Formale Ableitungskette des strikten ToC → Pillar B Rich/Closure/State-Dynamics-Pfads

Codebasis: `REALOQS_v0.0604`

Aktiver Pfad: ToC → Ω, L, β → `N_ToC` → ρ`TopStrong`/ρ`Top`/ρ`Region` → `Φ_FromA` → `quasiLocalClosureIsoMatrix_FromA` → `GNS` → Heisenberg → KMS → Modular

Ziel. Dieses Dokument dokumentiert den in `REALOQS_v0.0604` tatsächlich formalisierten strikten endlichen ToC→Pillar-B-Matrixpfad bis `modularDatum_top_ToC`. Jede starke Aussage wird auf eine definierende Lean-Konstante oder einen benannten Satz zurückgeführt.

Wichtige Präzisierung. Der Pfad ist im Code *nicht* linear, sondern ein Fork-Join-Graph: aus dem ToC-Handoff entstehen Zustands-, Abschluss-, GNS- und Dynamik/KMS-Zweige, die erst im Modular-Datum wieder zusammenlaufen.

Strikter Geltungsbereich. Bewiesen wird eine *endliche typ-I C^*-dynamische Netzstruktur auf dem exportierten Randträger*. Nicht behauptet werden continuum-AQFT, Wightman-Axiome, Reeh-Schlieder, Typ-III-Lokalität oder ein Kontinuumslimes.

Editor:	Jan Seeck (antaris)
Ghostwriter:	OpenAI ChatGPT
Grundlage:	hochgeladene Quelle `REALOQS_v0.0604.zip`
Datum:	23. März 2026

Inhaltsverzeichnis

1 Ziel, Geltungsbereich und methodische Vorbemerkungen **2**
1.1 Was in diesem Dokument als “vollständig” gemeint ist . . . 2
1.2 Die korrekte Form des Pfads . . . 2
1.3 Zwei kritische Präzisierungen aus der Codebasis . . . 2
1.4 Voraussetzungsoberfläche des strikten Pfads . . . 3

2 Die primitive ToC-Konstruktion in Pillar A **3**
2.1 Adressraum und endliche Approximanten . . . 3
2.2 Die minimale primitive Eingabe . . . 4
2.3 Kanonische Rand-/Innenzerlegung des Vertexraums . . . 4
2.4 Adjazenz, Kantengewichte und Laplace-Operator . . . 5

3 Die strikte A-seitige Operatorpipeline **6**
3.1 Blockzerlegung des Laplacians . . . 6
3.2 Stabilisierter Innenblock und Nichtsingularität . . . 6
3.3 Laplacian-abgeleiteter Zustand st_{Lap} . . . 7
3.4 Der echte DtN-/Kron-Reduktionsschritt . . . 8

4 Die strikte A-seitige Exportoberfläche **8**
4.1 Exportierter Träger und exportierter Randoperator . . . 8
4.2 Symmetrie von L . . . 9
4.3 A-seitige diagnostische Materiedaten . . . 10

5 Re-Exposition in Pillar B: $\Omega, L, T, N_{\mathrm{ToC}}$ **10**
5.1 Re-Export von Ω und L . . . 10
5.2 Das aktive lokale Netz . . . 11

6 Der Zustandszweig **11**
6.1 Der starke Gibbs-Zustand auf der Vollmatrixalgebra . . . 12
6.2 Topzustand und regionale Zustände . . . 13
6.3 Globaler Zustandskegel und quasilokaler Zustand . . . 14

7 Der Abschlusszweig **15**
7.1 Die Topfaser ist die volle Matrixalgebra . . . 15
7.2 Die quasilokale Algebra ist die Topfaser und damit die Vollmatrixalgebra . . . 15

8 Der GNS-Zweig **16**
8.1 Sesquilinearform und Darstellung . . . 16
8.2 Zyklischer Vektor . . . 16
8.3 Das konkrete GNS-Datum . . . 16

9 Der Dynamikzweig **17**
9.1 Volle Heisenberg-Dynamik und Zeitorientierung . . . 17
9.2 Transport auf die Topfaser . . . 17

10 Der KMS-Zweig **18**
10.1 Gibbs-Zustand auf Vollmatrixniveau . . . 18
10.2 Die formale KMS-Strip-Bedingung . . . 18
10.3 Der explizite Matrix-KMS-Kern . . . 19
10.4 Transport des KMS-Zustands auf die Topfaser . . . 20

11 Der Modular-Join **20**
11.1 Struktur des Modular-Datums 20

12 Die vollständige Ableitungskette als präziser Graph **21**

13 Gesamtsatz des strikten Pfads **22**

14 Was dieser Pfad beweist – und was nicht **23**

A Dateien und Lastträger des Pfads **23**

B Meta-Marker außerhalb des unmittelbaren Rich/State-Dynamics-Pfads **25**

1 Ziel, Geltungsbereich und methodische Vorbemerkungen

Der hier behandelte Pfad ist *nicht* der additive Observable-Subnetz-Pfad `N_add_ToC`, sondern der aktive reiche Vollmatrix-/Abschluss-/Zustands-/Dynamikpfad auf `N_ToC`. Das Endobjekt ist

`modularDatum_top_ToC`.

1.1 Was in diesem Dokument als "vollständig" gemeint ist

"Vollständig" bedeutet hier: Alle Objekte und Brückensätze, die auf dem strikten Pfad von der ToC-Konstruktion bis `modularDatum_top_ToC` tatsächlich getragen werden, werden explizit definiert und in ihrer logischen Abhängigkeit bewiesen. Nicht erneut abgedruckt werden dagegen die transitive Gesamtheit aller allgemeinen Hintergrundresultate aus Mathlib oder aus den generischen AQFT-Framework-Dateien, sofern der Pfad diese nur als bereits bewiesene Basissätze benutzt.

Das ist keine Ausrede, sondern eine mathematische Präzisierung. Ein kritischer Leser muss unterscheiden zwischen

1. der *Pfadvollständigkeit* des abgeleiteten Objekts und
2. der *transitiven Vollentfaltung* aller benutzten Bibliotheksgrundlagen.

Hier wird Ersteres vollständig geleistet. Wer die gesamte transitive Kette bis in jede Hilfsbibliothek hinein sehen will, muss zusätzlich die in den jeweiligen Beweisen zitierten Basissätze lesen; innerhalb des dokumentierten Pfads bleibt jedoch keine unbenannte Lücke offen.

1.2 Die korrekte Form des Pfads

Der Pfad ist im Code architektonisch zusammenhängend, aber nicht als bloße lineare Verkettung implementiert. Ab dem A-seitigen Export von Ω, L und β verzweigt die Konstruktion:

Zustandszweig: $\rho_{\mathrm{top}}^{\mathrm{strong}} \to \rho_{\mathrm{top}}, \rho_r \to \mathrm{cone}_{\mathrm{FromA}} \to \Phi_{\mathrm{FromA}}$,
Abschlusszweig: $N_{\mathrm{ToC}} \to$ Top-Isomorphie $\to$ quasilokale Isomorphie,
GNS-Zweig: $\rho_{\mathrm{top}} \to$ GNS,
Dynamikzweig: $L \to \alpha^-$,
KMS-Zweig: $(L, \beta, \alpha^-) \to$ KMS,
Join: (matrixKMS, topKMS, GNS) $\to$ modularDatum_top_ToC.

1.3 Zwei kritische Präzisierungen aus der Codebasis

Bemerkung 1.1 (Aktive Temperatur)**.** In `TreeOfCliquesAQFTHandoff.lean` existieren *zwei* Temperaturbegriffe:

(a) ein früher semantischer Begriff `beta(p)(st)=1+signalpst` mit

`signal(p)(st)=MatrixNorm.frob((Semp).boundaryOpst)`,

also einer Frobenius-Norm des exportierten Randoperators eines konkreten Zustands `st`;

(b) der für den strikten ToC→Pillar-B-Pfad tatsächlich benutzte Begriff

`betaFromA(p)=1+|signalFromA(p)|`,

wobei `signalFromA` über `signalAt` aus der kanonischen Skalenachse und damit aus `breakMeasure` gewonnen wird.

Diese beiden Begriffe sind also strukturell verschieden. Auf dem hier behandelten Pfad ist ausschließlich `betaFromA` aktiv, denn genau dieser Wert wird in `TreeOfCliquesFromPillarA.T` in die Matter-Template-Komponente β eingesetzt.

Bemerkung 1.2 (GNS wird nicht aus Φ_{FromA} konstruiert)**.** Der quasilokale Zustand Φ_{FromA} gehört architektonisch zum selben Pfad, aber `gnsDatum_top_ToC` wird definitorisch direkt aus dem Topzustand ρ`Top` gebaut, nicht erst durch Rückgriff auf `Φ_FromA` oder auf die quasilokale Abschlussisomorphie.

1.4 Voraussetzungsoberfläche des strikten Pfads

Damit der Text keine stillschweigenden Voraussetzungen versteckt, werden die im Code tatsächlich verwendeten Annahmen hier explizit notiert.

1. Global fixiert sind ein Verzweigungsfaktor $b : \mathbb{N}$ und eine Policy p : `Policy` b.
2. Die reine ToC-Konstruktion, der Export von Ω und L sowie der Abschlusszweig bis `quasiLocalClosureIsoMatrix_FromA` benötigen keine Positivitätsannahme $0 < b$.
3. Ab dem starken Zustandszweig wird im Code die Instanz `[Fact(0<b)]` verwendet. Sie liefert insbesondere `Nonempty(\Omega)` über `instNonemptyOmegaOfPosBase`.
4. Das GNS-Datum `boundaryMatrixTopGNSDatum`, das Top-KMS-Datum `boundaryMatrixTopKMSMinusAtBeta` und damit auch `modularDatum_top_ToC` leben daher formal im Regime $0 < b$.
5. Zusätzlich wird für den KMS-Zweig $0 < \beta$ benötigt; auf der ToC-Oberfläche ist dies exakt der Satz `betaFromA_pos`.

Theorem 1.3 (Präziser Endsatz des Dokuments)**.** *Unter den Annahmen $b > 0$ und p : `Policy` b existiert auf der Topfaser des aktiven Netzes `N_ToC` ein wohldefiniertes Modulardatum*

$$\mathit{modularDatum\_top\_ToC}(b, p),$$

dessen Komponenten genau der starke Matrix-KMS-Zweig, der transportierte Top-KMS-Zweig und das konkrete GNS-Datum des Topzustands sind.

Beweis. Dies ist keine zusätzliche mathematische Behauptung, sondern die exakte Spezialisierung der Struktur `BoundaryMatrixModularDatum` auf die aus Pillar A exportierten Daten L und $\beta = (Tp).\beta$. Formal ist dies die Definition `modularDatum_top_ToC` zusammen mit den Identitäten `modularDatum_top_ToC_matrixKMS`, `modularDatum_top_ToC_topKMS` und `modularDatum_top_ToC_gns`. □

2 Die primitive ToC-Konstruktion in Pillar A

2.1 Adressraum und endliche Approximanten

Die primitive IDEAL-Adressstruktur ist in `PillarA/Ideal/Substrate.lean` als

$$\mathrm{Addr}_b := \mathrm{List}(\mathrm{Fin}\, b)$$

definiert. Ein ToC-Zellindex ist also ein endliches Wort über einem Alphabet der Größe b.

Die Mengen der Adressen auf fester Tiefe bzw. bis zu einer Tiefe werden rekursiv definiert durch

$$\mathrm{cellsAtLevel}(b, k) := \{a \in \mathrm{Addr}_b \mid |a| = k\}, \tag{1}$$

$$\text{cellsUpTo}(b, 0) := \text{cellsAtLevel}(b, 0), \tag{2}$$
$$\text{cellsUpTo}(b, L+1) := \text{cellsUpTo}(b, L) \cup \text{cellsAtLevel}(b, L+1). \tag{3}$$

Formal ist `cellsAtLevel` als Bild von $\operatorname{Fin} k \to \operatorname{Fin} b$ unter `List.ofFn` implementiert; der zentrale Satz dazu ist

$$\texttt{mem\_cellsAtLevel\_iff\_length}: \qquad a \in \text{cellsAtLevel}(b, k) \iff |a| = k.$$

Definition 2.1 (ToC-Vertex, Rand, Inneres)**.** Für festes b und k definiert die Codebasis:

$$\begin{aligned} V_k &:= \text{Vertex}(b, k) = \text{cellsUpTo}(b, k), \\ B_k &:= \text{Boundary}(b, k) = \text{cellsAtLevel}(b, k), \\ I_k &:= \text{Interior}(b, k) = \text{cellsUpTo}(b, k) \setminus \text{cellsAtLevel}(b, k). \end{aligned}$$

Das sind genau die Lean-Objekte `Vertexbk`, `Boundarybk` und `Interiorbk`.

Proposition 2.2 (Endlicher ToC-Approximant)**.** *Für eine Policy p mit $k := p.L_{\max}$ ist der ToC-Approximant ein vollständig endlicher Träger*

$$V = V_k, \qquad B = B_k, \qquad I = I_k.$$

Alle späteren Operatoren werden auf diesen endlichen Typen konstruiert.

Beweis. Die Definitionen `Vertex`, `Boundary` und `Interior` sind Subtypen endlicher `Finset`-Träger. Da `cellsUpTo` und `cellsAtLevel` per Konstruktion endlich sind, sind auch V, B und I endlich. Genau diese Typen werden in `TreeOfCliquesApproxDtNStabilized.lean` als `V`, `B` und `I` abgekürzt. □

2.2 Die minimale primitive Eingabe

Die primitive Eingabe des strikten Pfads ist in `PillarA/Core/Policy.lean` bewusst minimal gehalten:

```
structure Policy (b : Nat) where
  L_max : Nat
```

Definition 2.3 (Policy)**.** Für fixen Verzweigungsfaktor b besteht die primitive Policy allein aus dem Trunkierungsregulator $L_{\max}$. Alle übrigen Größen des strikten Pfads werden kanonisch daraus abgeleitet.

Bemerkung 2.4. Diese Minimalität ist mathematisch relevant: Der Pfad ist nicht mit einer Sammlung freier physikalischer Parameter gestartet, sondern mit einem einzigen endlichen Regulator. Das heißt nicht, dass das resultierende Modell bereits physikalisch vollständig ist; wohl aber, dass die im strikten Pfad auftretenden Größen nicht zusätzlich hineingeschmuggelt werden.

2.3 Kanonische Rand-/Innenzerlegung des Vertexraums

`PillarA/Ideal/TreeOfCliques/Split.lean` konstruiert aus der Bedingung "Tiefe gleich k" eine Zerlegung

$$V_k \cong B_k \sqcup I_k.$$

Der Vorwärtsweg `toSum` testet, ob die Adresse Länge k hat; der Rückweg `fromSum` ist die kanonische Einbettung von Rand bzw. Innerem in den Gesamtraum.

Proposition 2.5 (Kanonische Split-Äquivalenz)**.** *Es gibt eine kanonische Äquivalenz*

$$\texttt{equivVertexSum} : V_k \simeq B_k \sqcup I_k,$$

und daraus den OQS-Split

$$\texttt{split(bk)} : \mathit{Split}(V_k).$$

Beweis. Die Vorwärtsabbildung ist definiert durch

$$\texttt{toSum}(v) = \begin{cases} \mathrm{inl}(v), & |v| = k, \\ \mathrm{inr}(v), & |v| \neq k. \end{cases}$$

Der Fall $|v| \neq k$ wird in `mem_interiorSet_of_not_length` exakt als Mitgliedschaft in I_k nachgewiesen. Die Rückabbildung `fromSum` benutzt `boundaryEmbedding` bzw. `interiorEmbedding`. In `equivVertexSum` werden dann beide Inversen explizit per Fallunterscheidung bewiesen. `split(bk)` ist lediglich die Bündelung dieser Äquivalenz in der Struktur `PillarA.OQS.Split`. □

2.4 Adjazenz, Kantengewichte und Laplace-Operator

Die ToC-Adjazenz ist in `PillarA/Ideal/TreeOfCliques/Edges.lean` rein kombinatorisch definiert. Formal gilt auf Adressen a_1, a_2, p, c:

$$\begin{aligned} \mathrm{parentRel}(p, c) &:\iff |p| + 1 = |c| \mathit{extundp} = \mathrm{dropLast}(c), \\ \mathrm{parentEdge}(a_1, a_2) &:\iff \mathrm{parentRel}(a_1, a_2) \vee \mathrm{parentRel}(a_2, a_1), \\ \mathrm{siblingEdge}(a_1, a_2) &:\iff a_1 \neq a_2 \wedge |a_1| = |a_2| \wedge \mathrm{dropLast}(a_1) = \mathrm{dropLast}(a_2), \\ \mathrm{adjAddr}(a_1, a_2) &:\iff \mathrm{parentEdge}(a_1, a_2) \vee \mathrm{siblingEdge}(a_1, a_2). \end{aligned}$$

Auf Vertex-Ebene gilt dann

$$\texttt{Adj}(x, y) :\iff \mathrm{adjAddr}(x.1, y.1).$$

Das kanonische Gewicht ist ungewichtet:

$$\mathrm{conductance}(x, y) = \begin{cases} 1, & \text{falls } x \sim y, \\ 0, & \text{sonst.} \end{cases}$$

Lemma 2.6 (Symmetrie und Nichtnegativität der Leitfähigkeit)**.** *Für alle Vertex-Indizes x, y gilt*

$$\mathrm{conductance}(x, y) = \mathrm{conductance}(y, x), \qquad \mathrm{conductance}(x, y) \geq 0.$$

Dies sind genau die Sätze `conductance_symm` und `conductance_nonneg`.

Beweis. Die Symmetrie folgt aus der Symmetrie von `parentEdge` und `siblingEdge`; der Code beweist zuerst `adjAddr_symm` und dann `Adj_symm` auf Vertex-Ebene. Die Gewichte sind per Definition nur 0 oder 1, also nichtnegativ. □

Definition 2.7 (ToC-Laplacian)**.** Der Laplace-Operator des endlichen ToC-Approximanten ist

$$L_0 := \texttt{TreeOfCliques.laplacian}(b, k) = \texttt{PillarA.LayerA.laplacian}(c),$$

mit $c = \mathrm{conductance}$.

Proposition 2.8 (Admissibilität von L_0)**.** *Der Satz `gate_L0_admissible` liefert*

$$\mathit{AdmissibleOp}(L_0).$$

Dabei bedeutet

$$\mathrm{AdmissibleOp}(\Lambda) :\iff \Lambda^{\mathsf{T}} = \Lambda \ \wedge \ \forall i : \ \sum_j \Lambda_{ij} = 0 \ \wedge \ \forall i \neq j : \ \Lambda_{ij} \leq 0.$$

Insbesondere ist L_0 symmetrisch, zeilensummenneutral und außerhalb der Diagonale nichtpositiv.

Beweis. `gate_L0_admissible` reduziert per `simpa` auf den allgemeinen Satz `gate_LAPLACIAN_ADMISSIBLE_treeOfCliques`. Dessen Voraussetzungen sind genau die zuvor bewiesene Symmetrie und Nichtnegativität der Leitfähigkeit. Also ist der Laplace-Operator admissibel; die erste Komponente dieser Admissibilität ist die Symmetrie $L_0^{\mathsf{T}} = L_0$, die zweite die Kirchhoff-Zeilensummenbedingung, und die dritte die Nichtpositivität aller Offdiagonaleinträge. □

3 Die strikte A-seitige Operatorpipeline

3.1 Blockzerlegung des Laplacians

Über den Split $V \cong B \sqcup I$ werden in `TreeOfCliquesApproxDtNStabilized.lean` die vier Blockmatrizen

$$L_{BB}^{(0)}, \qquad L_{II}^{(0)}, \qquad L_{BI}^{(0)}, \qquad L_{IB}^{(0)}$$

als stabile Abkürzungen `LBB0`, `LII0`, `LBI0`, `LIB0` definiert.

3.2 Stabilisierter Innenblock und Nichtsingularität

Der zu eliminierende Innenblock wird nicht roh invertiert, sondern zuerst symmetrisiert und stabilisiert. Formal:

$$\mathrm{symm}(A) := \frac{1}{2}(A + A^{\mathsf{T}}),$$

$$\mathrm{stabilize}(A, \delta) := A + \delta\,\mathbf{1},$$

$$\delta_{\mathrm{SPD}}(A) := p.\mathrm{eps} \cdot \max(1, \|A\|_F),$$

$$L_{II}^{\mathrm{stab}} := \mathrm{stabilize}\Big(\mathrm{symm}(L_{II}^{(0)}), \delta_{\mathrm{SPD}}(\mathrm{symm}(L_{II}^{(0)}))\Big) = \texttt{LII\_stabilized}(L_0).$$

Der Code setzt

$$\texttt{hdet0} : \det(L_{II}^{\mathrm{stab}}) \neq 0, \qquad \texttt{hpsd0} : \mathrm{symm}(L_{II}^{(0)}) \text{ ist positiv semidefinit.}$$

Theorem 3.1 (Positive Semidefinität des Innenblocks)**.** *Der Satz* ***`hpsd0_proved`*** *zeigt:*

$$\mathrm{symm}(L_{II}^{(0)}) \succeq 0.$$

Beweis. Der Satz `hpsd0_proved` wird im Code durch eine explizite Kette bereits bewiesener Zwischensätze aufgebaut; für die logische Last sind genau die folgenden vier Schritte entscheidend.

Schritt 1: Aus `gate_L0_admissible` folgt die Symmetrie von L_0, und mit `blockII_transpose_of_symm` folgt

$$(L_{II}^{(0)})^{\mathsf{T}} = L_{II}^{(0)}.$$

Mit `symm_LII0_eq` erhält man also

$$\mathrm{symm}(L_{II}^{(0)}) = L_{II}^{(0)}.$$

Die in den folgenden beiden Schritten benutzte quadratische Form ist in `DirichletLaplacian.lean` explizit definiert durch

$$\mathrm{quadForm}(A, f) := \sum_i f(i)\,(Af)(i) = \sum_{i,j} f(i) A_{ij} f(j)$$

auf dem reellen Funktionenraum über dem endlichen Träger.

Schritt 2: Der Code beweist in `quadForm_L0_nonneg`, dass die quadratische Form von L_0 nichtnegativ ist. Dafür wird der ToC-Laplacian auf die Dirichlet-Energie zurückgeführt:

$$\mathcal{E}_c(f) = \mathrm{quadForm}(L_0, f) \geq 0,$$

weil die Leitfähigkeit c nichtnegativ ist.

Schritt 3: In `quadForm_ext_eq_quadForm_blockII` wird eine Rand-Null-Erweiterung konstruiert. Für einen Innenvektor $x : I \to \mathbb{R}$ definiert man eine Funktion $f : V \to \mathbb{R}$, die auf B verschwindet und auf I gleich x ist. Dann gilt

$$\mathrm{quadForm}(L_0, f) = \mathrm{quadForm}(L_{II}^{(0)}, x).$$

Schritt 4: Aus der Nichtnegativität von $\mathrm{quadForm}(L_0, f)$ folgt also die Nichtnegativität von $\mathrm{quadForm}(L_{II}^{(0)}, x)$ für alle x. Zusammen mit der bereits gezeigten Symmetrie ergibt das positive Semidefinität von $L_{II}^{(0)}$, also auch von $\mathrm{symm}(L_{II}^{(0)})$. Die im Code dabei verwendete Brücke zwischen reeller quadratischer Form und der komplexwertigen Relation `Matrix.PosSemidef` ist das abschließende `simpa[quadForm]`: die Einträge liegen reell vor und werden nur über die kanonische Einbettung $\mathbb{R} \hookrightarrow \mathbb{C}$ in die C*-algebraische Positivitätsaussage überführt. □

Korollar 3.2 (Determinanten-Gate)**.** *Aus* `hpsd0_proved` *folgt mit* `hdet0_of_hpsd0` *und* `hdet0_inst` *die kanonische Nichtsingularität*

$$\det(L_{II}^{\mathrm{stab}}) \neq 0.$$

Beweis. Der generische Satz `det_LII_stabilized_ne_zero_of_posSemidef` ist mathematisch genau die Aussage

$$A \succeq 0 \text{ und } \delta > 0 \quad \Longrightarrow \quad A + \delta I \succ 0 \quad \Longrightarrow \quad A + \delta I \text{ ist invertierbar} \quad \Longrightarrow \quad \det(A + \delta I) \neq 0.$$

Im Lean-Beweis wird dies in derselben Reihenfolge abgearbeitet: Die Einheitsmatrix ist positiv definit, also auch δI für $\delta > 0$; aus *positiv semidefinit* plus *positiv definit* folgt wieder *positiv definit*; positiv definite Matrizen sind Einheiten, also ist auch ihre Determinante eine Einheit und damit von null verschieden. `hdet0_of_hpsd0` instanziiert genau dieses Schema für den ToC-Laplacian L_0, und `hdet0_inst` ist nur noch die kanonische Abkürzung des so erhaltenen Determinanten-Gates. □

3.3 Laplacian-abgeleiteter Zustand st_{Lap}

Die Struktur `ReducedOpState` speichert die vier Blöcke $L_{BB}, L_{II}, L_{BI}, L_{IB}$ und zusätzlich den Nachweis

$$\det\Big(\mathrm{stabilize}(\mathrm{symm}(L_{II}), \delta_{\mathrm{SPD}}(\mathrm{symm}(L_{II})))\Big) \neq 0,$$

also genau die numerisch abgesicherte Invertierbarkeit des stabilisierten Innenblocks. Dabei ist δ_{SPD} keine zusätzliche physikalische Größe, sondern die im Code deterministisch aus der universellen numerischen Schranke `epsFloor64` – in manchen Definitionen nur über die punktweise Schreibweise `p.eps` angesprochen, aber nicht als eigenes Policy-Feld – und der Frobenius-Norm konstruierte Jittergröße; formal ist dies `PillarA.OQS.Split.DtNStabilized.deltaSPD`. Der aus dem ToC-Laplacian kanonisch gewonnene Zustand ist

$$\texttt{stLap} := \texttt{ofLaplacian}(\mathit{hdet0\_inst}).$$

Proposition 3.3 (Rekonstruktion des vollen Operators)**.** *Es gilt*

$$\mathit{fullMatrix\_stLap}: \qquad \mathrm{fullMatrix}(st_{\mathrm{Lap}}) = L_0.$$

Beweis. Die allgemeine Rekonstruktionsaussage ist `fullMatrix_ofLaplacian`. Dort wird zunächst in `fullMatrix_ofMatrix` durch explizite Fallunterscheidung über die Summentypzerlegung $V \cong B \sqcup I$ gezeigt, dass das Zusammenbauen der vier Blockmatrizen genau die Ursprungsmatrix zurückliefert. `fullMatrix_stLap` ist dann nur noch die Spezialisierung auf L_0 mittels `simpa[stLap]`. □

3.4 Der echte DtN-/Kron-Reduktionsschritt

Der entscheidende Schritt ist *nicht* mehr die triviale Identität, sondern eine echte stabilisierte Schur-Komplement-Reduktion. Für einen Zustand st lautet sie

$$\mathrm{reduce}(st).L_{II} = st.L_{II}, \tag{4}$$
$$\mathrm{reduce}(st).L_{BB} = \mathrm{DtN}^{\mathrm{stab}}(\mathrm{fullMatrix}(st)), \tag{5}$$
$$\mathrm{reduce}(st).L_{BI} = 0, \tag{6}$$
$$\mathrm{reduce}(st).L_{IB} = 0. \tag{7}$$

Der verwendete stabilisierte DtN-Operator ist formal

$$\mathrm{DtN}^{\mathrm{stab}}(L) = L_{BB} - L_{BI}\,(L_{II}^{\mathrm{stab}})^{-1} L_{IB}.$$

Im Code ist das `PillarA.OQS.Split.DtNStabilized.DtN_stabilized_of`.

Proposition 3.4 (Reduktion ist idempotent)**.** *Für jeden reduzierten Zustand st gilt*

$$\textit{reduce\_idem}: \qquad \mathrm{reduce}(\mathrm{reduce}(st)) = \mathrm{reduce}(st).$$

Beweis. Nach einer Reduktion sind die Kreuzblöcke per Definition null. Für einen solchen Zustand vereinfacht sich die stabilisierte Schur-Komplement-Formel sofort zu

$$L_{BB} - L_{BI}\,(L_{II}^{\mathrm{stab}})^{-1} L_{IB} = L_{BB} - 0 \cdot (L_{II}^{\mathrm{stab}})^{-1} \cdot 0 = L_{BB}.$$

Genau dies ist `dtn_stabilized_fullMatrix_of_cross_zero`. `dtn_stabilized_fullMatrix_of_reduced` instanziiert die Aussage auf `reducest`. Danach stimmen in $\mathrm{reduce}(\mathrm{reduce}(st))$ und $\mathrm{reduce}(st)$ alle vier Blöcke wörtlich überein; also folgt komponentenweise `reduce_idem`. □

Theorem 3.5 (End-to-end-Satz der A-seitigen Reduktion)**.** *Für den Laplacian-abgeleiteten Zustand gilt*

$$\textit{reduce\_LBB\_eq\_dtn\_stab}: \qquad (\,\mathrm{reduce}(st_{\mathrm{Lap}})\,).L_{BB} = \mathrm{dtn\_stabilized\_of\_det}(b, p, k).$$

ausgesprochen: Die A-seitige Reduktion liefert genau den stabilisierten ToC-Randoperator.

Beweis. Formal reduziert sich der Satz auf eine definitorische Identifikation derselben stabilisierten Schur-Komplementformel auf beiden Seiten. Die benutzten Rewrite-Anker sind `reduce`, `stLap`, `ofLaplacian`, `TreeOfCliques.DtNStabilized.dtn_stabilized_of_det`, `PillarA.OQS.Split.DtNStabilized.DtN_stabilized_of`, `interiorInvStabilized_of_det`, `LII_stabilized`, `invStab_inv`, `PillarA.LayerA.DtN`. Inhaltlich steckt dahinter nur eine Aussage: beide Seiten sind dieselbe stabilisierte Schur-Komplementformel für denselben aus L_0 extrahierten Blockoperator. □

4 Die strikte A-seitige Exportoberfläche

Die Exportoberfläche liegt in `PillarA/Ideal/Adapter/TreeOfCliquesAQFTHandoff.lean`.

4.1 Exportierter Träger und exportierter Randoperator

Für eine feste Policy p werden definiert:

$$\Omega := B_k, \tag{8}$$
$$\mathrm{st0} := st_{\mathrm{Lap}}, \tag{9}$$
$$\mathrm{st1} := \mathrm{reduce}(\mathrm{st0}), \tag{10}$$
$$L := \mathrm{st1}.L_{BB}. \tag{11}$$

Das ist exakt die Lean-Oberfläche

```
abbrev Omega (p : Policy b) : Type := B (p := p)
abbrev st0 (p : Policy b) : ReducedOpState p := stLap (p := p)
abbrev st1 (p : Policy b) : ReducedOpState p := reduce (p := p) (st0 (p := p))
abbrev L (p : Policy b) : Matrix (Omega (p := p)) (Omega (p := p)) R := (st1 (p := p)).
    LBB
```

Korollar 4.1 (Bedeutung von L)**.** *Der exportierte Operator L ist genau der aus dem ToC-Laplacian per stabilisierter DtN-/Kron-Elimination gewonnene Randoperator.*

Beweis. Das ist die unmittelbare Kombination von `L_def` mit dem soeben bewiesenen Satz `reduce_LBB_eq_dtn_stab`. □

4.2 Symmetrie von L

Theorem 4.2 (Symmetrie des exportierten Randoperators)**.** *Es gilt*

$$L\_transpose\_eq : \qquad L^{\mathsf{T}} = L.$$

Beweis. Formal hängt `L_transpose_eq` an einer endlichen Kette expliziter Hilfssätze. Zuerst liefert `reduce_LBB_eq_dtn_stab` die Identifikation von L mit dem stabilisierten Schur-Komplement. Für den *Beweis von* `reduce_LBB_eq_dtn_stab` benutzt Lean nicht nur `reduce`, `stLap` und `ofLaplacian`, sondern im kritischen `simpa`-Anker auch explizit `TreeOfCliques.DtNStabilized.s`, `TreeOfCliques.DtNStabilized.L`, `PillarA.OQS.Split.DtNStabilized.DtN_stabilized_of`, `L0`, `sSplit` und `k`. Erst *nach* diesem Umschreiben verwendet `L_transpose_eq` das Resultat per `rw[reduce_LBB_eq_dtn_stab]`; danach liegt tatsächlich die explizite Schur-Komplement-Form von L vor,

$$L = L^{(0)}_{BB} - L^{(0)}_{BI}\,(L^{\mathrm{stab}}_{II})^{-1} L^{(0)}_{IB}.$$

Dann werden nacheinander gezeigt:

$$\begin{aligned}
(L^{(0)}_{BB})^{\mathsf{T}} &= L^{(0)}_{BB},\\
(L^{(0)}_{BI})^{\mathsf{T}} &= L^{(0)}_{IB},\\
(L^{(0)}_{IB})^{\mathsf{T}} &= L^{(0)}_{BI},\\
\big((L^{\mathrm{stab}}_{II})^{-1}\big)^{\mathsf{T}} &= (L^{\mathrm{stab}}_{II})^{-1}.
\end{aligned}$$

Die ersten drei Identitäten sind inhaltlich Blockaussagen über eine symmetrische Muttermatrix L_0: Aus $L_0^{\mathsf{T}} = L_0$ folgt für passende Rand- bzw. Innenindizes unmittelbar

$$(L^{(0)}_{BB})_{ij} = (L_0)_{ij} = (L_0)_{ji} = (L^{(0)}_{BB})_{ji},$$

und für die Kreuzblöcke

$$\begin{aligned}
((L_{BI})^{\mathsf{T}})_{ij} &= (L_{BI})_{ji} = (L_0)_{ji} = (L_0)_{ij} = L_{IB,ij},\\
((L_{IB})^{\mathsf{T}})_{ij} &= (L_{IB})_{ji} = (L_0)_{ji} = (L_0)_{ij} = L_{BI,ij}.
\end{aligned}$$

Die Gleichung $(L_{BI})_{ji} = L_{IB,ij}$ ist also kein isolierter Indextrick, sondern genau die Auswertung der Symmetrie $L_0^{\mathsf{T}} = L_0$ auf einen Rand- und einen Innenindex. Auf Codeebene werden die öffentlichen Blocksymmetriesätze durch `blockBB_transpose_of_symm` und `blockII_transpose_of_symm` getragen; die beiden Kreuzblockidentitäten werden im Handoff-Modul durch private Hilfslemmata aus derselben Symmetrie gewonnen und sind hier deshalb bewusst nicht als nachschlagbare öffentliche API-Namen referenziert.

Für den inversen Innenblock wird zunächst die Symmetrie von

$$A := L^{\mathrm{stab}}_{II} = \mathrm{stabilize}(\mathrm{symm}(L^{(0)}_{II}), \delta)$$

festgehalten. Sie folgt daraus, dass $\mathrm{symm}(L_{II}^{(0)})$ per Konstruktion symmetrisch ist und $\mathrm{stabilize}(S, \delta) = S + \delta I$ für symmetrisches S wieder symmetrisch bleibt, weil

$$(S + \delta I)^{\mathsf{T}} = S^{\mathsf{T}} + \delta I^{\mathsf{T}} = S + \delta I.$$

Dann wird aus der Symmetrie von A und der Inverseneigenschaft $A \cdot A^{-1} = I$ die Hilfsgleichung

$$(A^{-1})^{\mathsf{T}} A = I$$

gewonnen; genau dabei benutzt der Lean-Beweis die Komponente `hInv.2`. Danach wird $(A^{-1})^{\mathsf{T}}$ rechts mit $AA^{-1} = I$ erweitert und via Assoziativität sowie $(A^{-1})^{\mathsf{T}} A = I$ wieder auf A^{-1} reduziert. Das ist genau die elementare Rechnung

$$(A^{-1})^{\mathsf{T}} A = I, \qquad AA^{-1} = I \quad \Longrightarrow \quad (A^{-1})^{\mathsf{T}} = A^{-1}.$$

Setzt man dies in die Transposition der Schur-Komplementformel ein, erhält man unmittelbar wieder dieselbe Matrix. □

4.3 A-seitige diagnostische Materiedaten

Der aktive Hand-off der Temperatur läuft über die kanonische Scale-Breaking-Achse in derselben Datei. Zunächst werden definiert:

$$\mathrm{kWindow}(p) := p.K_{\max}, \tag{12}$$
$$\mathrm{kStar}(p) := \text{argmax der breakMeasure auf } \{0, \dots, \mathrm{kWindow}(p)\}, \tag{13}$$
$$\mathrm{signalAt}(p, k) := \big(\mathrm{breakMeasure}(k)\big)^{\mathrm{real}}, \tag{14}$$
$$\mathrm{signalFromA}(p) := \mathrm{signalAt}(p, \mathrm{kStar}(p)), \tag{15}$$
$$\beta_{\mathrm{A}}(p) := 1 + |\mathrm{signalFromA}(p)|. \tag{16}$$

Zusätzlich:

$$d_{\mathrm{A}}(p) := \mathrm{kWindow}(p) + 1, \tag{17}$$
$$\phi_{\mathrm{A}}(p)(i) := \mathrm{signalAt}(p, i). \tag{18}$$

Theorem 4.3 (Positivität der inversen Temperatur)**.** *Es gilt*

$$\texttt{betaFromA\_pos}: \qquad 0 < \beta_{\mathrm{A}}(p).$$

Beweis. Nach Definition ist

$$\beta_{\mathrm{A}}(p) = 1 + |\mathrm{signalFromA}(p)|.$$

Da $|\mathrm{signalFromA}(p)| \geq 0$, gilt $1 \leq \beta_{\mathrm{A}}(p)$, also erst recht $0 < \beta_{\mathrm{A}}(p)$. Genau so ist der Lean-Beweis formuliert. □

5 Re-Exposition in Pillar B: $\Omega, L, T, N_{\mathrm{ToC}}$

Die B-seitige Oberfläche liegt in `PillarB/AQFT/Derived/TreeOfCliquesFromPillarA.lean`.

5.1 Re-Export von Ω und L

In Namespace `TreeOfCliquesFromPillarA.ToC` werden Ω und L schlicht wieder aus der A-Seite übernommen. Zusätzlich wird das Matter-Template

$$T = (d, \beta, \phi)$$

definiert durch

$$T.d = d_{\mathrm{A}}, \qquad T.\beta = \beta_{\mathrm{A}}, \qquad T.\phi = \phi_{\mathrm{A}}.$$

Für `T.d` und `T.\phi` existieren in dieser Datei die expliziten `@[simp]`-Sätze `T_d` und `T_phi`. Für `T.\beta` gibt es in `TreeOfCliquesFromPillarA.lean` dagegen *kein* separates `@[simp]`-Lemma; die Identifikation mit `betaFromA` ist dort nur definitional. Ein explizites Lemma `T_beta_eq_betaFromA` erscheint erst in `TreeOfCliquesExtendedFromPillarA.lean`.

Bemerkung 5.1 (Aktiver Netzträger)**.** Der Träger des aktiven Netzes ist *nicht* das Matter-Template, sondern allein der Randträger Ω. Die Temperatur β wird erst auf der Zustands-/KMS-Seite relevant.

5.2 Das aktive lokale Netz

Definiert wird

$$\texttt{N\_ToC} := \texttt{boundaryMatrixLocalNet}(\Omega).$$

Um das mathematisch zu verstehen, muss man `BoundaryMatrixLocalNet.lean` lesen.

Definition 5.2 (Unterstützung in einer Region)**.** Für eine Randregion $r \subseteq \Omega$ heißt eine Matrix $U \in \mathrm{Mat}(\Omega)$ in $r \times r$ getragen, wenn

$$\forall i, j \in \Omega : \quad i \notin r \text{ oder } j \notin r \implies U_{ij} = 0.$$

Dies ist `SupportedOnrU`.

Definition 5.3 (Lokalisierte Matrix)**.** Eine Matrix $A \in \mathrm{Mat}(\Omega)$ ist in r lokalisiert, wenn es ein $\lambda \in \mathbb{C}$ gibt mit

$$A - \lambda \mathbf{1} \quad \text{getragen in } r \times r.$$

Dies ist `RegionLocalizedrA`.

Definition 5.4 (Lokale Algebra eines Gebietes)**.** Die lokale Faser über r ist der Subtyp

$$\mathrm{RegionBlockMat}(\Omega, r) := \{A \in \mathrm{Mat}(\Omega) \mid \mathrm{RegionLocalized}(r, A)\}.$$

Sie wird mit den offensichtlichen Operationen zu einer $*$-Algebra gemacht.

Proposition 5.5 (Lokales Netz der reiche Pfads)**.** *Das Objekt* `boundaryMatrixLocalNet` *ist ein* echtes *lokales $*$-Algebranetz auf dem Randträger Ω, dessen Faser über r aus den gerade beschriebenen lokalisierten Matrizen besteht.*

Beweis. `boundaryMatrixLocalNet` bündelt die Struktur `boundaryMatrixLocalAlgebraNet` zusammen mit der für jede Region bereits konstruierten $*$-Algebra `regionBlockStarAlgebra`. Die Isotonieabbildungen sind kanonische Inklusionen von Subtypen: Vergrößert man die Region, dann bleibt jede zuvor lokalisierte Matrix lokalisiert. Genau dies ist der Satz `regionLocalized_mono`. Alle Homomorphieeigenschaften werden dann in `incl_isHom` komponentenweise per `rfl` verifiziert. □

6 Der Zustandszweig

Im Folgenden wird $0 < b$ vorausgesetzt, genau wie im Codeabschnitt `sectionStrictMatrixState`. Diese Annahme garantiert einen nichtleeren exportierten Randträger.

6.1 Der starke Gibbs-Zustand auf der Vollmatrixalgebra

Der starke Topzustand ist

$$\rho_{\mathrm{top}}^{\mathrm{strong}} := \texttt{boundaryDensityStateAtBeta}(L, T.\beta, L^{\mathsf{T}} = L).$$

Die Funktionalform ist

$$\rho_{\mathrm{top}}^{\mathrm{strong}}(A) = \mathrm{Tr}(\rho_\beta\, A),$$

mit der Gibbs-Dichtematrix

$$\rho_\beta = Z_\beta(L)^{-1} e^{-\beta H(L)}, \qquad H(L) := \texttt{fullHamiltonian}(L), \qquad H(L)_{ij} = (L_{ij} : \mathbb{C}).$$

Äquivalent dazu kann man schreiben

$$\rho_\beta = Z_\beta(L)^{-1} e^{-H(\beta L)},$$

weil im Code $\texttt{boundaryDensityStateAtBeta}(L, \beta, h_L)$ per Definition gerade $\texttt{boundaryDensityState}((\beta : \mathbb{R}) \cdot L)$ ist. Die reine Hamilton-Identifikation

$$H(\beta L) = \beta\, H(L)$$

steht formal in `fullHamiltonian_smul_real`. Für die Exponential- und Gibbs-Schreibweise werden dann aber tatsächlich die stärkeren Sätze `hamExp_smul_real` und `gibbsOpCoreAtBeta_eq_gibbsOpCore_smul` benutzt; auf Dichtematrixniveau kulminiert das in `densityMatrixCoreAtBeta_eq_densityMatrixCore_smul`. Es gibt also *keinen* von L unabhängigen separaten Hamiltonoperator H.

Bemerkung 6.1 (ω_β und $\rho_{\mathrm{top}}^{\mathrm{strong}}$ sind auf Vollmatrixniveau dasselbe Funktional)**.** Im späteren KMS-Teil wird dieselbe Vollmatrix-Zustandsfunktionalform mit

$$\omega_\beta(A) := \mathrm{Tr}(\rho_\beta A)$$

notiert. Auf der Vollmatrixalgebra gilt also schlicht

$$\omega_\beta = \rho_{\mathrm{top}}^{\mathrm{strong}}.$$

Die unterschiedliche Notation dient nur dazu, im Zustandszweig den aus Pillar A kommenden starken Topzustand und im KMS-Zweig denselben Zustand in seiner thermodynamischen Rolle zu markieren.

Proposition 6.2 (Explizite Formel des starken Zustands)**.** *Für jede Matrix* $A \in \mathrm{Mat}(\Omega)$ *gilt*

$$\rho_{\mathrm{top}}^{\mathrm{strong}}(A) = \texttt{boundaryDensityStateMatAtBeta}(L, \beta, A).$$

Beweis. Das ist genau ρ`TopStrong_apply`, das sich wiederum auf `boundaryDensityStateAtBeta_apply` zurückführt. Dort wird `boundaryDensityStateAtBeta` als der bereits bekannte Gibbs-Zustand der skalierten Matrix $(\beta \cdot L)$ identifiziert. Formal ist

$$\texttt{boundaryDensityStateAtBeta}(L, \beta, h_L) = \texttt{boundaryDensityState}((\beta : \mathbb{R}) \cdot L).$$

Die konkrete Dichtematrixgleichung liefert `densityMatrixCoreAtBeta_eq_densityMatrixCore_smul`. Deren Beweis benutzt die Kette

`fullHamiltonian_smul_real` → `hamExp_smul_real`
→ `gibbsOpCoreAtBeta_eq_gibbsOpCore_smul`
→ `densityMatrixCoreAtBeta_eq_densityMatrixCore_smul`.

so dass auf Dichtematrixniveau genau die oben notierte Form

$$Z_\beta(L)^{-1} e^{-\beta H(L)} = Z_\beta(L)^{-1} e^{-H(\beta L)}$$

folgt. □

6.2 Topzustand und regionale Zustände

Der leichte Topzustand ρ_{top} und die regionalen Zustände entstehen in zwei benannten, aber eng verbundenen Schritten:

(a) zuerst als *starke* Restriktionen `boundaryMatrixRegionStrong`,

(b) dann als leichte API-Oberfläche `boundaryMatrixRegion`, die im Code lediglich ein `abbrev` für `boundaryMatrixRegionStrong` ist.

Damit gibt es zwei verschiedene Namen, aber keinen zweiten physikalischen Inhalt.

$$\rho_r^{\mathrm{strong}}(A) = \rho_{\mathrm{top}}^{\mathrm{strong}}(\iota_r(A)), \tag{19}$$

$$\rho_r := \rho_r^{\mathrm{strong}}, \tag{20}$$

$$\rho_{\mathrm{top}} := \rho_{\mathrm{top}}^{\mathrm{strong}} \restriction N_{\mathrm{ToC}}(\mathrm{top}). \tag{21}$$

Theorem 6.3 (Explizite regionale Formel, starke und leichte Fassung)**.** *Für jede Region r und jedes lokale Element $A \in N_{\mathrm{ToC}}(r)$ gilt*

$$\rho_r^{\mathrm{strong}}(A) = \mathrm{Tr}\big(\rho_\beta \,\mathrm{RegionBlockMat.toMatrix}(A)\big).$$

formal `boundaryMatrixRegionStrong_apply`*. Da* `boundaryMatrixRegion` *nur eine Abkürzung dafür ist, gilt ebenso*

$$\rho_r(A) = \mathrm{Tr}\big(\rho_\beta \,\mathrm{RegionBlockMat.toMatrix}(A)\big).$$

formal ρ`Region_apply`*. Der Spezialfall $r = \mathrm{top}$ ist* ρ`Top_apply`*.*

Beweis. Die Definition `boundaryMatrixRegionStrong` ist genau die Restriktion des starken Zustands entlang `boundaryMatrixRegionToMatrixHom`. Setzt man die bereits bewiesene Matrixformel des starken Zustands ein, erhält man

$$\rho_r^{\mathrm{strong}}(A) = \texttt{boundaryDensityStateMatAtBeta}(L, \beta, \mathrm{RegionBlockMat.toMatrix}(A)).$$

Genau das ist `boundaryMatrixRegionStrong_apply`. Die zweite Formel ist dann nur noch die definitorische Entfaltung von `boundaryMatrixRegion` zu `boundaryMatrixRegionStrong`; dafür existiert zusätzlich der explizite Satz `boundaryMatrixRegion_eq_regionStrong`. Also folgt unmittelbar auch ρ`Region_apply`. Für $r = \mathrm{top}$ ergibt sich dieselbe Aussage mit ρ`Top_apply`. □

Theorem 6.4 (Kompatibilität der lokalen Zustände)**.** *Sind $a \le b$ Regionen, so gilt zunächst für die starke Familie*

$$\rho_a^{\mathrm{strong}} = \rho_b^{\mathrm{strong}}|_{N_{\mathrm{ToC}}(a)},$$

formal `boundaryMatrixRegionStrong_compat`*. Für die leichte Oberfläche folgt daraus wegen* `boundaryMatrixRegion=boundaryMatrixRegionStrong` *definitorisch dieselbe kompatible Familie; global wird sie in* `boundaryMatrixStateConeFromStrongTop` *als Kegel mit* `\omega :=boundaryMatrixRegion` *paketiert. Weiter gilt für jede Region r*

$$\rho_r = \rho_{\mathrm{top}}|_{N_{\mathrm{ToC}}(r)},$$

formal `boundaryMatrixRegion_eq_restrict_top`*. Für $r = \mathrm{top}$ erhält man*

$$\rho_{\mathrm{top}} = \rho_{\mathrm{top}},$$

formal `boundaryMatrixRegion_top`*.*

Beweis. Die Isotonieeinbettungen des Netzes sind echte Untertyp-Inklusionen. Für $a \leq b$ und $A \in N_{\mathrm{ToC}}(a)$ sagt `boundaryMatrixRegionToMatrix_natural` exakt

$$\mathrm{RegionBlockMat.toMatrix}(\iota_{a\leq b}(A)) = \mathrm{RegionBlockMat.toMatrix}(A).$$

Mit anderen Worten: Dasselbe lokale Observable wird in der großen Vollmatrixalgebra durch genau dieselbe Matrix dargestellt, unabhängig davon, ob man es als Element von $N_{\mathrm{ToC}}(a)$ oder nach Inklusion als Element von $N_{\mathrm{ToC}}(b)$ auffasst. Da ρ_b^{strong} durch Auswertung des starken Vollmatrixzustands auf eben dieser Matrix definiert ist, müssen beide Auswertungswege denselben Zahlenwert liefern. Genau das ist `boundaryMatrixRegionStrong_compat`. Für `boundaryMatrixRegion_eq_restrict_top` wird der Topzustand zunächst definitional als starker Topzustand umgeschrieben; danach ist die Aussage genau dieselbe Kompatibilitätsaussage für die spezielle Einbettung `le_top'`. Für $r = \mathrm{top}$ fällt die Restriktion auf die Identität zurück; das ist formal `boundaryMatrixRegion_top`, also definitional `rfl`. □

6.3 Globaler Zustandskegel und quasilokaler Zustand

Aus der kompatiblen Familie $r \mapsto \rho_r$ wird der globale Zustandskegel konstruiert. Formal ist ein `GlobalStateCone` auf einem Zustandsnetz SN die Struktur

$$C = (\omega, \mathrm{compat}),$$

wobei ω jeder Region r einen lokalen Zustand $C.\omega(r) \in SN.\mathrm{State}(r)$ zuordnet und compat für alle Inklusionen $a \leq b$ die Restriktionsgleichung

$$SN.\mathrm{restrict}(a \leq b)(C.\omega(b)) = C.\omega(a)$$

fordert. Die konkrete ToC-Instanz ist

$$\texttt{cone\_FromA} = \texttt{boundaryMatrixStateConeFromStrongTop},$$

also genau die Familie $r \mapsto \rho_r$ zusammen mit der in Satz 6.3 bewiesenen Kompatibilität. Daraus wird durch den allgemeinen Lift-Mechanismus der quasilokale Zustand

$$\Phi_{\mathrm{FromA}} := \texttt{boundaryMatrixQuasiLocalStateFromStrongTop}.$$

Theorem 6.5 (Rückrestriktion des quasilokalen Zustands)**.** *Für jede Region r gilt*

$$\Phi_{\mathrm{FromA}} \restriction_r = \rho_r.$$

Auf der Topregion also insbesondere

$$\Phi_{\mathrm{FromA}} \restriction_{\mathrm{top}} = \rho_{\mathrm{top}}.$$

Beweis. Der Lift wird mit `QuasiLocal.stateLift` aus einem `GlobalStateCone` erzeugt. Seine Universaleigenschaft lautet allgemein

$$\mathrm{restrictToRegion}(\mathrm{stateLift}(C), r) = C.\omega(r).$$

Genau das ist `gate_restrictToRegion_stateLift`. Im vorliegenden Fall ist C der aus den lokalen Zuständen $r \mapsto \rho_r$ gebildete Kegel `boundaryMatrixStateConeFromStrongTop`; also ist $C.\omega(r) = \rho_r$. Setzt man dies in die allgemeine Lift-Gleichung ein, erhält man unmittelbar

$$\Phi_{\mathrm{FromA}} \restriction_r = \rho_r.$$

`restrictToRegion_univ_Φ_FromA` in `TreeOfCliquesFromPillarA.lean` ist genau die Spezialisierung auf die Topregion. □

7 Der Abschlusszweig

7.1 Die Topfaser ist die volle Matrixalgebra

Die Topregion ist $r = \Omega$. Dann ist jede Matrix automatisch in r lokalisiert, denn außerhalb von Ω gibt es keine Indizes. Entsprechend konstruiert `BoundaryMatrixLocalNet.lean` die Isomorphie

$$\texttt{boundaryMatrixTopIsoMatrix} : N_{\mathrm{ToC}}(\mathrm{top}) \xrightarrow{\cong} \mathrm{Mat}(\Omega).$$

Proposition 7.1 (Topfaser-Vollmatrix-Isomorphie)**.** *Die Abbildung*

$$A \longmapsto \mathrm{RegionBlockMat.toMatrix}(A)$$

ist ein $*$-Isomorphismus der Topfaser auf die volle Matrixalgebra, mit Inverser

$$M \longmapsto \mathrm{matrixToTopRegion}(M).$$

Beweis. Die Homomorphismen heißen `boundaryMatrixTopToMatrixHom` und `matrixToBoundaryMatrixTopHom`. Auf der Topregion ist jede Matrix lokalisiert; entsprechend ist

$$\mathrm{matrixToTopRegion}(M) := (M, \mathrm{regionLocalized\_univ}(M)),$$

also dieselbe Matrix zusammen mit dem trivialen Lokalisierungszertifikat auf der Topregion. In `boundaryMatrixTopIsoMatrix` wird die linke Inverse *auf der Topfaser* mit `RegionBlockMat.ext` gezeigt, während die rechte Inverse *auf der Vollmatrixseite* schlicht `rfl` ist. Also liegt ein echter $*$-Isomorphismus vor. □

7.2 Die quasilokale Algebra ist die Topfaser und damit die Vollmatrixalgebra

`BoundaryMatrixQuasiLocalClosure.lean` baut zunächst

$$\texttt{boundaryMatrixQuasiLocalIsoTop} : \mathcal{A}_\infty(N_{\mathrm{ToC}}) \cong N_{\mathrm{ToC}}(\mathrm{top}),$$

und anschließend durch Komposition mit der Top-Isomorphie

$$\texttt{boundaryMatrixQuasiLocalIsoMatrix} : \mathcal{A}_\infty(N_{\mathrm{ToC}}) \cong \mathrm{Mat}(\Omega).$$

Theorem 7.2 (Quasilokaler Abschluss des aktiven rich path)**.** *Die quasilokale Abschlussalgebra des aktiven ToC-Netzes ist kanonisch $*$-isomorph zur vollen Randmatrixalgebra:*

$$\mathcal{A}_\infty(N_{\mathrm{ToC}}) \cong \mathrm{Mat}(\Omega).$$

Der an der ToC-Oberfläche benutzte Name ist

$$\texttt{quasiLocalClosureIsoMatrix\_FromA}.$$

Beweis. `boundaryMatrixQuasiLocalIsoTop` benutzt die allgemeinen Top-Operationen `QuasiLocal.toTop` und `QuasiLocal.ofTop`; die Inversenbeweise sind genau die Sätze `QuasiLocal.ofTop_toTop` und `QuasiLocal.toTop_ofTop`. Danach wird in `boundaryMatrixQuasiLocalIsoMatrix` mit `boundaryMatrixTopIsoMatrix` komponiert. Die linke und rechte Inverse werden durch zweimaliges Einsetzen der jeweiligen Inverseneigenschaften gezeigt. Mehr passiert mathematisch nicht: Die quasilokale Algebra kollabiert hier wegen der Endlichkeit des Trägers auf die Topfaser, und diese ist bereits die Vollmatrixalgebra. □

Bemerkung 7.3 (Wichtige physikalische Konsequenz)**.** Dies ist eine *endliche Typ-I-Situation*. Sie ist algebraisch reichhaltig und für GNS/KMS/Modularität völlig ausreichend, aber sie ist noch nicht die Situation einer continuum-AQFT mit Typ-III-Lokalalgebren.

8 Der GNS-Zweig

8.1 Sesquilinearform und Darstellung

Die GNS-Sesquilinearform lautet

$$\langle X, Y\rangle_\rho := \rho_\beta(X^\dagger Y),$$

formal `boundaryMatrixGNSSesq`. Die Darstellung der Topfaser auf dem GNS-Raum $\mathrm{Mat}(\Omega)$ ist Linksmultiplikation:

$$\pi(A)X = AX.$$

Formal heißt dies `boundaryMatrixTopGNSPi`.

Proposition 8.1 (Konkrete Topdarstellung)**.** *Für jedes Top-Element A und jede Matrix X gilt*

$$\pi(A)X = \mathrm{RegionBlockMat.toMatrix}(A)\, X.$$

Beweis. Das ist unmittelbar `boundaryMatrixTopGNSPi_apply`. Die Darstellung ist nichts anderes als die aus `boundaryMatrixMatrixGNSPi` geerbte Linksmultiplikation mit dem Matrixbild des Top-Elements. □

8.2 Zyklischer Vektor

Der zyklische Vektor ist die Einheitsmatrix

$$\Omega_{\mathrm{cyc}} := \mathbf{1},$$

formal `boundaryMatrixGNSCyclicVec`.

Theorem 8.2 (Zyklizität)**.** *Jede Matrix $X \in \mathrm{Mat}(\Omega)$ wird durch Wirkung eines Top-Elements auf den zyklischen Vektor erzeugt:*

$$\exists A \in N_{\mathrm{ToC}}(\mathrm{top}) : \quad \pi(A)\Omega_{\mathrm{cyc}} = X.$$

Beweis. Wähle $A := \mathrm{matrixToTopRegion}(X)$. Dann ist $\mathrm{RegionBlockMat.toMatrix}(A) = X$. Also

$$\pi(A)\Omega_{\mathrm{cyc}} = X \cdot \mathbf{1} = X.$$

Dies ist exakt der Lean-Beweis von `boundaryMatrixTopGNS_cyclic`. □

Theorem 8.3 (Rekonstruktion des Topzustands aus dem zyklischen Vektor)**.** *Für jedes Top-Element A gilt*

$$\rho_{\mathrm{top}}(A) = \langle \Omega_{\mathrm{cyc}}, \pi(A)\Omega_{\mathrm{cyc}}\rangle_\rho.$$

Beweis. Setzt man $\Omega_{\mathrm{cyc}} = \mathbf{1}$ und $\pi(A)\Omega_{\mathrm{cyc}} = A\mathbf{1} = A$ in die Sesquilinearform ein, erhält man

$$\langle \mathbf{1}, A\rangle_\rho = \rho_\beta(\mathbf{1}^\dagger A) = \rho_\beta(A) = \rho_{\mathrm{top}}(A).$$

Formal ist das der Satz `boundaryMatrixTop_state_from_cyclic`. □

8.3 Das konkrete GNS-Datum

Daraus wird das Bündel

`boundaryMatrixTopGNSDatum`

erzeugt, auf der ToC-Oberfläche unter dem Namen

`gnsDatum_top_ToC`.

Proposition 8.4 (GNS-Datum des strikten ToC-Pfads)**.** *`gnsDatum_top_ToC` ist das GNS-Datum des Topzustands ρ_{top} mit*

$$H = \mathrm{Mat}(\Omega), \qquad \pi(A)X = AX, \qquad \Omega_{\mathrm{cyc}} = \mathbf{1}.$$

Beweis. Die Struktur `boundaryMatrixTopGNSDatum` setzt genau diese Komponenten ein. Zusätzlich enthält der zugrunde liegende Datentyp `GNSDatum` noch die Instanzfelder `instNormedAddCommGroup` und `instNormedSpace`; in der konkreten Matrixrealisierung werden auch diese auf dem Träger $H = \mathrm{Mat}(\Omega)$ explizit besetzt. Die Repräsentationseigenschaften $\pi(A+B) = \pi(A) + \pi(B)$, $\pi(AB) = \pi(A)\pi(B)$, $\pi(1) = 1$ werden mit den Sätzen `boundaryMatrixTopGNSPi_add`, `boundaryMatrixTopGNSPi_mul`, `boundaryMatrixTopGNSPi_one` nachgewiesen; Zyklizität und Zustandsrekonstruktion sind die beiden vorigen Sätze. □

9 Der Dynamikzweig

9.1 Volle Heisenberg-Dynamik und Zeitorientierung

Formell ist eine Dynamik im Repository eine Struktur `StarAlgDynamics` mit einer Familie α_t von $*$-Endomorphismen, so dass

$$\alpha_0 = \mathrm{id}, \qquad \alpha_{s+t} = \alpha_s \circ \alpha_t.$$

Die Kompositionsreihenfolge bedeutet also: zuerst Zeit t, dann Zeit s.

`BoundaryMatrixFullDynamics.lean` trennt explizit zwei Zeitrichtungen:

$$\alpha_t^{+}(A) = e^{+itH} A e^{-itH}, \tag{22}$$

$$\alpha_t^{-}(A) = e^{-itH} A e^{+itH}. \tag{23}$$

Der KMS-Pfad benutzt *nur* die Minus-Richtung. Formal:

$$\texttt{fullBoundaryDynMinus}(L, h_L).$$

Bemerkung 9.1 (Vorzeichenkonvention)**.** Der Repository-Standard ab KMS-6a/KMS-6b ist explizit:

`Plus`	positive Heisenberg-Zeitentwicklung,
`Minus`	negative/inverse Heisenberg-Zeitentwicklung.

Der Gibbs-Zustand ist an die Minus-Dynamik gekoppelt.

9.2 Transport auf die Topfaser

Die Topdynamik wird nicht neu erfunden, sondern per $*$-Isomorphie transportiert. Setzt man

$$\iota_{\mathrm{top}} := \texttt{boundaryMatrixTopIsoMatrix}, \qquad \mathcal{D}_L^{-} := \texttt{fullBoundaryDynMinus}(L, h_L),$$

so lautet die Transportdefinition

$$\texttt{boundaryMatrixTopDynMinus} = \mathrm{transportDynamics}(\iota_{\mathrm{top}}, \mathcal{D}_L^{-}).$$

Auf der ToC-Oberfläche heißt dieses Objekt

$$\texttt{dyn\_top\_ToC}.$$

Theorem 9.2 (Explizite Topdynamik)**.** *Für jedes Top-Element A und jede Zeit t gilt präzise*

$$\alpha_t^{\mathrm{top}}(A) = \mathrm{matrixToTopRegion}\Big(\mathcal{D}_L^-.\alpha_t\big(\mathrm{RegionBlockMat.toMatrix}(A)\big)\Big).$$

Schreibt man dafür abkürzend

$$\alpha_t^-(M) := \mathcal{D}_L^-.\alpha_t(M),$$

so lautet dieselbe Formel

$$\alpha_t^{\mathrm{top}}(A) = \mathrm{matrixToTopRegion}\big(\alpha_t^-(\mathrm{RegionBlockMat.toMatrix}(A))\big).$$

Beweis. Dies ist exakt das Auswertungslemma `boundaryMatrixTopDynMinus_alpha_apply`. Der Grund ist definitorisch: `boundaryMatrixTopDynMinus` ist die entlang von `boundaryMatrixTopIsoMatrix` transportierte Vollmatrixdynamik. Daher wird ein Top-Element A zunächst mit `RegionBlockMat.toMatrix` als Vollmatrix gelesen, dann die Minus-Dynamik auf dieser Vollmatrix angewandt, und das Ergebnis anschließend mit `matrixToTopRegion` wieder in die Topfaser zurückgeführt. Genau das ist die oben notierte Formel. □

10 Der KMS-Zweig

10.1 Gibbs-Zustand auf Vollmatrixniveau

Die Datei `BoundaryMatrixFullKMSFromL.lean` definiert für allgemeines $\beta > 0$:

$$\mathrm{gibbsOpCoreAtBeta}(L,\beta) := e^{-\beta H(L)}, \tag{24}$$

$$Z_\beta(L) := \Re\,\mathrm{Tr}(e^{-\beta H(L)}), \tag{25}$$

$$\rho_\beta := Z_\beta(L)^{-1} e^{-\beta H(L)}, \tag{26}$$

$$\omega_\beta(A) := \mathrm{Tr}(\rho_\beta A). \tag{27}$$

Formal ist der starke Zustand

$$\texttt{boundaryDensityStateAtBeta}(L,\beta,h_L).$$

Im Folgenden bleibt die leichtere Schreibweise ρ_β stehen, aber die Abhängigkeit von L ist stets über die Normierung $Z_\beta(L)$ und den Hamiltonoperator $H(L)$ mitzuführen.

Ein leichter KMS-Zustand ist im Repository die Struktur

$$\texttt{KMSState} = (\beta, \mathrm{dyn}, \mathrm{state}, \mathrm{isKMS}),$$

wobei `isKMS` aus Positivität von β, Dynamikinvarianz und der Streifenbedingung besteht. Auf C*-Niveau wird dies durch

$$\texttt{CStarKMSStateOn} = (\beta, \mathrm{dyn}, \mathrm{state}, \mathrm{isKMS})$$

verstärkt, wobei `state` nun ein echter `CStarStateOn` ist und die KMS-Bedingung über dessen `toStateOn`-Ansicht formuliert wird.

10.2 Die formale KMS-Strip-Bedingung

Der KMS-Begriff wird im Repository als explizite Streifenbedingung kodiert. Die Struktur `KMSStrip` für (D,β,φ) verlangt genau: Zu jedem Observablenpaar a,b existiert eine Funktion $F:\mathbb{C}\to\mathbb{C}$ mit

(i) Holomorphie auf dem offenen Streifen $0 < \mathrm{Im}\, z < \beta$,

(ii) Stetigkeit auf dem abgeschlossenen Streifen $0 \le \mathrm{Im}\, z \le \beta$,

(iii) Randwertformel $F(t) = \varphi(\alpha_t(a)b)$ für reelles t,

(iv) Randwertformel $F(t+i\beta) = \varphi(b\,\alpha_t(a))$ für reelles t.

Die nachfolgende konkrete Matrixkonstruktion ist genau ein solcher Zeuge.

10.3 Der explizite Matrix-KMS-Kern

Der KMS-Beweis wird nicht nur abstrakt behauptet, sondern über einen expliziten Kern geführt:

$$F_\beta(z) := \mathrm{Tr}\Big(\rho_\beta\, e^{-izH(L)} A e^{izH(L)} B\Big).$$

Dabei ist $H(L) = \texttt{fullHamiltonian}(L)$; äquivalent arbeitet der Code mit der zu $(\beta \cdot L)$ gehörigen Gibbs-Matrix `densityMatrixCoreAtBeta`. Das ist `fullKMSKernelMinusAtBeta`.

Die analytische Last wird im Code nicht verschwiegen: `fullKMSKernelMinusAtBeta_differentiable` zeigt, dass $z \mapsto F_\beta(z)$ auf ganz $\mathbb{C}$ differenzierbar ist. Inhaltlich ist das die Standardtatsache, dass $z \mapsto -izH(L)$ linear, die Matrixexponentialfunktion analytisch und sowohl Matrizenmultiplikation als auch Spur holomorphieerhaltend sind. Damit ist F_β insbesondere auf dem offenen Streifen $0 < \mathrm{Im}\, z < \beta$ holomorph und auf dem abgeschlossenen Streifen stetig.

Proposition 10.1 (KMS-Randwerte)**.** *Für reelles t gelten die beiden Randwertidentitäten*

$$F_\beta(t) = \omega_\beta(\alpha_t^-(A)B), \tag{28}$$
$$F_\beta(t + i\beta) = \omega_\beta(B\alpha_t^-(A)). \tag{29}$$

Formal sind dies `fullKMSKernelMinusAtBeta_re_boundary` und `fullKMSKernelMinusAtBeta_top_boundary`.

Beweis. Die erste Identität ist direkte Auswertung von F_β auf der reellen Achse:

$$F_\beta(t) = \mathrm{Tr}(\rho_\beta e^{-itH(L)} A e^{itH(L)} B) = \omega_\beta(\alpha_t^-(A)B).$$

Für die obere Randwertgleichung setzt man $z = t + i\beta$. Dann zerfallen die Exponentialfaktoren zu

$$e^{-i(t+i\beta)H(L)} = e^{\beta H(L)} e^{-itH(L)}, \qquad e^{i(t+i\beta)H(L)} = e^{itH(L)} e^{-\beta H(L)},$$

formal `unitaryFromLMinusC_top_atBeta` und `unitaryFromLPlusC_top_atBeta`. Damit wird

$$\begin{aligned}
F_\beta(t + i\beta) &= \mathrm{Tr}\Big(\rho_\beta e^{\beta H(L)} e^{-itH(L)} A e^{itH(L)} e^{-\beta H(L)} B\Big) \\
&= \mathrm{Tr}\Big(Z_\beta(L)^{-1} e^{-itH(L)} A e^{itH(L)} e^{-\beta H(L)} B\Big) && \text{da } \rho_\beta e^{\beta H(L)} = Z_\beta(L)^{-1}\mathbf{1} \\
&= \mathrm{Tr}\Big(e^{-\beta H(L)} B\, e^{-itH(L)} A e^{itH(L)}\Big) && \text{Zyklizität der Spur} \\
&= \mathrm{Tr}\Big(\rho_\beta B\, e^{-itH(L)} A e^{itH(L)}\Big) && \text{Rückaggregation zu } \rho_\beta \\
&= \omega_\beta(B\alpha_t^-(A)).
\end{aligned}$$

Genau diese beiden Randwertrechnungen werden im Code in `fullKMSKernelMinusAtBeta_re_boundary` und `fullKMSKernelMinusAtBeta_top_boundary` formalisiert. □

Theorem 10.2 (Starker KMS-Zustand auf der Vollmatrixalgebra)**.** *Für $\beta > 0$ und symmetrisches L ist*

$$\texttt{boundaryFullKMSMinusAtBeta}(L, \beta, h_\beta, h_L)$$

ein starker C^*-KMS-Zustand auf $\mathrm{Mat}(\Omega)$ *bezüglich der Minus-Dynamik.*

Beweis. Der zugrundeliegende Satz ist `boundaryDensityState_isKMS_minus_atBeta`. Er bündelt drei Dinge:

(a) $\beta > 0$,

(b) Invarianz des Gibbs-Zustands unter der Minus-Dynamik,

(c) den soeben beschriebenen KMS-Streifenbeweis mit dem analytischen Kern F_β.

Die Invarianz ist dabei die elementare Tatsache, dass die Gibbs-Dichtematrix $\rho_\beta = Z_\beta(L)^{-1}e^{-\beta H(L)}$ mit den Heisenberg-Unitaries $U_t = e^{-itH(L)}$ kommutiert, weil beide nur Funktionen desselben Operators $H(L)$ sind. Außerdem ist $H(L)$ hermitesch, da $L^{\mathsf{T}} = L$ und $H(L)$ nur die komplexifizierte Matrix von L ist; deshalb ist U_t unitär und es gilt $U_t^* U_t = \mathbf{1}$. Mit diesen beiden Tatsachen sowie der Zyklizität der Spur folgt

$$\begin{aligned} \mathrm{Tr}(\rho_\beta\, U_t A U_t^*) &= \mathrm{Tr}(U_t^* \rho_\beta U_t A) && \text{Zyklizität der Spur} \\ &= \mathrm{Tr}(\rho_\beta U_t^* U_t A) && \text{da } \rho_\beta U_t = U_t \rho_\beta \\ &= \mathrm{Tr}(\rho_\beta A). && \text{da } U_t^* U_t = \mathbf{1} \end{aligned}$$

Formal wird dies in `boundaryDensityStateAtBeta_invariant_fullDynMinus` über die Kommutativitätslemmae für `densityMatrixCoreAtBeta` und `unitaryFromLMinus` umgesetzt. Dass diese Kommutativität gilt, wird im Code letztlich aus der Exponentialrechnung derselben Hamiltonmatrix gewonnen; insbesondere werden die Produkte der Hamiltonexponentiale über `hamExp_add` zusammengezogen. `boundaryFullKMSMinusAtBeta` verpackt dieses Resultat dann nur noch in die Struktur `CStarKMSStateOn`; die mathematische Last liegt vollständig in `boundaryDensityState_isKMS_minus_atBeta`. □

10.4 Transport des KMS-Zustands auf die Topfaser

Über `ModularTransport.transportIsKMS` wird der starke Vollmatrix-KMS-Zustand entlang `boundaryMatrixTopIsoMatrix` auf die Topfaser transportiert. Das Ergebnis ist

`boundaryMatrixTopKMSMinusAtBeta`,

auf der ToC-Oberfläche

`kmsDatum_top_ToC`.

Theorem 10.3 (Top-KMS-Datum)**.** *Das Objekt `kmsDatum_top_ToC` ist ein KMS-Zustand auf der Topfaser mit*

$$\beta = T.\beta, \qquad \mathrm{dyn} = \texttt{dyn\_top\_ToC}, \qquad \mathrm{state} = \rho_{\mathrm{top}}.$$

Beweis. Die Struktur `boundaryMatrixTopKMSMinusAtBeta` hat drei nichttriviale Zutaten: (i) Transport der Dynamik via `ModularTransport.transportDynamics`, (ii) Transport der KMS-Eigenschaft via `ModularTransport.transportIsKMS`, (iii) Identifikation des transportierten Zustands mit `boundaryMatrixTop` via `boundaryMatrixTop_eq_transportState`. Inhaltlich passiert dabei Folgendes: Der analytische KMS-Zeuge F auf der Vollmatrixalgebra wird *dieselbe* holomorphe Funktion auf dem Streifen gelassen, aber nun auf die entlang des Isomorphismus transportierten Observablen angewandt. Genau das leistet `transportKMSStrip`. Die Randwerte bleiben korrekt, weil nach Anwendung von `e.hom` und anschließendem Rücktransport mit `e.inv` über `e.right_inv` wieder das ursprüngliche Observable erscheint. Erst danach folgen `boundaryMatrixTopKMSMinusAtBeta_beta`, `..._dyn` und `..._state` als definitorische Gleichheiten. □

11 Der Modular-Join

11.1 Struktur des Modular-Datums

In `BoundaryMatrixModular.lean` ist das Endbündel

`BoundaryMatrixModularDatum`(L, β, h_L)

definiert als Struktur mit Komponenten

matrixKMS, topKMS, gns,
matrixKMS_beta_eq, topKMS_beta_eq,
topKMS_dyn_eq, topKMS_state_eq, state_from_cyclic.

Es ist also *kein* neues zusätzliches Objekt, sondern die Zusammenfassung bereits konstruierter Zweige samt ihrer Kohärenzgleichungen.

Definition 11.1 (Konkretes Modular-Datum)**.** Das konkrete Bündel heißt

`boundaryMatrixTopModularDatum`,

und auf der ToC-Oberfläche

`modularDatum_top_ToC`.

Es wird gefüllt durch

$$\begin{aligned} \text{matrixKMS} &= \texttt{boundaryFullKMSMinusAtBeta}, \\ \text{topKMS} &= \texttt{boundaryMatrixTopKMSMinusAtBeta}, \\ \text{gns} &= \texttt{boundaryMatrixTopGNSDatum}. \end{aligned}$$

Theorem 11.2 (Expliziter Endpunkt des strikten Pfads)**.** *Für den strikten ToC-Pfad existiert ein kanonisch konstruiertes Objekt*

$$\mathit{modularDatum\_top\_ToC},$$

welches gleichzeitig enthält:

(1) den starken Vollmatrix-KMS-Zustand,

(2) den auf die Topfaser transportierten KMS-Zustand,

(3) das konkrete GNS-Datum,

(4) die Identifikation von Temperatur, Zustand und Dynamik,

(5) die Rekonstruktion des Topzustands aus dem zyklischen Vektor.

Beweis. `boundaryMatrixTopModularDatum` setzt die drei bereits konstruierten Komponenten direkt in die Struktur `BoundaryMatrixModularDatum` ein. Die Identitäten `matrixKMS_beta_eq`, `topKMS_beta_eq`, `topKMS_dyn_eq` und `topKMS_state_eq` sind sämtlich definitorisch. Die letzte Komponente `state_from_cyclic` ist exakt der zuvor bewiesene GNS-Satz `boundaryMatrixTop_state_from_cyclic`. Auf der ToC-Oberfläche ist `modularDatum_top_ToC` nur noch die Spezialisierung dieser Struktur auf den aus Pillar A exportierten L- und β-Wert. □

12 Die vollständige Ableitungskette als präziser Graph

Die vollständige logische Struktur des strikten Pfads ist:

$$\begin{array}{l} \text{Primitive Eingabe: } \texttt{Policy}(b).L_{\max} \\ \Downarrow \\ k := L_{\max}, \quad B_k, I_k, V_k, \quad L_0 = \text{ToC-Laplacian} \\ \Downarrow \\ st_{\text{Lap}} \xrightarrow{\text{reduce}} st_1 \\ \Downarrow \qquad\qquad \Downarrow \\ \Omega := B_k \qquad\quad L := (st_1).L_{BB} \end{array}$$

$$
\begin{array}{ccccc}
 & & \Downarrow & & \\
 & & \beta := \beta_{\mathrm{A}} = 1 + |\mathrm{signalFromA}| & & \\
 & \Downarrow & & & \\
 & N_{\mathrm{ToC}} = \mathrm{boundaryMatrixLocalNet}(\Omega) & & & \\
 & \swarrow \quad \downarrow \quad \searrow & & & \\
\rho_{\mathrm{top}}^{\mathrm{strong}} \to \rho_{\mathrm{top}}, \rho_r \to \Phi_{\mathrm{FromA}} & & \mathcal{A}_\infty(N_{\mathrm{ToC}}) \cong \mathrm{Mat}(\Omega) & & \mathrm{GNS}(\rho_{\mathrm{top}}) \\
 & & \uparrow & & \\
 & & \alpha^-_{\mathrm{top}} \leftarrow \alpha^-_{\mathrm{full}}(L) & & \\
 & & \downarrow & & \\
 & & \mathrm{KMS}_{\mathrm{top}}(L, \beta) & & \\
 & & \downarrow & & \\
 & & \texttt{modularDatum\_top\_ToC}. & &
\end{array}
$$

13 Gesamtsatz des strikten Pfads

Theorem 13.1 (Formale Ableitungskette des strikten ToC→Pillar-B-Rich-Pfads)**.** *Sei $b \in \mathbb{N}$ ein Verzweigungsfaktor und p eine Policy mit Trunkierungsregulator $L_{\max}$. Zusätzlich sei $0 < b$, sodass der exportierte Randträger nichtleer ist. Dann ist in* `REALOQS_v0.0604` *kanonisch und ohne zusätzliche freie Modellparameter jenseits von b und $L_{\max}$ folgendes abgeleitet:*

(1) ein endlicher ToC-Approximant (V_k, B_k, I_k, L_0),

(2) daraus per stabilisierter DtN-/Kron-Elimination ein symmetrischer Randoperator L,

(3) daraus und aus der A-seitigen Scale-Breaking-Diagnostik eine positive inverse Temperatur β,

(4) darauf das reiche lokale Netz N_{ToC} auf $\Omega = B_k$,

(5) darauf einen kompatiblen lokalen und quasilokalen Zustandszweig $\rho_r, \Phi_{\mathrm{FromA}}$,

(6) unabhängig davon eine kanonische Identifikation des quasilokalen Abschlusses mit $\mathrm{Mat}(\Omega)$,

(7) aus dem Topzustand ein konkretes GNS-Datum,

(8) aus L die Minus-Heisenbergdynamik,

(9) aus (L, β) einen starken und einen Top-KMS-Zustand,

(10) und schließlich das kohärente Endobjekt `modularDatum_top_ToC`*.*

Insbesondere ist der gesamte aktive Rich/Closure/State-Dynamics-Pfad von der ToC-Konstruktion bis zum Modular-Datum formal geschlossen.

Beweis. (1) folgt aus den Abschnitten 2 und 3.
(2) folgt aus `reduce_LBB_eq_dtn_stab` und `L_transpose_eq`.
(3) folgt aus der Definition von `betaFromA` und `betaFromA_pos`.
(4) folgt aus der Definition von `N_ToC=boundaryMatrixLocalNet`.
(5) folgt aus ρ`TopStrong`, ρ`Region`, `boundaryMatrixRegionStrong_compat`, `boundaryMatrixStateConeFromStrongTop` und `boundaryMatrixQuasiLocalStateFromStrongTop_restrict`.
(6) folgt aus `boundaryMatrixTopIsoMatrix`, `boundaryMatrixQuasiLocalIsoTop` und `boundaryMatrixQuasiLocalIsoMatrix`.
(7) folgt aus `boundaryMatrixTopGNSDatum`.
(8) folgt aus `fullBoundaryDynMinus`, `boundaryMatrixTopDynMinus` und `dyn_top_ToC`.

(9) folgt aus `boundaryFullKMSMinusAtBeta`, `boundaryMatrixTopKMSMinusAtBeta` und `kmsDatum_top_ToC`.
(10) folgt aus `boundaryMatrixTopModularDatum` bzw. `modularDatum_top_ToC`.
Damit ist der Pfad geschlossen. □

14 Was dieser Pfad beweist – und was nicht

Bemerkung 14.1 (Was bewiesen ist)**.** Bewiesen ist eine finitedimensionale operatoralgebraische Struktur auf dem exportierten ToC-Randträger:

- konkretes lokales Netz aus lokalisierten Matrizen,
- kompatibler lokaler und quasilokaler Zustandszweig,
- konkretes GNS-Datum,
- explizite Minus-Heisenbergdynamik,
- starker und transportierter KMS-Zustand,
- gebündeltes Modular-Datum.

Bemerkung 14.2 (Was *nicht* bewiesen ist)**.** Nicht bewiesen ist hier insbesondere:

- eine continuum-lokale AQFT auf einer vorgegebenen Raumzeit,
- eine Wightman-Theorie,
- Reeh–Schlieder, Typ III oder Tomita–Takesaki in der üblichen QFT-Stärke,
- bereits die vollständige Identifikation des minimalen Netzes mit einem echten HK-Netz jenseits der hier eingebauten Matrixträgerstruktur.

Das Modular-Datum ist hier ein *endliches* topfaserbasiertes modular/KMS/GNS-Paket, nicht automatisch das Endstadium einer physikalisch vollständigen AQFT.

A Dateien und Lastträger des Pfads

`PillarA/Core/Policy.lean`
Definiert die primitive Eingabe `Policy`; auf dem hier behandelten Pfad bleibt davon nur `L_max` als freier Trunkierungsregulator übrig.

`PillarA/Ideal/Substrate.lean`
Grundlegender Adressraum des ToC: `Addr`, `cellsAtLevel`, `cellsUpTo` sowie der Charakterisierungssatz `mem_cellsAtLevel_iff_length`.

`PillarA/Ideal/TreeOfCliques/Boundary.lean`
Definiert `Boundary` und `Interior` samt Einbettungen in den Vertex-Typ.

`PillarA/Ideal/TreeOfCliques/Split.lean`
Liefert die kanonische Zerlegung des Vertexraums über `equivVertexSum` und `split`.

`PillarA/Ideal/TreeOfCliques/Edges.lean`
Adjazenz- und Kantengewichtsstruktur: `Adj`, `conductance` sowie Symmetrie- und Nichtnegativitätsresultate.

`PillarA/Ideal/TreeOfCliques/Laplacian.lean`
Definiert den ToC-Laplacian `laplacian`; lasttragend ist `gate_LAPLACIAN_ADMISSIBLE_treeOfCliques`.

PillarA/Ideal/Adapter/TreeOfCliquesApproxDtNStabilized.lean
Kern des A-seitigen Operatorpfads: L0, hpsd0_proved, hdet0_inst, stLap, reduce, reduce_LBB_eq_dtn_stab.

PillarA/Ideal/Adapter/TreeOfCliquesAQFTHandoff.lean
Exportoberfläche nach Pillar B: Ω, st0, st1, L, L_transpose_eq, signalFromA, betaFromA, dFromA, phiFromA.

PillarB/AQFT/Derived/BoundaryMatrixLocalNet.lean
Lokales Rich-Netz auf dem Randträger: SupportedOn, RegionLocalized, RegionBlockMat, boundaryMatrixLocalNet, boundaryMatrixTopIsoMatrix.

PillarB/AQFT/Derived/BoundaryMatrixStateNet.lean
Zustandszweig: boundaryMatrixRegionStrong, boundaryMatrixTop, boundaryMatrixStateConeFromStrongTop, boundaryMatrixQuasiLocalStateFromStrongTop.

PillarB/AQFT/Derived/BoundaryMatrixQuasiLocalClosure.lean
Abschlusszweig: boundaryMatrixQuasiLocalIsoTop und boundaryMatrixQuasiLocalIsoMatrix.

PillarB/AQFT/Derived/BoundaryMatrixGNS.lean
GNS-Zweig: boundaryMatrixGNSSesq, boundaryMatrixTopGNSPi, boundaryMatrixGNSCyclicVec, boundaryMatrixTopGNSDatum.

PillarB/AQFT/Derived/BoundaryMatrixFullDynamics.lean
Dynamikzweig mit fullBoundaryDynMinus als expliziter Minus-Heisenberg-Zeitentwicklung.

PillarB/AQFT/Derived/BoundaryMatrixFullKMSFromL.lean
Voller Matrix-KMS-Zweig: boundaryDensityStateAtBeta, fullKMSKernelMinusAtBeta, boundaryFullKMSMinusAtBeta.

PillarB/AQFT/Derived/BoundaryMatrixModular.lean
Transport- und Modularzweig: boundaryMatrixTopDynMinus, boundaryMatrixTopKMSMinusAtBeta, BoundaryMatrixModularDatum, boundaryMatrixTopModularDatum.

PillarB/AQFT/Derived/TreeOfCliquesFromPillarA.lean
B-seitige ToC-Oberfläche mit N_ToC, ρTopStrong, ρTop, ρRegion, Φ_FromA, quasiLocalClosureIsoMatrix_FromA, gnsDatum_top_ToC, dyn_top_ToC, kmsDatum_top_ToC, modularDatum_top_ToC.

PillarB/AQFT/Derived/TreeOfCliquesExtendedFromPillarA.lean
Erweiterte strikte Oberfläche oberhalb des Kernpfads: D_ToC, locality_ToC, isotony_ToC, splitProperty_ToC, T_beta_eq_betaFromA sowie weitere orientierte Dynamik- und KMS-Aliase. Diese Datei ist für die im Meta-Anhang benutzten Locality/Isotony/Split-Marker lasttragend.

PillarA/Core/DirichletLaplacian.lean
Grundlagen der Dirichlet-Quadratform, insbesondere quadForm. Diese Datei trägt im ToC-Pfad die Positivitätsargumente für quadForm_L0_nonneg und damit den semidefiniten Schritt für $L_{II}^{(0)}$.

PillarA/OQS/SysEnv.lean
Blockzerlegungs-Grundlagen des OQS-Splits, insbesondere blockBB_transpose_of_symm und blockII_transpose_of_symm. Diese Sätze tragen auf A-Seite die Symmetrievererbung der Rand- und Innenblöcke.

Meta/PillarB_Closure_FromA_Strict.lean
Meta-Marker für quasilokale Restriktion, Locality, Isotony, Split, Topzustand und GNS-Zustandsrekonstruktion.

B Meta-Marker außerhalb des unmittelbaren Rich/State-Dynamics-Pfads

Die Datei `Meta/PillarB_Closure_FromA_Strict.lean` enthält Marker, die den strikten Pfad auf Meta-Ebene absichern. Im Code werden dort ausdrücklich *zwei* Namespaces geöffnet:

`TreeOfCliquesFromPillarA.ToC`,
`TreeOfCliquesExtendedFromPillarA.ToC`.

Daher stammen `N_ToC`, ρ`Top`, `gnsDatum_top_ToC`, `dyn_top_ToC`, `kmsDatum_top_ToC` und `modularDatum_top_ToC` aus `TreeOfCliquesFromPillarA.ToC`, während `D_ToC`, `locality_ToC`, `isotony_ToC` und `splitProperty_ToC` aus `TreeOfCliquesExtendedFromPillarA.ToC` kommen. Ohne genau diese Zweifach-Öffnung wäre die folgende Markeroberfläche nicht vollständig bestimmbar:

`toc_quasilocal_univ_marker`,
`toc_locality_marker`,
`toc_isotony_marker`,
`toc_split_marker`,
`toc_top_state_marker`,
`toc_gns_state_from_cyclic_marker`.

Diese Marker sind nicht die primären Definitionslastträger des hier behandelten Zustands-/Dynamik-/Modularpfads, aber sie bestätigen, dass die Oberflächenobjekte auf der Meta-Ebene korrekt zusammenpassen: Der in `modularDatum_top_ToC` enthaltene Top-KMS-Zustand ist tatsächlich ρ`Top`, und die GNS-Rekonstruktion des Zustands wird nochmals separat bestätigt.