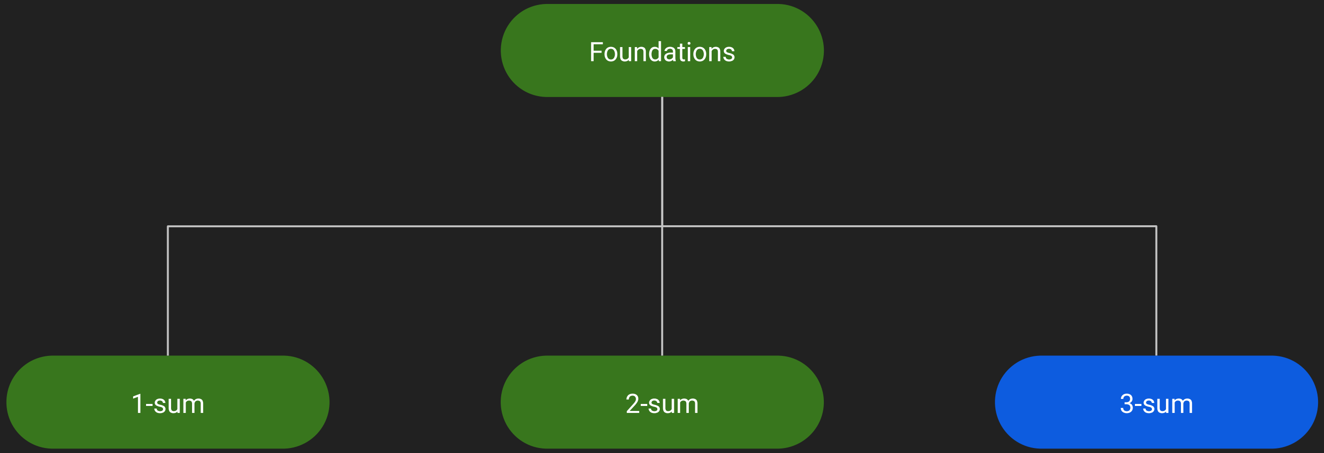
Seymour’s Theorem Formalization
Project Status Update

Martin Dvořák, Ivan Sergeev

12 June 2025

High-level Overview

Overall Status



Current Status of 3-Sum

01	3-Sum of Matrices	● Basic definitions ● API
02	Re-signing of Matrices	● Re-signing 3x3 matrices ● Re-signing matrices based on 3x3 submatrices ● Total unimodularity
03	Canonical Signing of 3-Sum	● Re-signing of summands ● Canonical signing of 3-sum ● Total unimodularity
04	Family of 3-Sum-Like Matrices	● Structural definition ● 3-sums are generalized ● Total unimodularity
05	3-Sum of Matroids	● Standard representations ● Matroids having a standard representation ● Regularity

Progress Summary

Formalization of 3-sum	
The 3-sum of matrices	● Improved basic definition and API
Canonical signing of matrices	● Already complete, fully retained
Canonical signing of 3-sum	● Stated all necessary lemmas ● Proved total unimodularity results about adjoining columns
Family of 3-sum-like matrices	● Stated definition and key lemmas ● Proved lemmas that directly reduce to simpler results
Blueprint	
Content & dependency graph	● Aligned more closely with implementation
Publicly displayed version	● Updated to current version
CI/CD workflows	● Re-enabled and updated

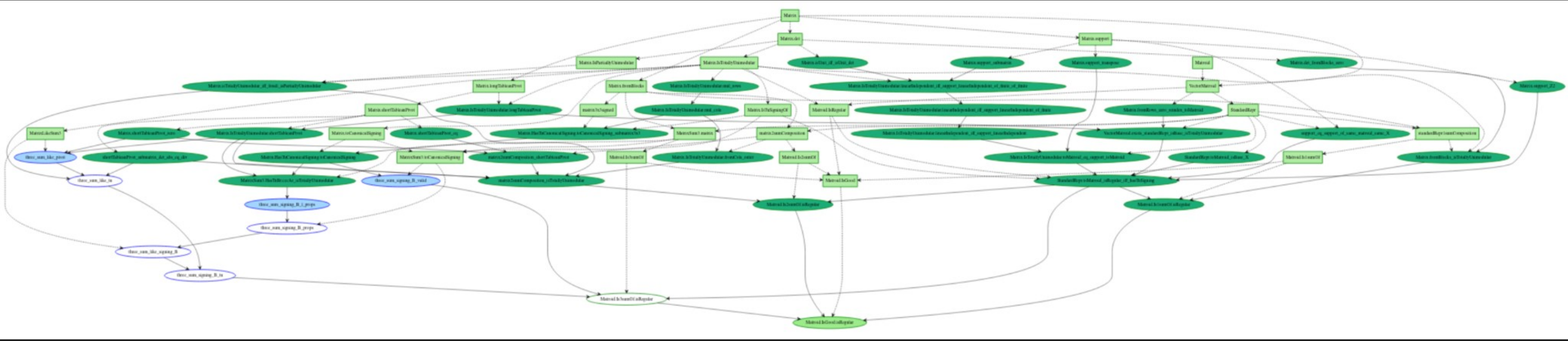
Next Steps

Family of 3-sum-like matrices	● State remaining results ● Prove remaining statements
Canonical signing of 3-sum	● Prove remaining statements
Blueprint	● Keep up-to-date
External contributors	● Potentially delegate selected proofs

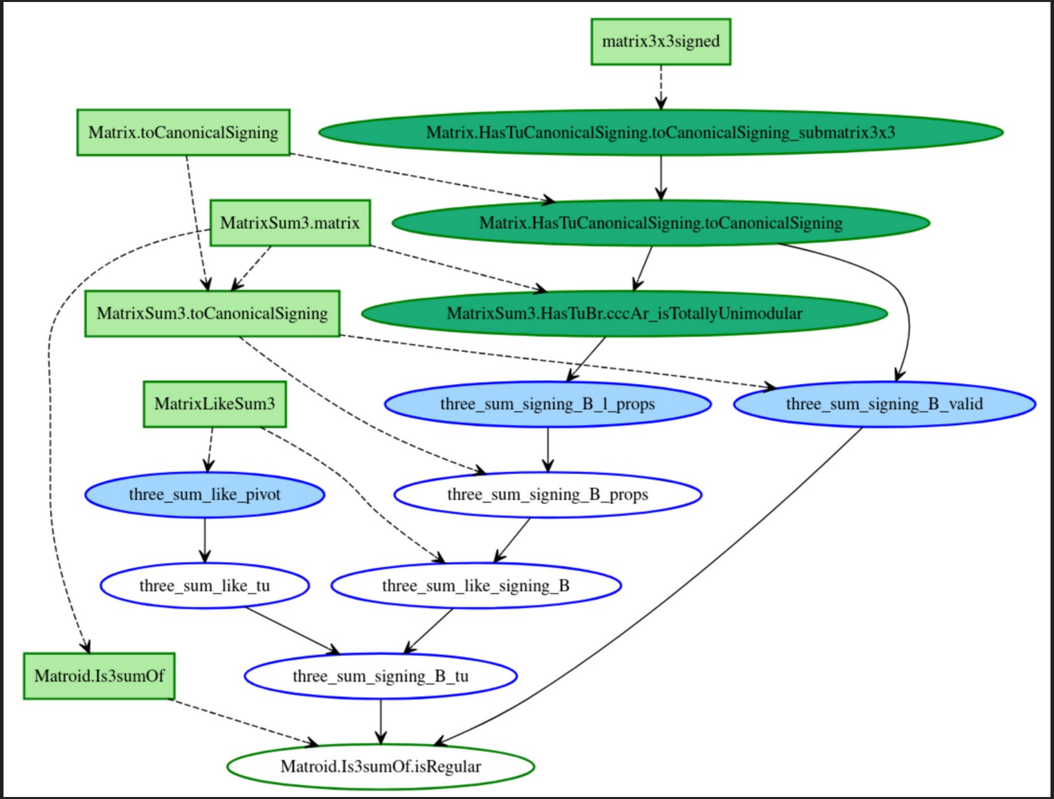
Additional Slides

Dependency Graph

Entire Dependency Graph

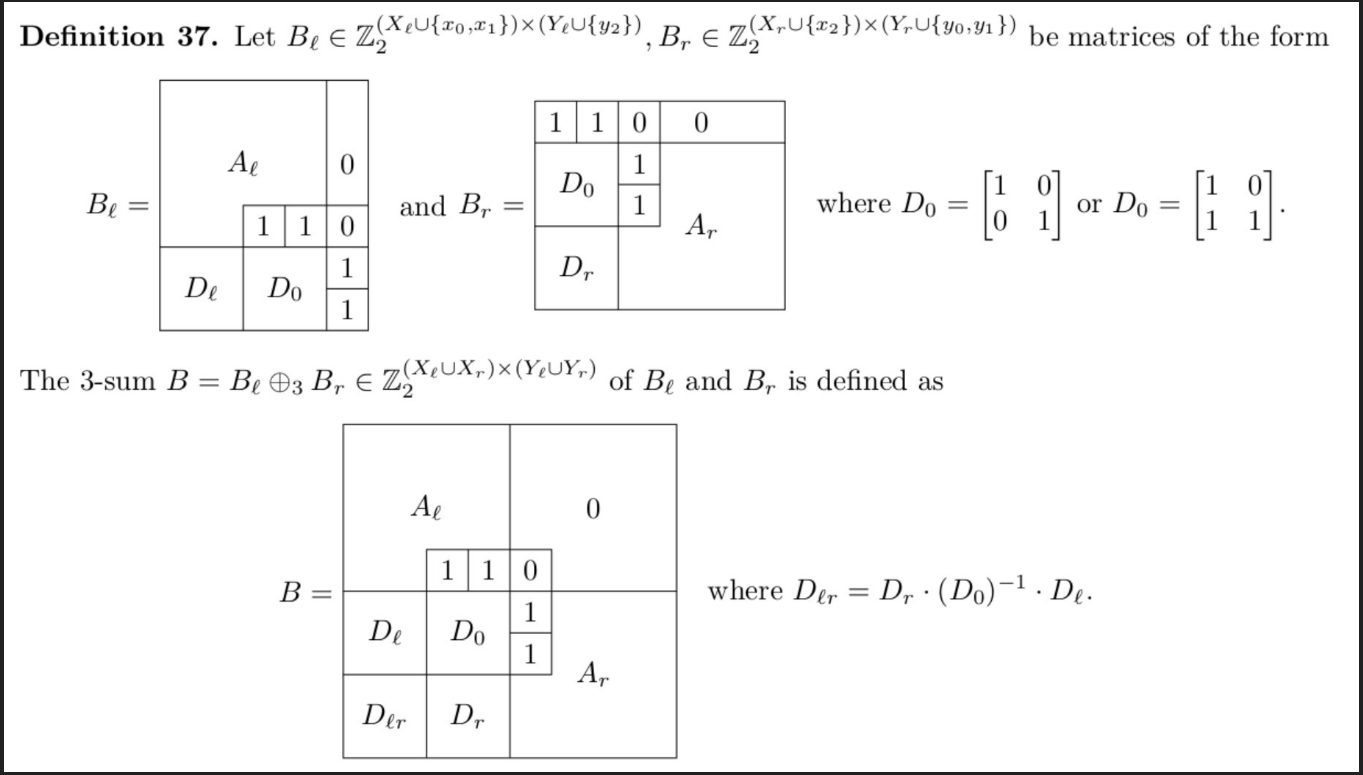


Dependency Graph of 3-Sum


matrix3x3signed
Matrix.toCanonicalSigning
Matrix.HasTuCanonicalSigning.toCanonicalSigning_submatrix3x3
MatrixSum3.matrix
Matrix.HasTuCanonicalSigning.toCanonicalSigning
MatrixSum3.toCanonicalSigning
MatrixSum3.HasTuBr.cccAr_isTotallyUnimodular
MatrixLikeSum3
three_sum_signing_B_l_props
three_sum_signing_B_valid
three_sum_like_pivot
three_sum_signing_B_props
three_sum_like_tu
three_sum_like_signing_B
Matroid.Is3sumOf
three_sum_signing_B_tu
Matroid.Is3sumOf.isRegular

Improvements in 3-Sum

Refresher on Matrix Structure



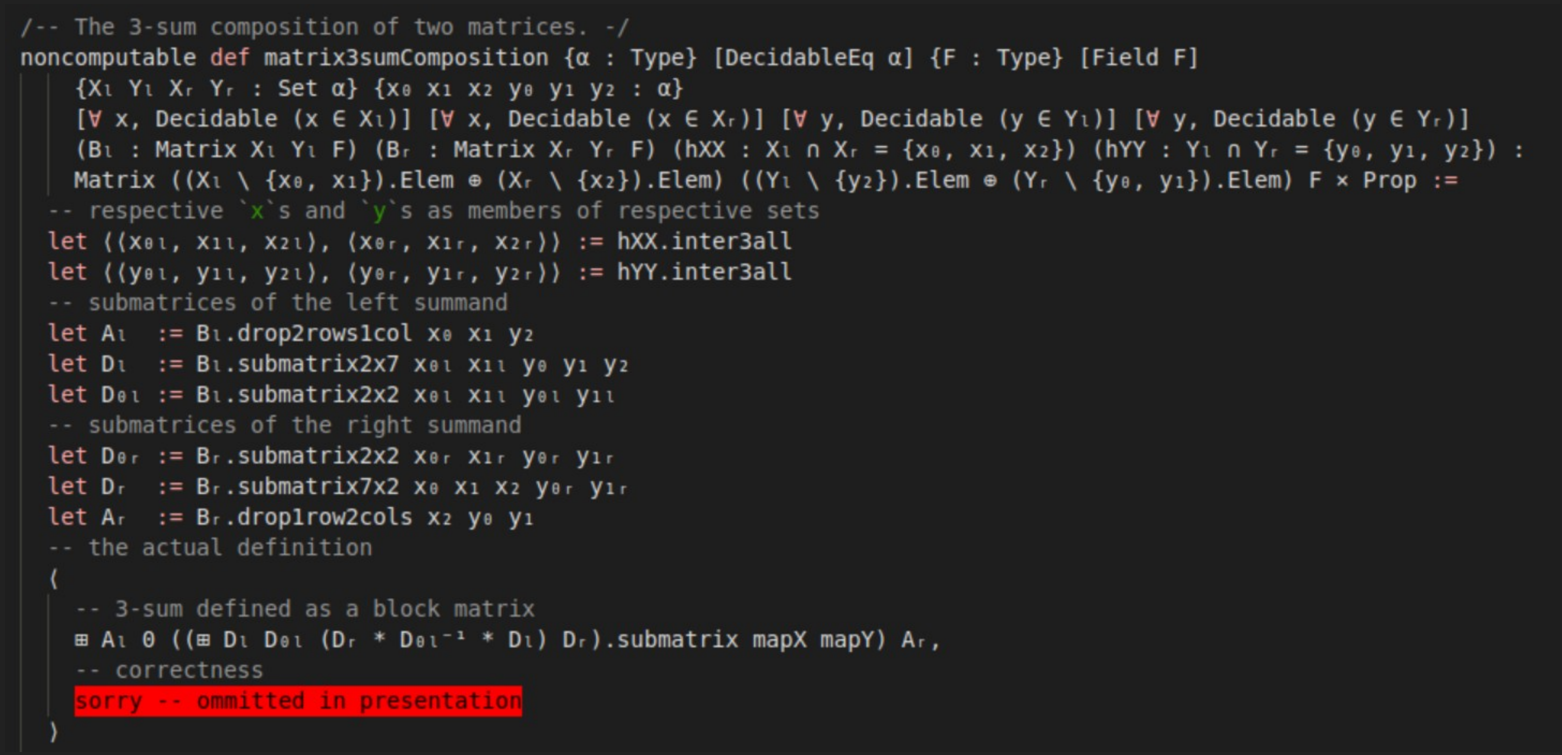
Definition 37. Let $B_\ell \in \mathbb{Z}_2^{(X_\ell \cup \{x_0, x_1\}) \times (Y_\ell \cup \{y_2\})}$, $B_r \in \mathbb{Z}_2^{(X_r \cup \{x_2\}) \times (Y_r \cup \{y_0, y_1\})}$ be matrices of the form

$$B_\ell = \begin{array}{|c|c|c|c|}\hline \multicolumn{3}{|c|}{A_\ell} & 0 \\ \cline{2-4} & 1 & 1 & 0 \\ \hline D_\ell & \multicolumn{2}{c|}{D_0} & 1 \\ \cline{4-4} & \multicolumn{2}{c|}{} & 1 \\ \hline \end{array} \quad \text{and } B_r = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 0 & 0 \\ \hline \multicolumn{2}{|c|}{D_0} & 1 & \\ \cline{3-3} \multicolumn{2}{|c|}{} & 1 & A_r \\ \cline{1-3} \multicolumn{2}{|c|}{D_r} & \multicolumn{2}{c|}{} \\ \hline \end{array} \quad \text{where } D_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ or } D_0 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

The 3-sum $B = B_\ell \oplus_3 B_r \in \mathbb{Z}_2^{(X_\ell \cup X_r) \times (Y_\ell \cup Y_r)}$ of B_ℓ and B_r is defined as

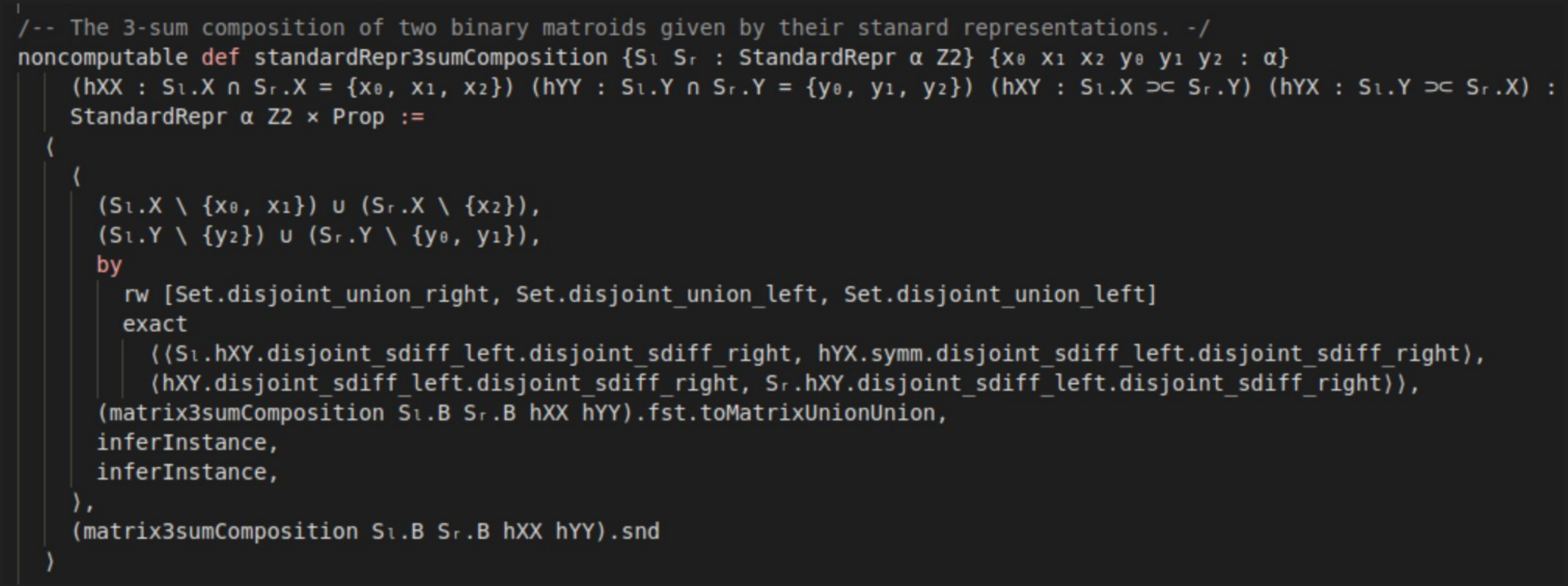
$$B = \begin{array}{|c|c|c|c|}\hline \multicolumn{2}{|c|}{A_\ell} & \multicolumn{2}{c|}{0} \\ \cline{2-3} & 1\ \ 1 & 0 & \\ \hline D_\ell & D_0 & 1 & \\ \cline{3-3} & & 1 & A_r \\ \hline D_{\ell r} & D_r & \multicolumn{2}{c|}{} \\ \hline \end{array} \quad \text{where } D_{\ell r} = D_r \cdot (D_0)^{-1} \cdot D_\ell.$$

Previous Implementation: Matrices



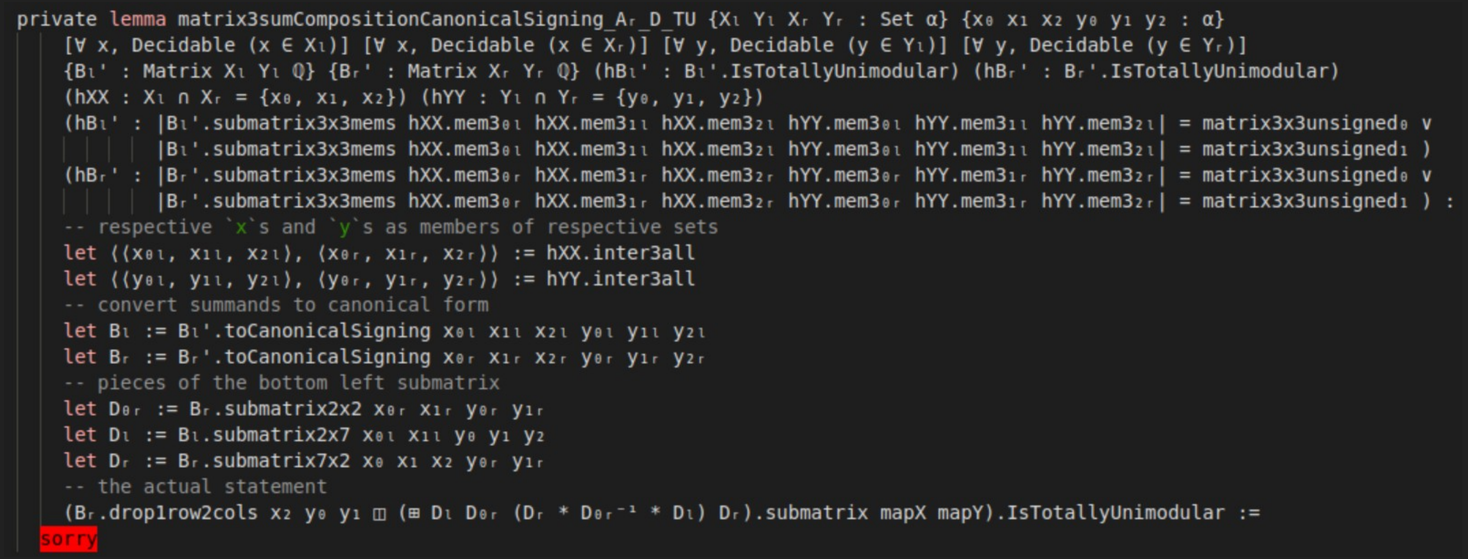
```
/-- The 3-sum composition of two matrices. -/
noncomputable def matrix3sumComposition {α : Type} [DecidableEq α] {F : Type} [Field F]
    {Xₗ Yₗ Xᵣ Yᵣ : Set α} {x₀ x₁ x₂ y₀ y₁ y₂ : α}
    [∀ x, Decidable (x ∈ Xₗ)] [∀ x, Decidable (x ∈ Xᵣ)] [∀ y, Decidable (y ∈ Yₗ)] [∀ y, Decidable (y ∈ Yᵣ)]
    (Bₗ : Matrix Xₗ Yₗ F) (Bᵣ : Matrix Xᵣ Yᵣ F) (hXX : Xₗ ∩ Xᵣ = {x₀, x₁, x₂}) (hYY : Yₗ ∩ Yᵣ = {y₀, y₁, y₂}) :
    Matrix ((Xₗ \ {x₀, x₁}).Elem ⊕ (Xᵣ \ {x₂}).Elem) ((Yₗ \ {y₂}).Elem ⊕ (Yᵣ \ {y₀, y₁}).Elem) F × Prop :=
  -- respective `x`s and `y`s as members of respective sets
  let ⟨⟨x₀ₗ, x₁ₗ, x₂ₗ⟩, ⟨x₀ᵣ, x₁ᵣ, x₂ᵣ⟩⟩ := hXX.inter3all
  let ⟨⟨y₀ₗ, y₁ₗ, y₂ₗ⟩, ⟨y₀ᵣ, y₁ᵣ, y₂ᵣ⟩⟩ := hYY.inter3all
  -- submatrices of the left summand
  let Aₗ  := Bₗ.drop2rows1col x₀ x₁ y₂
  let Dₗ  := Bₗ.submatrix2x7 x₀ₗ x₁ₗ y₀ y₁ y₂
  let D₀ₗ := Bₗ.submatrix2x2 x₀ₗ x₁ₗ y₀ₗ y₁ₗ
  -- submatrices of the right summand
  let D₀ᵣ := Bᵣ.submatrix2x2 x₀ᵣ x₁ᵣ y₀ᵣ y₁ᵣ
  let Dᵣ  := Bᵣ.submatrix7x2 x₀ x₁ x₂ y₀ᵣ y₁ᵣ
  let Aᵣ  := Bᵣ.drop1row2cols x₂ y₀ y₁
  -- the actual definition
  ⟨
    -- 3-sum defined as a block matrix
    ⊞ Aₗ 0 ((⊞ Dₗ D₀ₗ (Dᵣ * D₀ₗ⁻¹ * Dₗ) Dᵣ).submatrix mapX mapY) Aᵣ,
    -- correctness
    sorry -- ommitted in presentation
  ⟩
```

Previous Implementation: Standard Representations



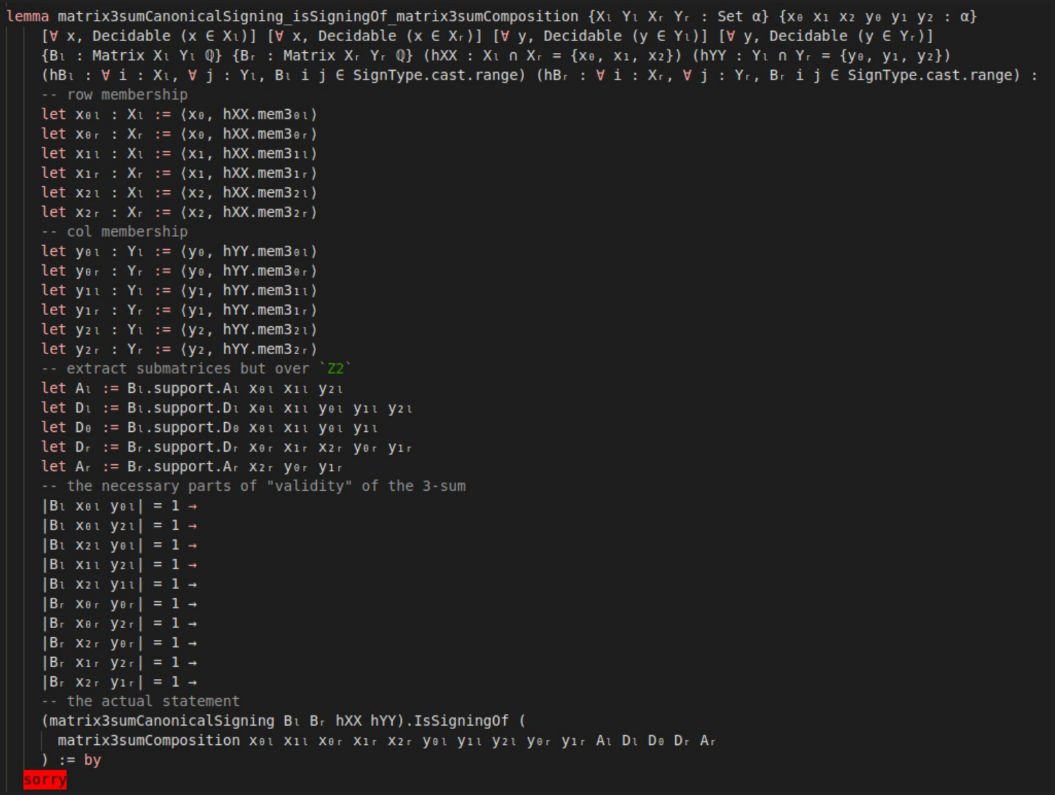
```
/-- The 3-sum composition of two binary matroids given by their stanard representations. -/
noncomputable def standardRepr3sumComposition {Sₗ Sᵣ : StandardRepr α Z2} {x₀ x₁ x₂ y₀ y₁ y₂ : α}
    (hXX : Sₗ.X ∩ Sᵣ.X = {x₀, x₁, x₂}) (hYY : Sₗ.Y ∩ Sᵣ.Y = {y₀, y₁, y₂}) (hXY : Sₗ.X ⫗ Sᵣ.Y) (hYX : Sₗ.Y ⫗ Sᵣ.X) :
    StandardRepr α Z2 × Prop :=
  ⟨
    ⟨
      (Sₗ.X \ {x₀, x₁}) ∪ (Sᵣ.X \ {x₂}),
      (Sₗ.Y \ {y₂}) ∪ (Sᵣ.Y \ {y₀, y₁}),
      by
        rw [Set.disjoint_union_right, Set.disjoint_union_left, Set.disjoint_union_left]
        exact
          ⟨⟨Sₗ.hXY.disjoint_sdiff_left.disjoint_sdiff_right, hYX.symm.disjoint_sdiff_left.disjoint_sdiff_right⟩,
          ⟨hXY.disjoint_sdiff_left.disjoint_sdiff_right, Sᵣ.hXY.disjoint_sdiff_left.disjoint_sdiff_right⟩⟩,
      (matrix3sumComposition Sₗ.B Sᵣ.B hXX hYY).fst.toMatrixUnionUnion,
      inferInstance,
      inferInstance,
    ⟩,
    (matrix3sumComposition Sₗ.B Sᵣ.B hXX hYY).snd
  ⟩
```

First Signs of Trouble



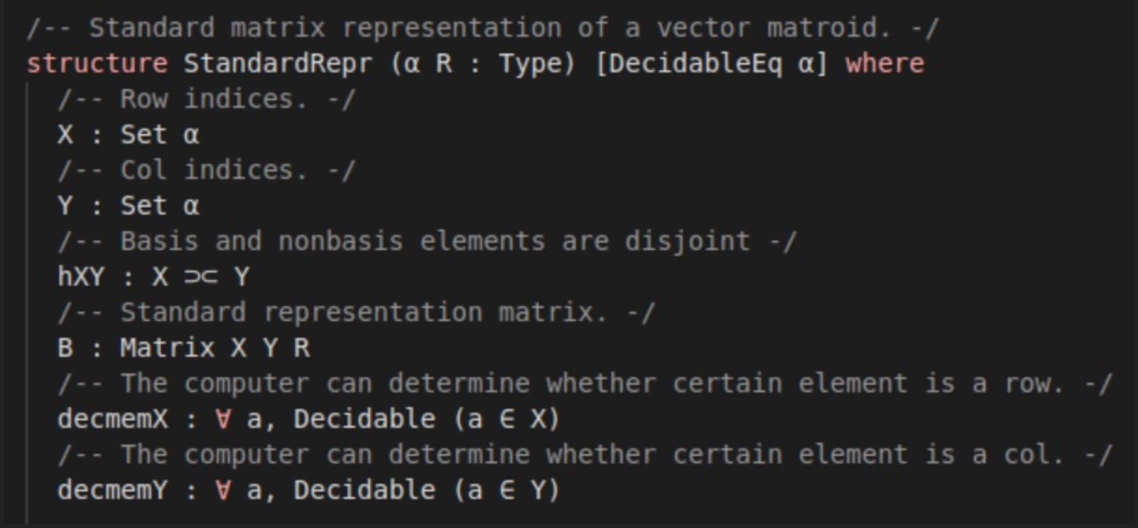
```
private lemma matrix3sumCompositionCanonicalSigning_Aᵣ_D_TU {Xₗ Yₗ Xᵣ Yᵣ : Set α} {x₀ x₁ x₂ y₀ y₁ y₂ : α}
    [∀ x, Decidable (x ∈ Xₗ)] [∀ x, Decidable (x ∈ Xᵣ)] [∀ y, Decidable (y ∈ Yₗ)] [∀ y, Decidable (y ∈ Yᵣ)]
    {Bₗ' : Matrix Xₗ Yₗ ℚ} {Bᵣ' : Matrix Xᵣ Yᵣ ℚ} (hBₗ' : Bₗ'.IsTotallyUnimodular) (hBᵣ' : Bᵣ'.IsTotallyUnimodular)
    (hXX : Xₗ ∩ Xᵣ = {x₀, x₁, x₂}) (hYY : Yₗ ∩ Yᵣ = {y₀, y₁, y₂})
    (hBₗ' : |Bₗ'.submatrix3x3mems hXX.mem3₀ₗ hXX.mem3₁ₗ hXX.mem3₂ₗ hYY.mem3₀ₗ hYY.mem3₁ₗ hYY.mem3₂ₗ| = matrix3x3unsigned₀ ∨
            |Bₗ'.submatrix3x3mems hXX.mem3₀ₗ hXX.mem3₁ₗ hXX.mem3₂ₗ hYY.mem3₀ₗ hYY.mem3₁ₗ hYY.mem3₂ₗ| = matrix3x3unsigned₁ )
    (hBᵣ' : |Bᵣ'.submatrix3x3mems hXX.mem3₀ᵣ hXX.mem3₁ᵣ hXX.mem3₂ᵣ hYY.mem3₀ᵣ hYY.mem3₁ᵣ hYY.mem3₂ᵣ| = matrix3x3unsigned₀ ∨
            |Bᵣ'.submatrix3x3mems hXX.mem3₀ᵣ hXX.mem3₁ᵣ hXX.mem3₂ᵣ hYY.mem3₀ᵣ hYY.mem3₁ᵣ hYY.mem3₂ᵣ| = matrix3x3unsigned₁ ) :
    -- respective `x`s and `y`s as members of respective sets
    let ⟨⟨x₀ₗ, x₁ₗ, x₂ₗ⟩, ⟨x₀ᵣ, x₁ᵣ, x₂ᵣ⟩⟩ := hXX.inter3all
    let ⟨⟨y₀ₗ, y₁ₗ, y₂ₗ⟩, ⟨y₀ᵣ, y₁ᵣ, y₂ᵣ⟩⟩ := hYY.inter3all
    -- convert summands to canonical form
    let Bₗ := Bₗ'.toCanonicalSigning x₀ₗ x₁ₗ x₂ₗ y₀ₗ y₁ₗ y₂ₗ
    let Bᵣ := Bᵣ'.toCanonicalSigning x₀ᵣ x₁ᵣ x₂ᵣ y₀ᵣ y₁ᵣ y₂ᵣ
    -- pieces of the bottom left submatrix
    let D₀ᵣ := Bᵣ.submatrix2x2 x₀ᵣ x₁ᵣ y₀ᵣ y₁ᵣ
    let Dₗ := Bₗ.submatrix2x7 x₀ₗ x₁ₗ y₀ y₁ y₂
    let Dᵣ := Bᵣ.submatrix7x2 x₀ x₁ x₂ y₀ᵣ y₁ᵣ
    -- the actual statement
    (Bᵣ.drop1row2cols x₂ y₀ y₁ ◫ (⊞ Dₗ D₀ᵣ (Dᵣ * D₀ᵣ⁻¹ * Dₗ) Dᵣ).submatrix mapX mapY).IsTotallyUnimodular :=
  sorry
```

What Tipped the Scale



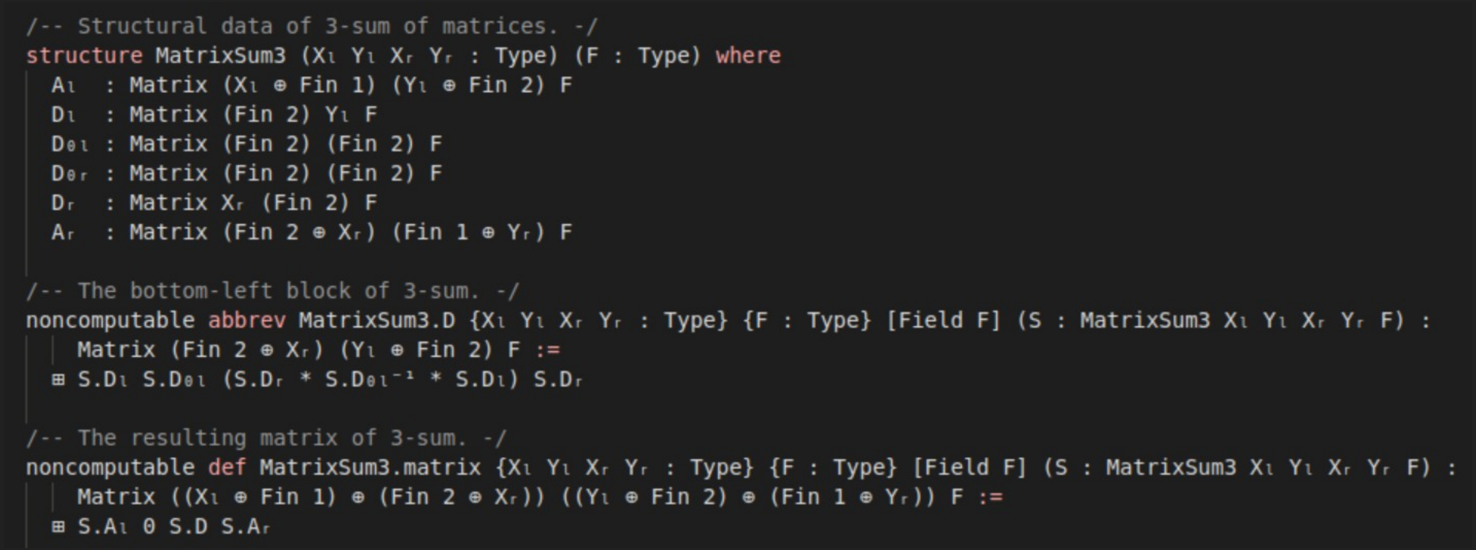
```lean
lemma matrix3sumCanonicalSigning_isSigningOf_matrix3sumComposition {Xₗ Yₗ Xᵣ Yᵣ : Set α} {x₀ x₁ x₂ y₀ y₁ y₂ : α}
    [∀ x, Decidable (x ∈ Xₗ)] [∀ x, Decidable (x ∈ Xᵣ)] [∀ y, Decidable (y ∈ Yₗ)] [∀ y, Decidable (y ∈ Yᵣ)]
    {Bₗ : Matrix Xₗ Yₗ ℚ} {Bᵣ : Matrix Xᵣ Yᵣ ℚ} (hXX : Xₗ ∩ Xᵣ = {x₀, x₁, x₂}) (hYY : Yₗ ∩ Yᵣ = {y₀, y₁, y₂})
    (hBₗ : ∀ i : Xₗ, ∀ j : Yₗ, Bₗ i j ∈ SignType.cast.range) (hBᵣ : ∀ i : Xᵣ, ∀ j : Yᵣ, Bᵣ i j ∈ SignType.cast.range) :
    -- row membership
    let x₀ₗ : Xₗ := ⟨x₀, hXX.mem3₀ₗ⟩
    let x₀ᵣ : Xᵣ := ⟨x₀, hXX.mem3₀ᵣ⟩
    let x₁ₗ : Xₗ := ⟨x₁, hXX.mem3₁ₗ⟩
    let x₁ᵣ : Xᵣ := ⟨x₁, hXX.mem3₁ᵣ⟩
    let x₂ₗ : Xₗ := ⟨x₂, hXX.mem3₂ₗ⟩
    let x₂ᵣ : Xᵣ := ⟨x₂, hXX.mem3₂ᵣ⟩
    -- col membership
    let y₀ₗ : Yₗ := ⟨y₀, hYY.mem3₀ₗ⟩
    let y₀ᵣ : Yᵣ := ⟨y₀, hYY.mem3₀ᵣ⟩
    let y₁ₗ : Yₗ := ⟨y₁, hYY.mem3₁ₗ⟩
    let y₁ᵣ : Yᵣ := ⟨y₁, hYY.mem3₁ᵣ⟩
    let y₂ₗ : Yₗ := ⟨y₂, hYY.mem3₂ₗ⟩
    let y₂ᵣ : Yᵣ := ⟨y₂, hYY.mem3₂ᵣ⟩
    -- extract submatrices but over `Z2`
    let Aₗ := Bₗ.support.Aₗ x₀ₗ x₁ₗ y₂ₗ
    let Dₗ := Bₗ.support.Dₗ x₀ₗ x₁ₗ y₀ₗ y₁ₗ y₂ₗ
    let D₀ := Bₗ.support.D₀ x₀ₗ x₁ₗ y₀ₗ y₁ₗ
    let Dᵣ := Bᵣ.support.Dᵣ x₀ᵣ x₁ᵣ x₂ᵣ y₀ᵣ y₁ᵣ
    let Aᵣ := Bᵣ.support.Aᵣ x₂ᵣ y₀ᵣ y₁ᵣ
    -- the necessary parts of "validity" of the 3-sum
    |Bₗ x₀ₗ y₀ₗ| = 1 →
    |Bₗ x₀ₗ y₂ₗ| = 1 →
    |Bₗ x₂ₗ y₀ₗ| = 1 →
    |Bₗ x₁ₗ y₂ₗ| = 1 →
    |Bₗ x₂ₗ y₁ₗ| = 1 →
    |Bᵣ x₀ᵣ y₀ᵣ| = 1 →
    |Bᵣ x₀ᵣ y₂ᵣ| = 1 →
    |Bᵣ x₂ᵣ y₀ᵣ| = 1 →
    |Bᵣ x₁ᵣ y₂ᵣ| = 1 →
    |Bᵣ x₂ᵣ y₁ᵣ| = 1 →
    -- the actual statement
    (matrix3sumCanonicalSigning Bₗ Bᵣ hXX hYY).IsSigningOf (
      matrix3sumComposition x₀ₗ x₁ₗ x₀ᵣ x₁ᵣ x₂ᵣ y₀ₗ y₁ₗ y₂ₗ y₀ᵣ y₁ᵣ Aₗ Dₗ D₀ Dᵣ Aᵣ
    ) := by
  sorry
```

Inspiration from the Past



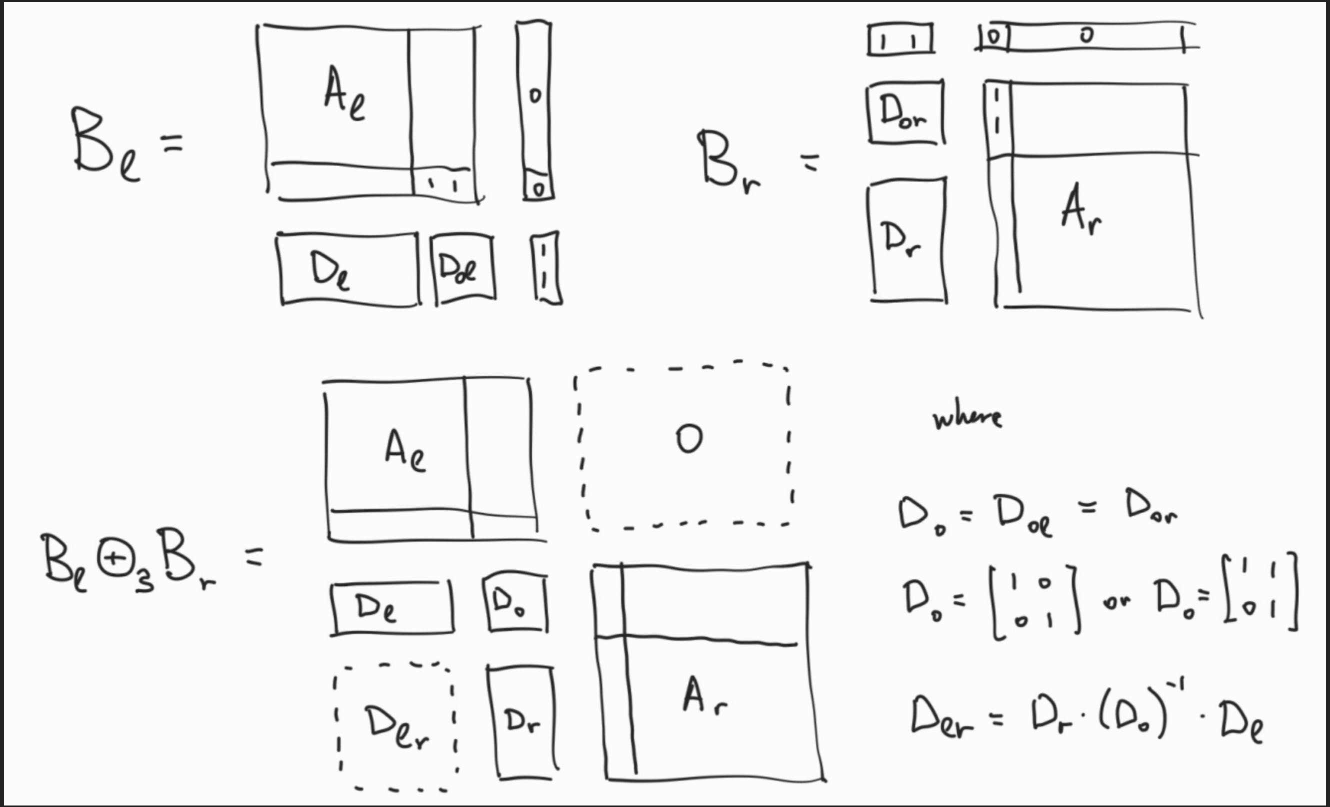
```
/-- Standard matrix representation of a vector matroid. -/
structure StandardRepr (α R : Type) [DecidableEq α] where
  /-- Row indices. -/
  X : Set α
  /-- Col indices. -/
  Y : Set α
  /-- Basis and nonbasis elements are disjoint -/
  hXY : X ⫗ Y
  /-- Standard representation matrix. -/
  B : Matrix X Y R
  /-- The computer can determine whether certain element is a row. -/
  decmemX : ∀ a, Decidable (a ∈ X)
  /-- The computer can determine whether certain element is a col. -/
  decmemY : ∀ a, Decidable (a ∈ Y)
```

Updated 3-Sum of Matrices



```
/-- Structural data of 3-sum of matrices. -/
structure MatrixSum3 (Xₗ Yₗ Xᵣ Yᵣ : Type) (F : Type) where
  Aₗ  : Matrix (Xₗ ⊕ Fin 1) (Yₗ ⊕ Fin 2) F
  Dₗ  : Matrix (Fin 2) Yₗ F
  D₀ₗ : Matrix (Fin 2) (Fin 2) F
  D₀ᵣ : Matrix (Fin 2) (Fin 2) F
  Dᵣ  : Matrix Xᵣ (Fin 2) F
  Aᵣ  : Matrix (Fin 2 ⊕ Xᵣ) (Fin 1 ⊕ Yᵣ) F

/-- The bottom-left block of 3-sum. -/
noncomputable abbrev MatrixSum3.D {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type} [Field F] (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F) :
    Matrix (Fin 2 ⊕ Xᵣ) (Yₗ ⊕ Fin 2) F :=
  ⊞ S.Dₗ S.D₀ₗ (S.Dᵣ * S.D₀ₗ⁻¹ * S.Dₗ) S.Dᵣ

/-- The resulting matrix of 3-sum. -/
noncomputable def MatrixSum3.matrix {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type} [Field F] (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F) :
    Matrix ((Xₗ ⊕ Fin 1) ⊕ (Fin 2 ⊕ Xᵣ)) ((Yₗ ⊕ Fin 2) ⊕ (Fin 1 ⊕ Yᵣ)) F :=
  ⊞ S.Aₗ 0 S.D S.Aᵣ
```

Matrix Structure Redux



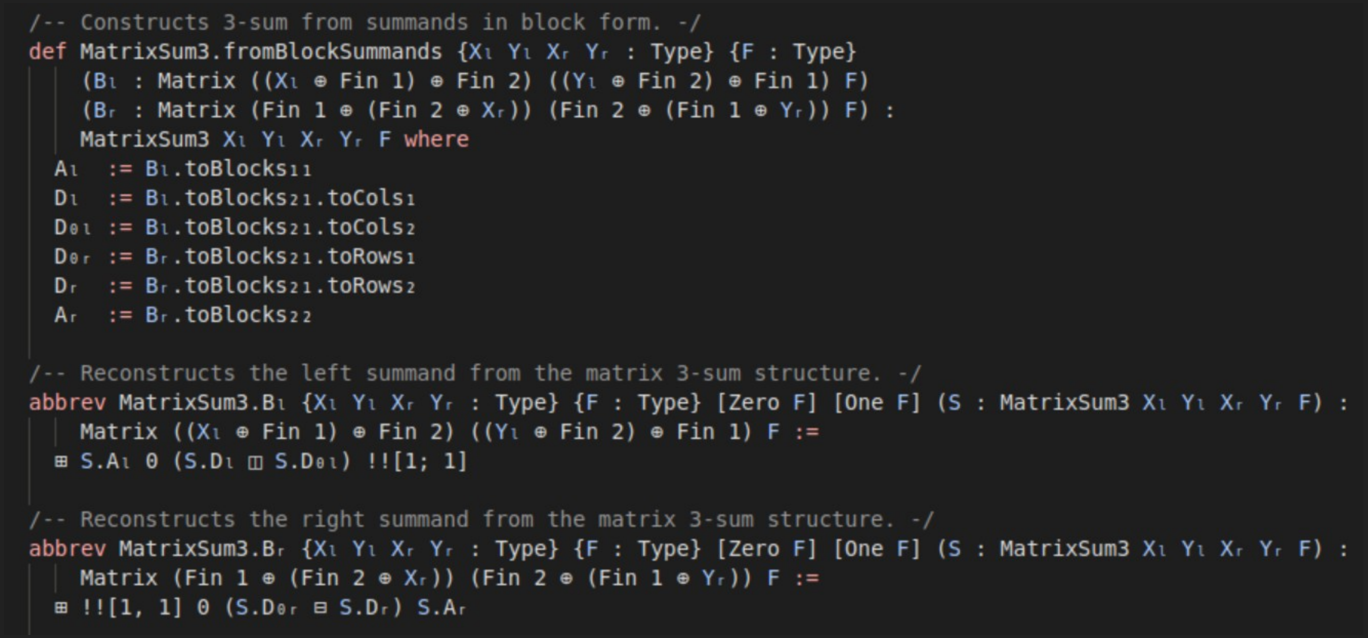
where

$$D_o = D_{o\ell} = D_{or}$$

$$D_o = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ or } D_o = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

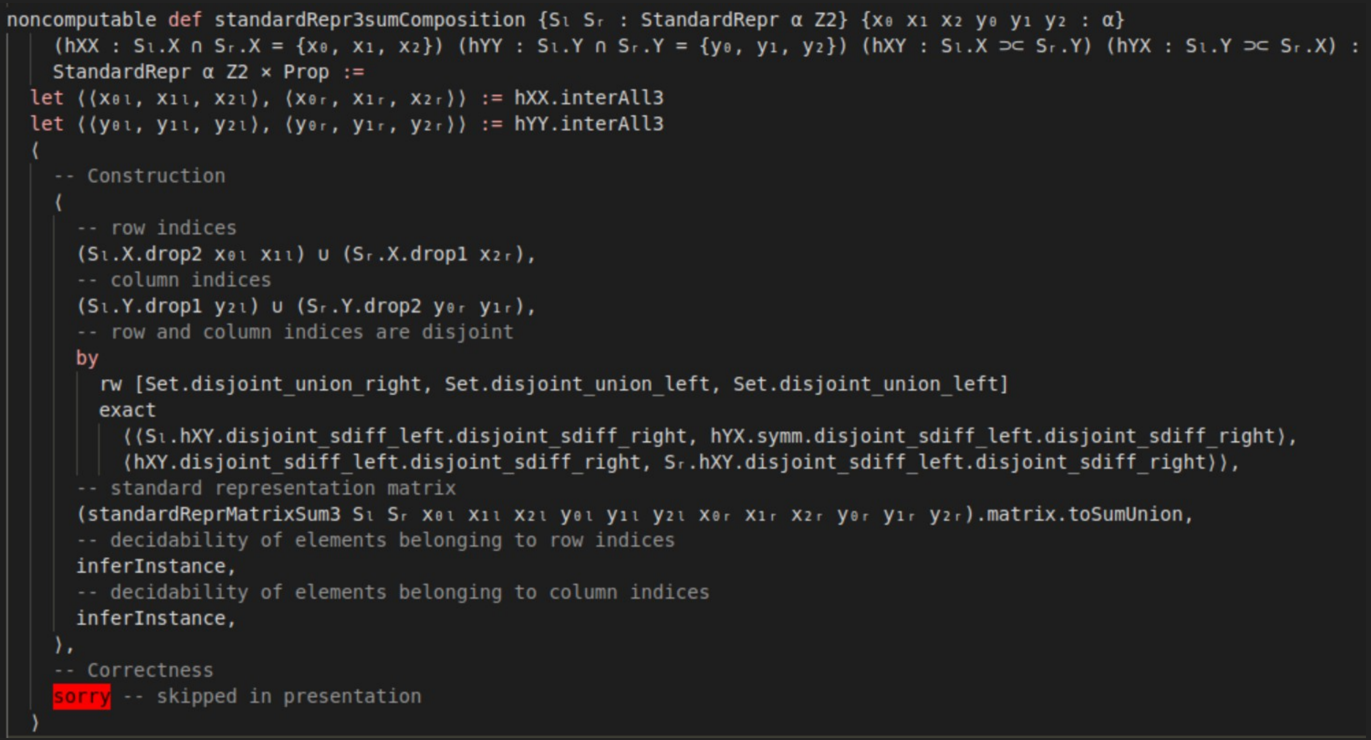
$$D_{\ell r} = D_r \cdot (D_o)^{-1} \cdot D_\ell$$

Summands in Block Form

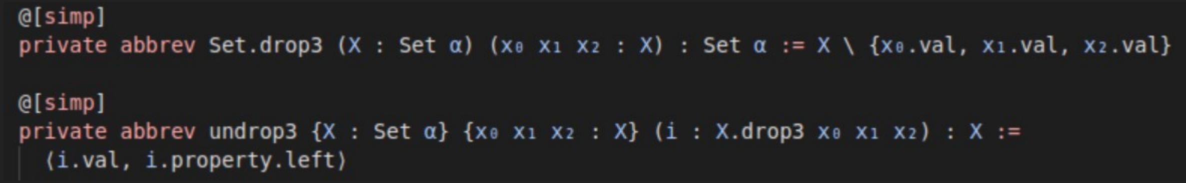


```lean
/-- Constructs 3-sum from summands in block form. -/
def MatrixSum3.fromBlockSummands {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type}
    (Bₗ : Matrix ((Xₗ ⊕ Fin 1) ⊕ Fin 2) ((Yₗ ⊕ Fin 2) ⊕ Fin 1) F)
    (Bᵣ : Matrix (Fin 1 ⊕ (Fin 2 ⊕ Xᵣ)) (Fin 2 ⊕ (Fin 1 ⊕ Yᵣ)) F) :
    MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F where
  Aₗ  := Bₗ.toBlocks₁₁
  Dₗ  := Bₗ.toBlocks₂₁.toCols₁
  D₀ₗ := Bₗ.toBlocks₂₁.toCols₂
  D₀ᵣ := Bᵣ.toBlocks₂₁.toRows₁
  Dᵣ  := Bᵣ.toBlocks₂₁.toRows₂
  Aᵣ  := Bᵣ.toBlocks₂₂

/-- Reconstructs the left summand from the matrix 3-sum structure. -/
abbrev MatrixSum3.Bₗ {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type} [Zero F] [One F] (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F) :
    Matrix ((Xₗ ⊕ Fin 1) ⊕ Fin 2) ((Yₗ ⊕ Fin 2) ⊕ Fin 1) F :=
  ⊞ S.Aₗ 0 (S.Dₗ ◫ S.D₀ₗ) !![1; 1]

/-- Reconstructs the right summand from the matrix 3-sum structure. -/
abbrev MatrixSum3.Bᵣ {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type} [Zero F] [One F] (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F) :
    Matrix (Fin 1 ⊕ (Fin 2 ⊕ Xᵣ)) (Fin 2 ⊕ (Fin 1 ⊕ Yᵣ)) F :=
  ⊞ !![1, 1] 0 (S.D₀ᵣ ⊟ S.Dᵣ) S.Aᵣ
```

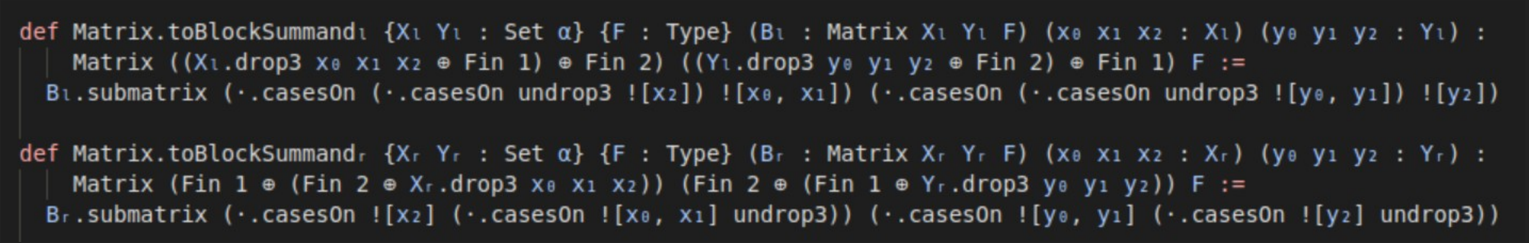
Standard Representations



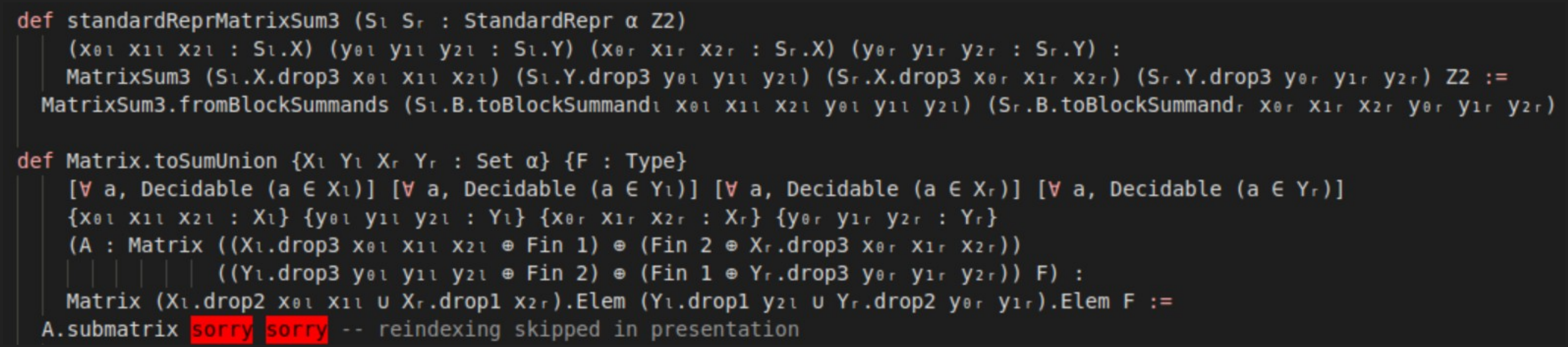
```
noncomputable def standardRepr3sumComposition {Sₗ Sᵣ : StandardRepr α Z2} {x₀ x₁ x₂ y₀ y₁ y₂ : α}
    (hXX : Sₗ.X ∩ Sᵣ.X = {x₀, x₁, x₂}) (hYY : Sₗ.Y ∩ Sᵣ.Y = {y₀, y₁, y₂}) (hXY : Sₗ.X ⫗ Sᵣ.Y) (hYX : Sₗ.Y ⫗ Sᵣ.X) :
    StandardRepr α Z2 × Prop :=
  let ⟨⟨x₀ₗ, x₁ₗ, x₂ₗ⟩, ⟨x₀ᵣ, x₁ᵣ, x₂ᵣ⟩⟩ := hXX.interAll3
  let ⟨⟨y₀ₗ, y₁ₗ, y₂ₗ⟩, ⟨y₀ᵣ, y₁ᵣ, y₂ᵣ⟩⟩ := hYY.interAll3
  ⟨
    -- Construction
    ⟨
      -- row indices
      (Sₗ.X.drop2 x₀ₗ x₁ₗ) ∪ (Sᵣ.X.drop1 x₂ᵣ),
      -- column indices
      (Sₗ.Y.drop1 y₂ₗ) ∪ (Sᵣ.Y.drop2 y₀ᵣ y₁ᵣ),
      -- row and column indices are disjoint
      by
        rw [Set.disjoint_union_right, Set.disjoint_union_left, Set.disjoint_union_left]
        exact
          ⟨⟨Sₗ.hXY.disjoint_sdiff_left.disjoint_sdiff_right, hYX.symm.disjoint_sdiff_left.disjoint_sdiff_right⟩,
          ⟨hXY.disjoint_sdiff_left.disjoint_sdiff_right, Sᵣ.hXY.disjoint_sdiff_left.disjoint_sdiff_right⟩⟩,
      -- standard representation matrix
      (standardReprMatrixSum3 Sₗ Sᵣ x₀ₗ x₁ₗ x₂ₗ y₀ₗ y₁ₗ y₂ₗ x₀ᵣ x₁ᵣ x₂ᵣ y₀ᵣ y₁ᵣ y₂ᵣ).matrix.toSumUnion,
      -- decidability of elements belonging to row indices
      inferInstance,
      -- decidability of elements belonging to column indices
      inferInstance,
    ⟩,
    -- Correctness
    sorry -- skipped in presentation
  ⟩
```



```
@[simp]
private abbrev Set.drop3 (X : Set α) (x₀ x₁ x₂ : X) : Set α := X \ {x₀.val, x₁.val, x₂.val}

@[simp]
private abbrev undrop3 {X : Set α} {x₀ x₁ x₂ : X} (i : X.drop3 x₀ x₁ x₂) : X :=
  ⟨i.val, i.property.left⟩
```

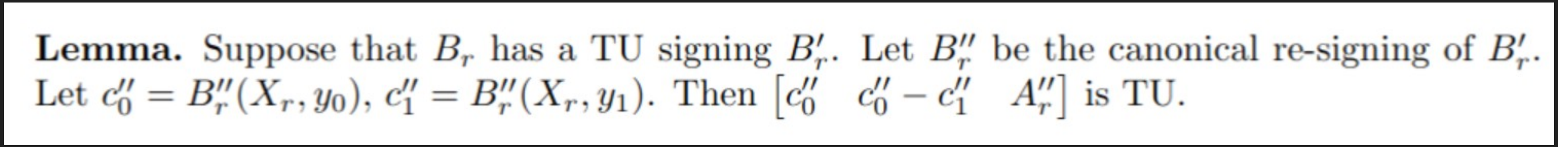


```
def Matrix.toBlockSummandₗ {Xₗ Yₗ : Set α} {F : Type} (Bₗ : Matrix Xₗ Yₗ F) (x₀ x₁ x₂ : Xₗ) (y₀ y₁ y₂ : Yₗ) :
    Matrix ((Xₗ.drop3 x₀ x₁ x₂ ⊕ Fin 1) ⊕ Fin 2) ((Yₗ.drop3 y₀ y₁ y₂ ⊕ Fin 2) ⊕ Fin 1) F :=
  Bₗ.submatrix (·.casesOn (·.casesOn undrop3 ![x₂]) ![x₀, x₁]) (·.casesOn (·.casesOn undrop3 ![y₀, y₁]) ![y₂])

def Matrix.toBlockSummandᵣ {Xᵣ Yᵣ : Set α} {F : Type} (Bᵣ : Matrix Xᵣ Yᵣ F) (x₀ x₁ x₂ : Xᵣ) (y₀ y₁ y₂ : Yᵣ) :
    Matrix (Fin 1 ⊕ (Fin 2 ⊕ Xᵣ.drop3 x₀ x₁ x₂)) (Fin 2 ⊕ (Fin 1 ⊕ Yᵣ.drop3 y₀ y₁ y₂)) F :=
  Bᵣ.submatrix (·.casesOn ![x₂] (·.casesOn ![x₀, x₁] undrop3)) (·.casesOn ![y₀, y₁] (·.casesOn ![y₂] undrop3))
```



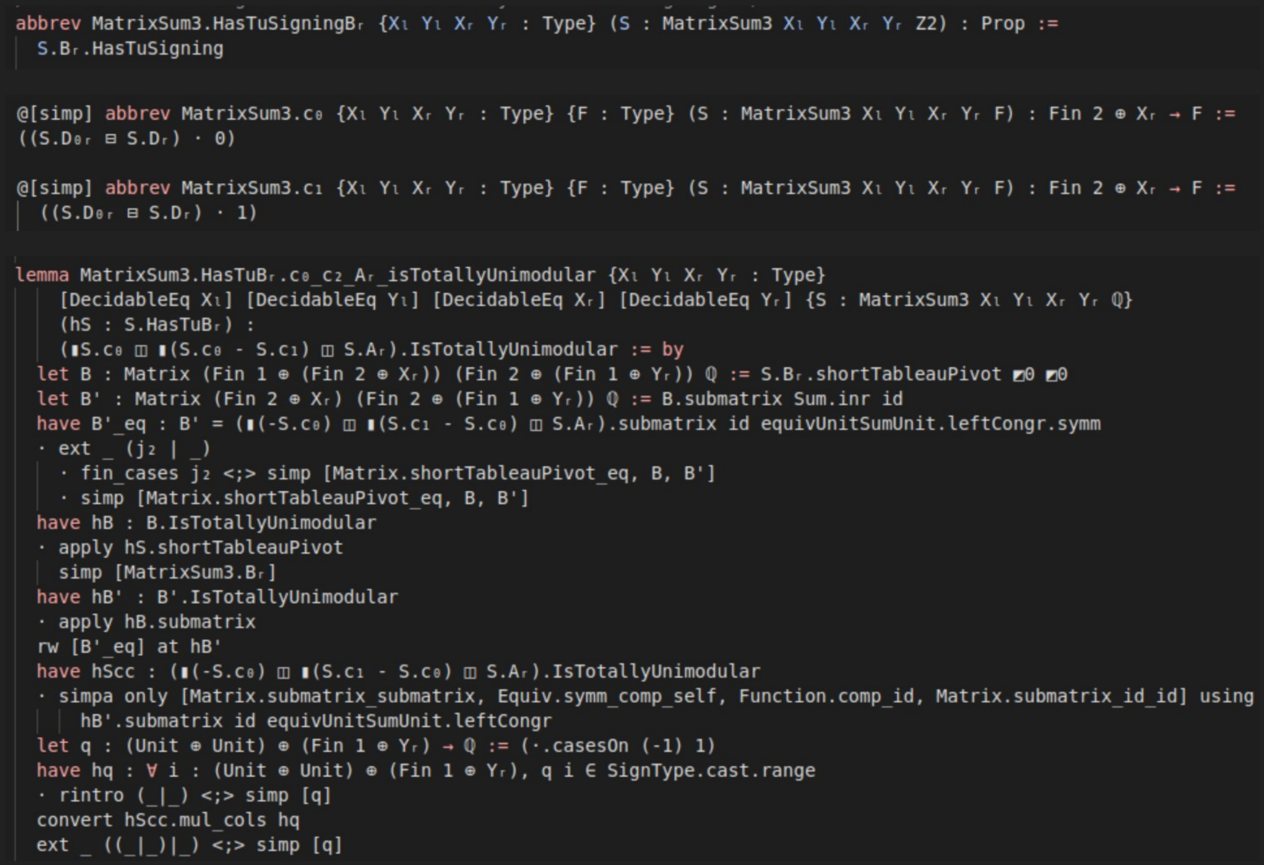
```
def standardReprMatrixSum3 (Sₗ Sᵣ : StandardRepr α Z2)
    (x₀ₗ x₁ₗ x₂ₗ : Sₗ.X) (y₀ₗ y₁ₗ y₂ₗ : Sₗ.Y) (x₀ᵣ x₁ᵣ x₂ᵣ : Sᵣ.X) (y₀ᵣ y₁ᵣ y₂ᵣ : Sᵣ.Y) :
    MatrixSum3 (Sₗ.X.drop3 x₀ₗ x₁ₗ x₂ₗ) (Sₗ.Y.drop3 y₀ₗ y₁ₗ y₂ₗ) (Sᵣ.X.drop3 x₀ᵣ x₁ᵣ x₂ᵣ) (Sᵣ.Y.drop3 y₀ᵣ y₁ᵣ y₂ᵣ) Z2 :=
  MatrixSum3.fromBlockSummands (Sₗ.B.toBlockSummandₗ x₀ₗ x₁ₗ x₂ₗ y₀ₗ y₁ₗ y₂ₗ) (Sᵣ.B.toBlockSummandᵣ x₀ᵣ x₁ᵣ x₂ᵣ y₀ᵣ y₁ᵣ y₂ᵣ)

def Matrix.toSumUnion {Xₗ Yₗ Xᵣ Yᵣ : Set α} {F : Type}
    [∀ a, Decidable (a ∈ Xₗ)] [∀ a, Decidable (a ∈ Yₗ)] [∀ a, Decidable (a ∈ Xᵣ)] [∀ a, Decidable (a ∈ Yᵣ)]
    {x₀ₗ x₁ₗ x₂ₗ : Xₗ} {y₀ₗ y₁ₗ y₂ₗ : Yₗ} {x₀ᵣ x₁ᵣ x₂ᵣ : Xᵣ} {y₀ᵣ y₁ᵣ y₂ᵣ : Yᵣ}
    (A : Matrix ((Xₗ.drop3 x₀ₗ x₁ₗ x₂ₗ ⊕ Fin 1) ⊕ (Fin 2 ⊕ Xᵣ.drop3 x₀ᵣ x₁ᵣ x₂ᵣ))
                ((Yₗ.drop3 y₀ₗ y₁ₗ y₂ₗ ⊕ Fin 2) ⊕ (Fin 1 ⊕ Yᵣ.drop3 y₀ᵣ y₁ᵣ y₂ᵣ)) F) :
    Matrix (Xₗ.drop2 x₀ₗ x₁ₗ ∪ Xᵣ.drop1 x₂ᵣ).Elem (Yₗ.drop1 y₂ₗ ∪ Yᵣ.drop2 y₀ᵣ y₁ᵣ).Elem F :=
  A.submatrix sorry sorry -- reindexing skipped in presentation
```

Benefits of Refactoring



Lemma. Suppose that B_r has a TU signing B_r'. Let B_r'' be the canonical re-signing of B_r'. Let $c_0'' = B_r''(X_r, y_0)$, $c_1'' = B_r''(X_r, y_1)$. Then $\begin{bmatrix} c_0'' & c_0'' - c_1'' & A_r'' \end{bmatrix}$ is TU.

```
lemma Matrix.IsTotallyUnimodular.signing_expansion₀ {X Y : Set α} {Q : Matrix X Y ℚ} (hQ : Q.IsTotallyUnimodular)
    {x₂ y₀ y₁ : α} (hx₂ : x₂ ∈ X) (hy₀ : y₀ ∈ Y) (hy₁ : y₁ ∈ Y) (hyy : y₀ ≠ y₁)
    (hQy₀ : Q ⟨x₂, hx₂⟩ ⟨y₀, hy₀⟩ = 1)
    (hQy₁ : Q ⟨x₂, hx₂⟩ ⟨y₁, hy₁⟩ = 1)
    (hQy : ∀ y : Y, y.val ≠ y₀ ∧ y.val ≠ y₁ → Q ⟨x₂, hx₂⟩ y = 0) :
    let c₀ := Q._col ⟨y₀, hy₀⟩
    let c₁ := Q._col ⟨y₁, hy₁⟩
    let Q' := Q.drop1row2cols x₂ y₀ y₁
    (Q' ◫ ▮c₀ ◫ ▮(c₀ - c₁)).IsTotallyUnimodular := by
  intro c₀ c₁ Q'
  let B : Matrix X Y ℚ := Q.shortTableauPivot ⟨x₂, hx₂⟩ ⟨y₀, hy₀⟩
  let B' : Matrix (X \ {x₂}).Elem Y ℚ := B.submatrix Set.diff_subset.elem id
  let e : ((Y \ {y₀, y₁}).Elem ⊕ Unit) ⊕ Unit ≃ Y := ⟨
    (·.casesOn (·.casesOn Set.diff_subset.elem ↓⟨y₀, hy₀⟩) ↓⟨y₁, hy₁⟩),
    fun ⟨y, hy⟩ => if hy₀ : y = y₀ then ◩◪() else if hy₁ : y = y₁ then ◪() else ◩◩⟨y, by simp [*]⟩,
    ↓(by aesop),
    ↓(by aesop)⟩
  have B'_eq : B' = (Q' ◫ ▮(-c₀) ◫ ▮(c₁ - c₀)).submatrix id e.symm
  · ext ⟨i, hi⟩ ⟨j, hj⟩
    have := hi.right
    if j = y₀ then
      simp_all [Matrix.shortTableauPivot_eq, e, B, B', c₀]
    else if j = y₁ then
      simp_all [Matrix.shortTableauPivot_eq, e, B, B', c₀, c₁]
    else
      simp_all [Matrix.shortTableauPivot_eq, e, B, B', Q']
  have hB : B.IsTotallyUnimodular
  · apply hQ.shortTableauPivot
    rw [hQy₀]
    exact Rat.zero_ne_one.symm
  have hB' : B'.IsTotallyUnimodular
  · apply hB.submatrix
  rw [B'_eq] at hB'
  have hQcc : (Q' ◫ ▮(-c₀) ◫ ▮(c₁ - c₀)).IsTotallyUnimodular
  · simpa using hB'.submatrix id e
  let q : ((Y \ {y₀, y₁}).Elem ⊕ Unit) ⊕ Unit → ℚ := (·.casesOn (·.casesOn 1 (-1)) (-1))
  have hq : ∀ i : ((Y \ {y₀, y₁}).Elem ⊕ Unit) ⊕ Unit, q i ∈ SignType.cast.range
  · rintro ((_|_)|_) <;> simp [q]
  convert hQcc.mul_cols hq
  ext _ ((_|_)|_) <;> simp [q]
```



```
abbrev MatrixSum3.HasTuSigningBᵣ {Xₗ Yₗ Xᵣ Yᵣ : Type} (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ Z2) : Prop :=
  S.Bᵣ.HasTuSigning

@[simp] abbrev MatrixSum3.c₀ {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type} (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F) : Fin 2 ⊕ Xᵣ → F :=
((S.D₀ᵣ ⊟ S.Dᵣ) · 0)

@[simp] abbrev MatrixSum3.c₁ {Xₗ Yₗ Xᵣ Yᵣ : Type} {F : Type} (S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ F) : Fin 2 ⊕ Xᵣ → F :=
  ((S.D₀ᵣ ⊟ S.Dᵣ) · 1)

lemma MatrixSum3.HasTuBᵣ.c₀_c₂_Aᵣ_isTotallyUnimodular {Xₗ Yₗ Xᵣ Yᵣ : Type}
    [DecidableEq Xₗ] [DecidableEq Yₗ] [DecidableEq Xᵣ] [DecidableEq Yᵣ] {S : MatrixSum3 Xₗ Yₗ Xᵣ Yᵣ ℚ}
    (hS : S.HasTuBᵣ) :
    (▮S.c₀ ◫ ▮(S.c₀ - S.c₁) ◫ S.Aᵣ).IsTotallyUnimodular := by
  let B : Matrix (Fin 1 ⊕ (Fin 2 ⊕ Xᵣ)) (Fin 2 ⊕ (Fin 1 ⊕ Yᵣ)) ℚ := S.Bᵣ.shortTableauPivot ◩0 ◩0
  let B' : Matrix (Fin 2 ⊕ Xᵣ) (Fin 2 ⊕ (Fin 1 ⊕ Yᵣ)) ℚ := B.submatrix Sum.inr id
  have B'_eq : B' = (▮(-S.c₀) ◫ ▮(S.c₁ - S.c₀) ◫ S.Aᵣ).submatrix id equivUnitSumUnit.leftCongr.symm
  · ext _ (j₂ | _)
    · fin_cases j₂ <;> simp [Matrix.shortTableauPivot_eq, B, B']
    · simp [Matrix.shortTableauPivot_eq, B, B']
  have hB : B.IsTotallyUnimodular
  · apply hS.shortTableauPivot
    simp [MatrixSum3.Bᵣ]
  have hB' : B'.IsTotallyUnimodular
  · apply hB.submatrix
  rw [B'_eq] at hB'
  have hScc : (▮(-S.c₀) ◫ ▮(S.c₁ - S.c₀) ◫ S.Aᵣ).IsTotallyUnimodular
  · simpa only [Matrix.submatrix_submatrix, Equiv.symm_comp_self, Function.comp_id, Matrix.submatrix_id_id] using
      hB'.submatrix id equivUnitSumUnit.leftCongr
  let q : (Unit ⊕ Unit) ⊕ (Fin 1 ⊕ Yᵣ) → ℚ := (·.casesOn (-1) 1)
  have hq : ∀ i : (Unit ⊕ Unit) ⊕ (Fin 1 ⊕ Yᵣ), q i ∈ SignType.cast.range
  · rintro (_|_) <;> simp [q]
  convert hScc.mul_cols hq
  ext _ ((_|_)|_) <;> simp [q]
```