizite Wightman-/Spektral-/Vakuum-Dateien jenseits von `MinkowskiWindow`, sowie `ConformalWindow`, `FLRWSectorWindow`, randdominierte CFT-/VOA-/modulare Spezialfenster, solange diese nur Zielhorizont sind

Bereich	Optional / später
sekundäre Exportschichten	zusätzliche `Exports.lean`-Dateien pro Unterbaum, sobald der Kern stabil ist
tiefe Spezialbeispiele	vollständige Replay-Portierung aller Altbeispiele, solange Bright-Recovery noch nicht formal geschlossen ist

D. Restlückenliste nach Zip-Audit

Die wichtigsten Punkte, die in der ersten CNNA-Fassung **noch fehlten**, sind jetzt klar benennbar:

1. **Top-Level-Interface-Schicht**
2. **Tooling-/Policy-Parität** (`CStarFoundationPolicy` , Check-Skripte)
3. **Examples/BuildAll**
4. **explizite Pillar-D-Ordnerstruktur**
5. **A-Kinematik-/Symmetrie-Seeds** als bewusst frühe Übernahme
6. **BInterfaces-Zusatzmodule** aus A (`CovariantNets` , `GlobalCone` , `RelativeEntropyHook`)
7. **PillarC/Integration/Locality** als expliziter Kandidat
8. **Recovery des** `BoundaryMatrix*` **-Zweigs** nicht nur als Satz, sondern als eigener Arbeitsstrang
9. **Parameter-Closure-Pfad** (`BranchingWitness` , `SpectralDimensionFlow` , `EffectiveLambda` , `HorizonLevel` , `BackreactionFixedPoint`)
10. **Regularisierungs-Closure** als expliziter methodischer Arbeitsstrang
11. **Dualitäts-/Glueing-Strang** (`FunctorialGlue` , `BulkBoundaryDictionary` , `BoundaryStates` , `Defects` , `DualityDictionary`)
12. **spätes conformal/modular window** als klar abgegrenztes Spezialfenster
13. **AS-Kompatibilitätsstrang** (`BackgroundIndependentCoarseGraining` , `CutoffModeCounting` , `SpectralCutoffMap` , `RunningBoundaryData` , `SectorRGSeed`)

XIII. Initiales Repo-Scaffold für CNNA

Dies ist das empfohlene **erste tatsächliche Dateigerüst**. Es ist größer als der minimale inhaltliche Startsatz, aber klein genug, um den Stamm sofort korrekt zu setzen.

```
CNNA/
  Basic.lean
  BuildAll.lean
  All.lean
  ImportPolicy.md
  CStarFoundationPolicy.md
  ArchitecturePolicy.md
  Interface.lean
  PillarA.lean
  PillarB.lean
  PillarC.lean
  PillarD.lean
  tools/
```

```
    check_import_policy.py
    check_cstar_foundation.py
  Core/
    Approximant.lean
    RegionCore.lean
    RegionNet.lean
    InfiniteCarrier.lean
    Selection.lean
    Interfaces.lean
    AQFTHandoff.lean
    Determinism.lean
    TailInterface.lean
    BoundaryPorts.lean
    DirichletLaplacian.lean
    DirichletLaplacianBridge.lean
    BranchPatch.lean
    ComplementSectorFamily.lean
    SectorSplit.lean
    InterfaceKernel.lean
    BranchingWitness.lean
    ParameterClosure.lean
    Exports.lean
  ToC/
    Substrate.lean
    Addresses.lean
    ParentPath.lean
    Cells.lean
    BranchPatchInstances.lean
    ComplementSectorInstances.lean
  PillarA/
    Interface.lean
    All.lean
    BuildAll.lean
  OQS/
    DtN.lean
    DtNStabilized.lean
    GeneralizedDtN.lean
    MultiSchur.lean
    SectorSysEnv.lean
    SectorChannels.lean
    Backreaction.lean
    BackreactionFixedPoint.lean
    RegularizationClosure.lean
  AQFT/
    StarAlgebra.lean
    CStarNorm.lean
    CStarState.lean
    CStarKMSState.lean
    State.lean
    StateNet.lean
    LocalNet.lean
```

```
    ComplementLocalNet.lean
    ComplementStateNet.lean
    InterfaceLocalNet.lean
    GNS.lean
    ComplementGNS.lean
    KMS.lean
    ComplementKMS.lean
    RelativeEntropy.lean
    QuasiLocal/
      Basic.lean
      ComplementClosure.lean
    HaagKastler/
      Basic.lean
      Complement.lean
      Support.lean
      SupportFull.lean
      SplitProperty.lean
  PillarB/
    Interface.lean
    BuildAll.lean
  PillarC/
    Interface.lean
    BuildAll.lean
    OQS.lean
    Integration.lean
  Integration/
    TailElimAsChannel.lean
    DerivedSpacetime.lean
    ComplementInfluence.lean
    SpacetimeOfComplementFamily.lean
    LocalCovarianceSeed.lean
    EffectiveLambda.lean
    HorizonLevel.lean
  PillarD/
    Interface.lean
    BuildAll.lean
    DerivedSpacetime.lean
    ComplementGeometry.lean
    InterfaceCausality.lean
    LocalCovariance.lean
    SpectralDimensionFlow.lean
    MinkowskiWindow.lean
  Gates/
    BranchPatch.lean
    ComplementSectorFamily.lean
    GeneralizedDtN.lean
    ComplementNet.lean
    ComplementState.lean
    ComplementKMS.lean
    InterfaceEntropy.lean
    DerivedSpacetime.lean
```

```
    BuildAll.lean
  Meta/
    OpenPoints.lean
    VacuityAudit.lean
    CriticalPath.lean
    NoLegacyInCriticalPath.lean
    BrightDarkInterfaceSurface.lean
  Examples/
    BuildAll.lean
    ToC_BranchPatch.lean
    ToC_ComplementFamily.lean
    ToC_GeneralizedDtN.lean
    ToC_ComplementNet.lean
    ToC_InterfaceEntropy.lean
    ToC_DerivedSpacetime.lean
```

Minimaler Scaffold-Pass (wirklich zuerst anlegen)

> Dieser Abschnitt beschreibt den **wirklich buildbaren Erstpass**. Der frühere inhaltliche Minimalstartsatz in Abschnitt IX ist nur der fachliche Kern und darf **nicht** mit dem kompilierbaren Repo-Minimum verwechselt werden.

Falls der Stamm **extrem kontrolliert** beginnen soll, zuerst nur diese Dateien anlegen:

- Top-Level: `Basic.lean`, `BuildAll.lean`, `All.lean`, `ImportPolicy.md`, `CStarFoundationPolicy.md`, `ArchitecturePolicy.md`, `Interface.lean`, `PillarA.lean`, `PillarB.lean`, `PillarC.lean`, `PillarD.lean`
- Core: `Approximant.lean`, `RegionCore.lean`, `InfiniteCarrier.lean`, `BranchPatch.lean`, `ComplementSectorFamily.lean`, `SectorSplit.lean`
- ToC: `Addresses.lean`, `ParentPath.lean`, `Cells.lean`
- OQS: `GeneralizedDtNScaffold.lean`
- AQFT: `ComplementLocalNetScaffold.lean`, `ComplementStateNetScaffold.lean`
- Integration: `ComplementInfluence.lean`
- Meta: `CriticalPath.lean`
- Tools: beide Check-Skripte

Danach sofort erstes Build-Gate.

Reihenfolge des allerersten Scaffold-Commits

1. Top-Level + Policies + Tools
2. Core-Minimum
3. ToC-Minimum
4. OQS `GeneralizedDtN`
5. AQFT `ComplementLocalNet` / `ComplementStateNet`
6. Integration `ComplementInfluence`
7. Meta `CriticalPath`
8. `BuildAll` / `All` schließen

Damit ist CNNA nicht nur konzeptionell, sondern auch **repo-strukturell** von Anfang an sauber gesetzt.

XIV. Phase 0/1 – konkretes Scaffold mit Stub-Inhalten

Ziel dieses Abschnitts ist **nicht** die finale Implementation, sondern ein **kompilierbares, streng minimales Anfangsgerüst**. Die Inhalte sind daher bewusst schlank und sollen nur:

1. Importe stabilisieren,
2. Namensräume fixieren,
3. die neuen Kernobjekte setzen,
4. den späteren Migrationspfad offenhalten.

Die folgenden Inhalte sind als **erste Stub-Fassung** zu lesen.

A. Phase 0 – Top-Level, Policies, Tools

> Derived-only-Hinweis: Die folgenden Dateien setzen nur den Scaffold. Sie definieren **nicht** den aktiven Kernpfad; dieser beginnt erst, wenn SourceMap-/DerivedOnly-Audit-Regeln greifen.

`CNNA/Basic.lean`

```
import Mathlib

namespace CNNA

abbrev VersionString := String

def projectName : String := "CNNA"

def projectVersion : VersionString := "0.1.0-phase0"

end CNNA
```

`CNNA/Interface.lean`

```
import CNNA.Basic

namespace CNNA

class HasBrightDarkInterface (α : Type _) where
  brightCarrier : α → Type _
  darkCarrier   : α → Type _
  ifaceCarrier  : α → Type _

end CNNA
```

`CNNA/PillarA.lean`

```
import CNNA.Core.Approximant
import CNNA.Core.RegionCore
import CNNA.Core.InfiniteCarrier
import CNNA.Core.BranchPatch
import CNNA.Core.ComplementSectorFamily
import CNNA.Core.SectorSplit
```

`CNNA/PillarB.lean`

```
import CNNA.AQFT.ComplementLocalNetScaffold
import CNNA.AQFT.ComplementStateNetScaffold
```

`CNNA/PillarC.lean`

```
import CNNA.OQS.GeneralizedDtNScaffold
import CNNA.Integration.ComplementInfluence
```

`CNNA/PillarD.lean`

```
import CNNA.Basic
```

`CNNA/BuildAll.lean`

```
import CNNA.PillarA
import CNNA.PillarB
import CNNA.PillarC
import CNNA.PillarD
```

`CNNA/All.lean`

```
import CNNA.BuildAll
import CNNA.Meta.CriticalPath
```

`CNNA/ImportPolicy.md`

```
- Critical path imports must remain acyclic.
- New files should prefer `CNNA.Basic` over broad umbrella imports.
- Boundary-first legacy modules may only be imported through explicit
recovery files.
```

`CNNA/CStarFoundationPolicy.md`

```
- Reuse proven C*/state/KMS machinery where possible.
- Do not duplicate algebraic foundations merely for renaming.
- New complement/interface layers should sit above, not beside, the stable C*
basis.
```

`CNNA/ArchitecturePolicy.md`

```
- Sector-first, not boundary-first.
- Bright/Dark/Interface are primary architectural carriers.
- Old bright-boundary AQFT is treated as a recoverable specialization.
```

`CNNA/tools/check_import_policy.py`

```
from pathlib import Path

ROOT = Path(__file__).resolve().parents[1]


def main() -> int:
    lean_files = list(ROOT.rglob("*.lean"))
    if not lean_files:
        print("No Lean files found.")
        return 0
    print(f"Checked {len(lean_files)} Lean files for import-policy
scaffold.")
    return 0


if __name__ == "__main__":
    raise SystemExit(main())
```

`CNNA/tools/check_cstar_foundation.py`

```
from pathlib import Path


def main() -> int:
    print("C* foundation policy scaffold check: OK")
    return 0


if __name__ == "__main__":
    raise SystemExit(main())
```

B. Phase 1 – Core / ToC / OQS / AQFT / Integration / Meta

`CNNA/Core/Approximant.lean`

```
import CNNA.Basic

namespace CNNA
namespace Core

structure Approximant (Ω : Type _) where
  carrier : Finset Ω

end Core
end CNNA
```

`CNNA/Core/RegionCore.lean`

```
import CNNA.Basic

namespace CNNA
namespace Core

structure Region (Ω : Type _) where
  cells : Set Ω

namespace Region

variable {Ω : Type _}

instance : SetLike (Region Ω) where
  coe R := R.cells
  coe_injective' := by
    intro A B h
    cases A
    cases B
    cases h
    rfl

end Region

end Core
end CNNA
```

`CNNA/Core/InfiniteCarrier.lean`

```
import CNNA.Basic

namespace CNNA
```

```
namespace Core

class InfiniteCarrier (Ω : Type _) : Prop where
  instInfinite : Infinite Ω

attribute [instance] InfiniteCarrier.instInfinite

end Core
end CNNA
```

`CNNA/Core/BranchPatch.lean`

> Hinweis: In Phase 2 sollte hier zusätzlich die Verzweigungs-Nichttrivialität in Richtung `BranchingWitness` angeschlossen werden.

`CNNA/Core/BranchingWitness.lean`

```
import CNNA.Basic

namespace CNNA
namespace Core

structure BranchingWitness (b : Nat) where
  left  : Fin b
  right : Fin b
  neq   : left ≠ right

end Core
end CNNA
```

`CNNA/Core/ParameterClosure.lean`

```
import CNNA.Core.BranchingWitness

namespace CNNA
namespace Core

inductive ParameterStatus where
  | onticMicro
  | dynamicEmergent
  | numericSecondary

end Core
end CNNA
```

CNNA/Core/BranchPatch.lean

```
import CNNA.Core.Approximant

namespace CNNA
namespace Core

structure BranchPatch (Ω : Type _) where
  anchor      : Ω
  bright      : Finset Ω
  uvCut       : Nat
  irCut       : Nat
  sideProfile : Finset Ω

namespace BranchPatch

variable {Ω : Type _}

protected def toApproximant (p : BranchPatch Ω) : Approximant Ω :=
  ⟨p.bright⟩

theorem bright_finite (p : BranchPatch Ω) : (↑p.bright : Set Ω).Finite := by
  exact Finset.finite_toSet _

end BranchPatch

end Core
end CNNA
```

CNNA/Core/ComplementSectorFamily.lean

```
import CNNA.Core.BranchPatch

namespace CNNA
namespace Core

structure ComplementSectorFamily (Ω : Type _) where
  patch        : BranchPatch Ω
  commonTrunk  : Finset Ω
  privateIR    : Finset Ω
  sideBranches : Finset (Finset Ω)
  ownUVTails   : Finset (Finset Ω)

namespace ComplementSectorFamily

variable {Ω : Type _}

abbrev darkSupport (C : ComplementSectorFamily Ω) : Set Ω :=
  ↑C.commonTrunk ∪ ↑C.privateIR
```

```
theorem darkSupport_def (C : ComplementSectorFamily Ω) :
    C.darkSupport = (↑C.commonTrunk ∪ ↑C.privateIR : Set Ω) := rfl

end ComplementSectorFamily

end Core
end CNNA
```

CNNA/Core/SectorSplit.lean

```
import CNNA.Basic

namespace CNNA
namespace Core

structure SectorSplit (V : Type _) where
  Bright     : Type _
  Trunk      : Type _
  Side       : Type _
  Tail       : Type _
  totalEquiv : V ≃ (((Bright ⊕ Trunk) ⊕ Side) ⊕ Tail)

end Core
end CNNA
```

CNNA/ToC/Addresses.lean

```
import CNNA.Basic

namespace CNNA
namespace ToC

abbrev Address := List Nat

end ToC
end CNNA
```

CNNA/ToC/ParentPath.lean

```
import CNNA.ToC.Addresses

namespace CNNA
namespace ToC

def parentPath : Address → Address
  | [] => []
```

```
  | xs => xs.dropLast

end ToC
end CNNA
```

`CNNA/ToC/Cells.lean`

```
import CNNA.ToC.Addresses

namespace CNNA
namespace ToC

structure Cell where
  addr : Address

end ToC
end CNNA
```

`CNNA/OQS/GeneralizedDtNScaffold.lean`

`CNNA/OQS/BackreactionFixedPoint.lean`

```
import CNNA.Basic

namespace CNNA
namespace OQS

structure BackreactionFixedPoint (α : Type _) where
  carrier : α
  selfConsistent : Prop

end OQS
end CNNA
```

`CNNA/OQS/RegularizationClosure.lean`

```
import CNNA.Basic

namespace CNNA
namespace OQS

inductive RegularizationStatus where
  | eliminated
  | secondaryDerived

end OQS
end CNNA
```

`CNNA/OQS/GeneralizedDtNScaffold.lean`

```
import CNNA.Core.ComplementSectorFamily
import CNNA.Core.SectorSplit

namespace CNNA
namespace OQS

structure GeneralizedDtNData (𝕜 : Type _) (Ω : Type _) where
  sectors    : Core.ComplementSectorFamily Ω
  stabilized : Bool

class HasGeneralizedDtN (𝕜 : Type _) (Ω : Type _) where
  generalizedDtN : GeneralizedDtNData 𝕜 Ω → Type _

end OQS
end CNNA
```

`CNNA/AQFT/ComplementLocalNetScaffold.lean`

```
import CNNA.Core.RegionCore

namespace CNNA
namespace AQFT

structure ComplementLocalNet (Region Alg : Type _) where
  toAlg    : Region → Alg
  isotony  : Prop
  locality : Prop

end AQFT
end CNNA
```

`CNNA/AQFT/ComplementStateNetScaffold.lean`

```
import CNNA.AQFT.ComplementLocalNet

namespace CNNA
namespace AQFT

structure ComplementStateNet (Region Alg State : Type _) where
  net     : ComplementLocalNet Region Alg
  toState : Region → State

end AQFT
end CNNA
```

`CNNA/Integration/ComplementInfluence.lean`

`CNNA/Integration/EffectiveLambdaSeed.lean`

```
import CNNA.Basic

namespace CNNA
namespace Integration

structure EffectiveLambda where
  value : Rat

end Integration
end CNNA
```

`CNNA/Integration/HorizonLevelSeed.lean`

```
import CNNA.Basic

namespace CNNA
namespace Integration

structure HorizonLevel where
  value : Nat

end Integration
end CNNA
```

`CNNA/Integration/ComplementInfluence.lean`

```
import CNNA.Core.ComplementSectorFamily

namespace CNNA
namespace Integration

structure ComplementInfluence (Ω : Type _) where
  sectors    : Core.ComplementSectorFamily Ω
  influences : Set (Ω × Ω)

end Integration
end CNNA
```

`CNNA/PillarD/SpectralDimensionFlowSeed.lean`

```
import CNNA.Basic

namespace CNNA
```

```
namespace PillarD

structure SpectralDimensionFlow where
  sampleTimes : List Rat
  sampleValues : List Rat

end PillarD
end CNNA
```

`CNNA/Meta/CriticalPath.lean`

```
import CNNA.PillarA
import CNNA.PillarB
import CNNA.PillarC
import CNNA.PillarD

namespace CNNA
namespace Meta

abbrev CriticalPathReady : Prop := True

theorem criticalPathReady : CriticalPathReady := by
  trivial

end Meta
end CNNA
```

C. Hinweise zur ersten Lean-Runde

1. Diese Stubs setzen **bewusst nur Namen und Abhängigkeiten**.
2. Theoreme mit echtem mathematischen Gehalt kommen **erst ab Phase 2/3**.
3. `ParentPath.lean` ist hier nur ein Platzhalter; die echte ToC-Elternpfadlogik muss aus dem Altblock sauber portiert werden.
4. `ComplementLocalNet.isotony` und `.locality` sind absichtlich noch `Prop`; erst nach Stabilisierung der Regionssprache sollten dort konkrete Formulierungen stehen.
5. `HasGeneralizedDtN` ist nur ein Scaffold-Hook; die echte Signatur sollte erst nach Entscheidung über Matrix-/Operatorbasis fixiert werden.
6. Alle hier verwendeten generischen Platzhalter (`Region`, `Alg`, `State`, freie Scalar-Container) müssen vor Eintritt in den aktiven Kernpfad durch SourceMap-/Derived-only-Regeln zurückgebunden werden.

D. Erster Build-Gate-Satz

Sobald diese Dateien stehen, sollte der erste Gate-Satz lauten:

- alle Dateien importieren ohne Zyklus,
- `CNNA/BuildAll.lean` baut,
- `CNNA/All.lean` baut,
- `CNNA.Meta.criticalPathReady` ist provabel,

- keine Platzhalterdatei importiert boundary-first Legacy direkt.

Das ist das richtige Ende von **Phase 0/1**.