

FORMALIZING THE MATRIX-TREE THEOREM USING LEAN AND ARISTOTLE

22 June 2026

KMS 80th Anniversary International Conference

Session on Machine-Assisted Mathematics: Formalization and Automated Reasoning

Harry Richman (NCTS, Taipei)

MATRIX-TREE THEOREM

Setup:

- G = finite graph
- L_0 = reduced Laplacian matrix
- *spanning tree* = subgraph which is connected, acyclic, contains all vertices

Theorem. (Kirchhoff, 1860s)

$$\det L_0 = \#(\text{spanning trees})$$

MATRIX-TREE THEOREM: PARTS

Using:

- Factorization with incidence matrix $L = BB^T$

Theorem. (Kirchhoff, 1860s)

$$\det L_0 = \#(\text{spanning trees})$$

↓

$$\det(B_0 B_0^T) = \#(\text{spanning trees})$$

MATRIX-TREE THEOREM: PARTS

Using:

- Cauchy-Binet identity

Theorem. (Kirchhoff, 1860s)

$$\det(B_0 B_0^T) = \#(\text{spanning trees})$$

↓

$$\sum_{|I|=n-1} (\det B_0[\cdot, I]) (\det B_0^T[I, \cdot]) = \#(\text{spanning trees})$$

MATRIX-TREE THEOREM: PARTS

Using:

- Reduced incidence matrix minors are ± 1 or 0

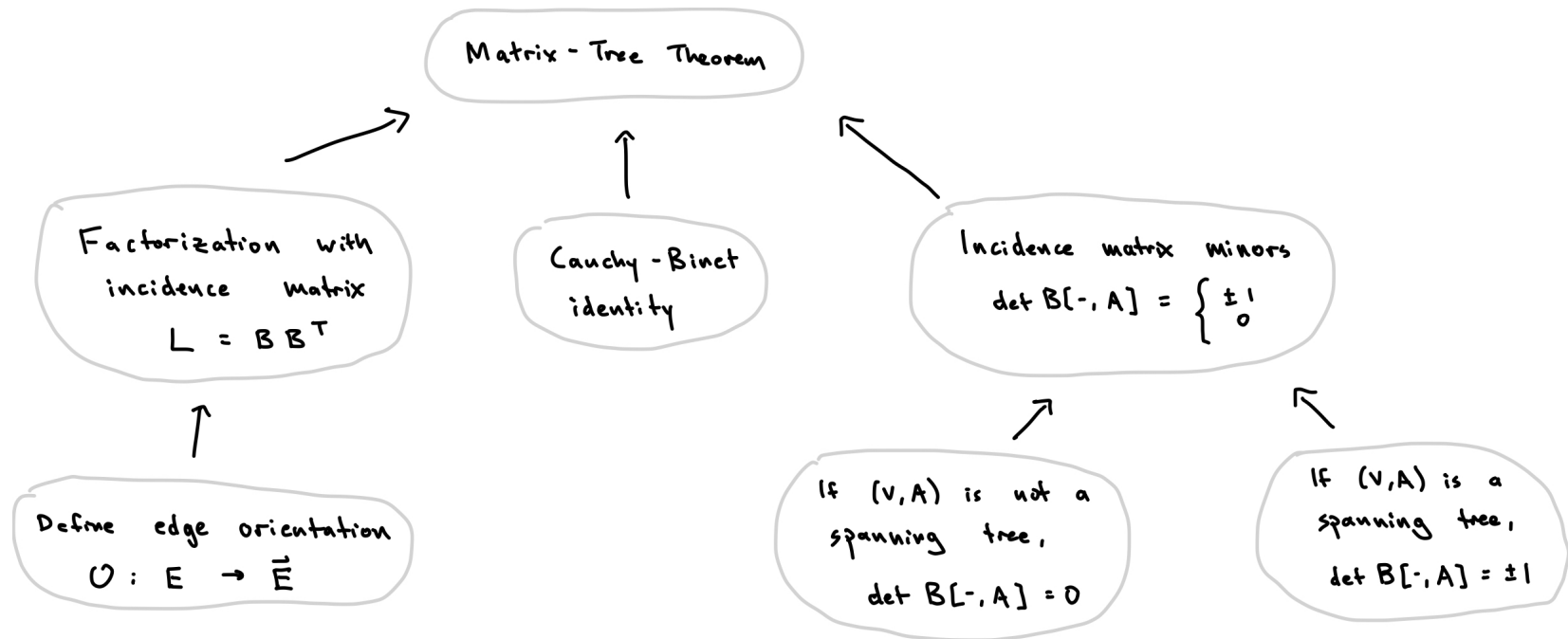
Theorem. (Kirchhoff, 1860s)

$$\sum_{|I|=n-1} (\det B_0[\cdot, I])^2 = \#(\text{spanning trees})$$

↓

$$\#(I : \det B_0[\cdot, I] \neq 0) = \#(\text{spanning trees})$$

MATRIX-TREE THEOREM: PARTS



AUTOFORMALIZATION WITH ARISTOTLE

- Asked to formalize Matrix-Tree theorem
 - (March 2026) failed, but provided a theorem statement [\[link\]](#)
- Asked to formalize one of the three "parts"
 - (March 2026) Cauchy-Binet succeeded [\[link, 131 lines\]](#)
 - (June 2026) Incidence matrix factorization succeeded [\[link, 97 lines\]](#)
 - (June 2026) Incidence matrix minors succeeded [\[link pt. 1, 159 lines\]](#) [\[link pt. 2, 393 lines\]](#)

AUTOFORMALIZATION WITH ARISTOTLE

- Asked to formalize Matrix-Tree theorem
 - (March 2026) failed, but provided a theorem statement [\[link\]](#)

```
Output summary 04:49 AM

Here's what it did:
Matrix Tree Theorem Formalization in Lean
Status in Mathlib
As of Mathlib v4.28.0, the Matrix Tree Theorem (Kirchhoff's Theorem) has not been formalized in Lean 4 /
Mathlib, and I am not aware of any public standalone formalization in Lean 4. Mathlib does have the Laplacian
matrix (SimpleGraph.lapMatrix) and some spectral properties (e.g., card_ConnectedComponent_eq_rank_ker_lapMatrix),
but not the matrix tree theorem itself.

What I've provided
I created a clean formal statement of the theorem in RequestProject/MatrixTreeTheorem.lean, including:
```

•
•
•

The Cauchy-Binet formula – $\det(A * B^T) = \sum_S \det(A_S) \cdot \det(B_S)$ for rectangular matrices – is not in Mathlib. Oriented incidence matrices and the factorization $L = B \cdot B^T$ would need to be built from scratch. The combinatorial argument that $\det(\tilde{B}_S)^2 \in \{0, 1\}$ with value 1 iff S forms a spanning tree requires additional graph-theoretic lemmas. Building this infrastructure from scratch would be a significant multi-day formalization project (likely 1000+ lines of Lean).

-  Aristotle finished



AUTOFORMALIZATION WITH ARISTOTLE

- Asked to formalize Matrix-Tree theorem
 - (March 2026) failed, but provided a theorem statement [\[link\]](#)

•
•
•

The Cauchy-Binet formula – $\det(A * B^T) = \sum_S \det(A_S) \cdot \det(B_S)$ for rectangular matrices – is not in Mathlib. Oriented incidence matrices and the factorization $L = B \cdot B^T$ would need to be built from scratch. The combinatorial argument that $\det(\tilde{B}_S)^2 \in \{0, 1\}$ with value 1 iff S forms a spanning tree requires additional graph-theoretic lemmas. Building this infrastructure from scratch would be a significant multi-day formalization project (likely 1000+ lines of Lean).

• ☑ Aristotle finished



CAUCHY-BINET IDENTITY

Setup:

- $A = m \times n$ matrix
- $B = n \times n$ matrix
- submatrix notation: $A[\cdot, I]$

Theorem.

$$\det(AB) = \sum_{|I|=m} (\det A[\cdot, I]) (\det B[I, \cdot])$$

CAUCHY-BINET IN LEAN

Theorem.

$$\det(AB) = \sum_{|I|=m} (\det A[\cdot, I]) (\det B[I, \cdot])$$

```
variable {R : Type*} [CommRing R]
variable {M N : Type} [Fintype M] [Fintype N] [DecidableEq M] [DecidableEq N]
  [LinearOrder M] [LinearOrder N]

/-- Cauchy-Binet, https://en.wikipedia.org/wiki/Cauchy%E2%80%93Binet -/
theorem Matrix.det_mul' (A : Matrix M N R) (B : Matrix N M R) :
  det (A * B) = ∑ f : M ↪o N,
  det (A.submatrix id f) * det (B.submatrix f id) := by
  ...
```

CAUCHY-BINET IN LEAN

Many choices for "spelling" formalization

```
variable [Fintype M] ... [LinearOrder M] [LinearOrder N]

theorem Matrix.det_mul' (A : Matrix M N R) (B : Matrix N M R) :
  det (A * B) =  $\sum f : M \hookrightarrow_o N,$ 
  det (A.submatrix id f) * det (B.submatrix f id) := by
  ...
```

- From Meta's autoformalizing textbooks project

```
theorem cauchyBinet {n m :  $\mathbb{N}$ }
  (A : Matrix (Fin n) (Fin m) R) (B : Matrix (Fin m) (Fin n) R) :
  (A * B).det =  $\sum S \in (\text{Finset.univ} : \text{Finset (Fin m)}).powersetCard$ 
  if h : S.card = n then
  (colsSubmatrix A S h).det * (rowsSubmatrix B S h).det
  else 0 := by
  ...
```


CAUCHY-BINET IN LEAN

Many choices for "spelling" formalization

```
theorem Matrix.det_mul' (A : Matrix M N R) (B : Matrix N M R) :  
  det (A * B) =  $\sum f : M \hookrightarrow N$ ,  
  det (A.submatrix id f) * det (B.submatrix f id) := by  
  ...
```

- From Ho Boon Suan [project](#)

```
/-! We prove the Cauchy-Binet formula in the "function form":  
`det(A * B) =  $\sum f : m \rightarrow n$ , ( $\prod i$ , A i (f i)) * det(B.submatrix f id)  
and then restrict to injective functions.  
... -/
```

```
theorem Matrix.det_mul_sum (A : Matrix m n R) (B : Matrix n m R) :  
  (A * B).det =  $\sum f : m \rightarrow n$ ,  
  ( $\prod i$ , A i (f i)) * (B.submatrix f id).det := by  
  ...
```


FRICTION USING ARISTOTLE

- Some lemmas are repeats of Mathlib

```
20 |
21 |/-- Column permutation of determinant. -/
22 |lemma Matrix.det_submatrix_id_perm {n : Type} [Fintype n] [DecidableEq n]
23 | | (A : Matrix n n R) (σ : Equiv.Perm n) :
24 | | (A.submatrix id σ).det = Equiv.Perm.sign σ * A.det := by
25 | | rw [← det_transpose, transpose_submatrix, det_permute, det_transpose]
26 |
```

... but in Mathlib docs:

theorem Matrix.det_permute[source](#)

```
{n : Type u_2} [DecidableEq n] [Fintype n] {R : Type v} [CommRing R] (σ : Equiv.Perm n)
(M : Matrix n n R) :
(M.submatrix (↑σ) id).det = ↑↑(Equiv.Perm.sign σ) * M.det
```

Permuting the columns changes the sign of the determinant.

theorem Matrix.det_permute'[source](#)

```
{n : Type u_2} [DecidableEq n] [Fintype n] {R : Type v} [CommRing R] (σ : Equiv.Perm n)
(M : Matrix n n R) :
(M.submatrix id ↑σ).det = ↑↑(Equiv.Perm.sign σ) * M.det
```

Permuting the rows changes the sign of the determinant.

FRICTION USING ARISTOTLE

```
20 |  
21 |/-- Column permutation of determinant. -/  
22 lemma Matrix.det_submatrix_id_perm {n : Type} [Fintype n] [DecidableEq n]  
23 | (A : Matrix n n R) (σ : Equiv.Perm n) :  
24 | (A.submatrix id σ).det = Equiv.Perm.sign σ * A.det := by  
25 | rw [← det_transpose, transpose_submatrix, det_permute, det_transpose]  
26 |
```

... but in Mathlib docs:

theorem Matrix.det_permute

[source](#)

```
{n : Type u_2} [DecidableEq n] [Fintype n] {R : Type v} [CommRing R] (σ : Equiv.Perm n)  
(M : Matrix n n R) :  
(M.submatrix (↑σ) id).det = ↑↑(Equiv.Perm.sign σ) * M.det
```

Permuting the columns changes the sign of the determinant.

theorem Matrix.det_permute'

[source](#)

```
{n : Type u_2} [DecidableEq n] [Fintype n] {R : Type v} [CommRing R] (σ : Equiv.Perm n)  
(M : Matrix n n R) :  
(M.submatrix id ↑σ).det = ↑↑(Equiv.Perm.sign σ) * M.det
```

Permuting the rows changes the sign of the determinant.

FRICTION USING LEAN

- Dummy variables overwritten in sum manipulation

```
1 import Mathlib
2
3 variable {M N : Type} [Fintype M] [Fintype N] [DecidableEq M]
4 [DecidableEq N] [LinearOrder M] [LinearOrder N]
5
6 theorem Matrix.det_mul' (A : Matrix M N Z) (B : Matrix N M Z) :
7   det (A * B) =  $\sum f : M \hookrightarrow N$ , det (A.submatrix id f) * det (B.submatrix f id) := by
8   -- expand determinant in matrix entries as sum over permutations, on LHS
9   rw [Matrix.det_apply (A * B)] -- uses  $\sigma$  for summing permutations
10  -- expand entries of matrix product A*B
11  simp_rw [Matrix.mul_apply] -- uses  $x$  for summing permutations :(
12
13  sorry
```

[[Lean playground](#)]

- Zulip discussion [[link](#)]

CAUCHY-BINET: SMALL CASE

- Start with 2×3 matrices

```
variable {R : Type*} [CommRing R]
lemma det_mul_2_by_3 (A : Matrix (Fin 2) (Fin 3) R)
  (B : Matrix (Fin 3) (Fin 2) R) :
  det (A * B) =  $\sum f : (\text{Fin } 2) \hookrightarrow (\text{Fin } 3)$ ,
  det (A.submatrix id f) * det (B.submatrix f id) := by
  ...
```

CAUCHY-BINET: SMALL CASE

```
import Mathlib

open Matrix

lemma injec_2_3 : (Finset.univ : Finset ( Fin 2 ↪o Fin 3 )) =
{
  OrderEmbedding.ofStrictMono ( ![0,1] ) ( by decide ),
  OrderEmbedding.ofStrictMono ( ![0,2] ) ( by decide ),
  OrderEmbedding.ofStrictMono ( ![1,2] ) ( by decide )
} := by
ext f
simp [Finset.mem_univ]
rcases f with ⟨ f, hf ⟩
rcases f with ⟨ f, hf ⟩
fin_cases f <;> simp +decide at hf ⊢
```

[\[Lean playground\]](#)

FORMALIZING THE MATRIX-TREE THEOREM USING LEAN AND ARISTOTLE

```
variable {R : Type*} [CommRing R]
lemma det_mul_2_by_3 (A : Matrix (Fin 2) (Fin 3) R)
  (B : Matrix (Fin 3) (Fin 2) R) :
  det (A * B) =  $\sum f : (\text{Fin } 2) \hookrightarrow_0 (\text{Fin } 3)$ ,
  det (A.submatrix id f) * det (B.submatrix f id) := by
  ...
```

Thank you for listening!