In[∘]:=
```
SetDirectory[NotebookDirectory[]];
<< OperatorGB.m
```

```
Package OperatorGB version 1.2.1
Copyright 2019, Institute for Algebra, JKU
by Clemens Hofstadler, clemens.hofstadler@jku.at
```

Notebook file with all the automated proofs for the paper
“Algebraic proof methods for identities of matrices and operators: improvements of Hartwig’s triple reverse order law”

Theorem 2.1 (Hartwig original version)

Let A, B, C with Moore-Penrose inverses $A^{\dagger}$, $B^{\dagger}$, $C^{\dagger}$ be such that $M = A\,B\,C$ is defined. Let $P = A^{\dagger}\,A\,B\,C\,C^{\dagger}$ and $Q = C\,C^{\dagger}\,B^{\dagger}\,A^{\dagger}\,A$. Then, the following are equivalent:

i) $M^{\dagger} = C^{\dagger}\,B^{\dagger}\,A^{\dagger}$;

ii) $Q \in P\{1, 2\}$ and both of $A^{*}\,A\,P\,Q$ and $Q\,P\,C\,C^{*}$ are Hermitian;

iii) $Q \in P\{1, 2\}$ and both of $A^{*}\,A\,P\,Q$ and $Q\,P\,C\,C^{*}$ are EP;

iv) $Q \in P\{1\}$, $R(A^{*}\,A\,P) = R(Q^{*})$, and $R(C\,C^{*}\,P^{*}) = R(Q)$;

v) $P\,Q = (P\,Q)^{2}$, $R(A^{*}\,A\,P) = R(Q^{*})$, and $R(C\,C^{*}\,P^{*}) = R(Q)$;

In[∘]:=
```
Q = {{a, 3, 4}, {adj[a], 4, 3}, {aa, 4, 3}, {adj[aa], 3, 4},
     {b, 2, 3}, {adj[b], 3, 2}, {bb, 3, 2}, {adj[bb], 2, 3},
     {c, 1, 2}, {adj[c], 2, 1}, {cc, 2, 1}, {adj[cc], 1, 2},
     {mm, 4, 1}, {adj[mm], 1, 4},
     {u, 3, 3}, {adj[u], 3, 3}, {v, 2, 2}, {adj[v], 2, 2},
     {w, 2, 2}, {adj[w], 2, 2}, {s, 3, 3}, {adj[s], 3, 3}};
```

```
In[•]:= PinvA = Pinv[a, aa];
PinvB = Pinv[b, bb];
PinvC = Pinv[c, cc];
p = aa ** a ** b ** c ** cc;
q = c ** cc ** bb ** aa ** a;
m = a ** b ** c;
PinvM = Pinv[m, mm];
cond1 = {mm - cc ** bb ** aa};
cond2 = {p ** q ** p - p, q ** p ** q - q, adj[a] ** a ** p ** q - adj[adj[a] ** a ** p ** q],
   q ** p ** c ** adj[c] - adj[q ** p ** c ** adj[c]]};
cond3 = {p ** q ** p - p, q ** p ** q - q,
   adj[a] ** a ** p ** q ** u - adj[adj[a] ** a ** p ** q],
   adj[a] ** a ** p ** q - adj[adj[a] ** a ** p ** q] ** s,
   q ** p ** c ** adj[c] ** v - adj[q ** p ** c ** adj[c]],
   q ** p ** c ** adj[c] - adj[q ** p ** c ** adj[c]] ** w};
cond4 = {p ** q ** p - p, adj[q] - adj[a] ** a ** p ** v, c ** adj[c] ** adj[p] - q ** u,
   adj[q] ** w - adj[a] ** a ** p, c ** adj[c] ** adj[p] ** s - q};
cond5 = {p ** q ** p ** q - p ** q, adj[q] - adj[a] ** a ** p ** v, c ** adj[c] ** adj[p] -
    q ** u, adj[q] ** w - adj[a] ** a ** p, c ** adj[c] ** adj[p] ** s - q};

u1 = b ** c ** adj[c] ** adj[b] ** adj[a] ** adj[aa];
u2 = adj[bb] ** adj[cc] ** cc ** bb ** aa ** a;
v1 = adj[b] ** adj[a] ** a ** b ** c ** cc;
v2 = bb ** aa ** adj[aa] ** adj[bb] ** adj[cc] ** adj[c];

cond4Goal = {p ** q ** p - p, adj[a] ** a ** p - adj[q] ** v1,
   adj[q] - adj[a] ** a ** p ** v2,
   c ** adj[c] ** adj[p] - q ** u1,
   q - c ** adj[c] ** adj[p] ** u2};
cond5Goal = {p ** q ** p ** q - p ** q, adj[a] ** a ** p - adj[q] ** v1,
   adj[q] - adj[a] ** a ** p ** v2,
   c ** adj[c] ** adj[p] - q ** u1,
   q - c ** adj[c] ** adj[p] ** u2};
```

(i) <=> (ii)

(i) => (ii)

In[∘]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond1] // AddAdj;
certificate = Certify[assumptions, cond2, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond2
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(ii) => (i)

In[∘]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond2] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(ii) <=> (iii)

(ii) => (iii)

In[∘]:=
```
(* This holds trivially *)
```

(iii) => (ii)

In[∘]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond3] // AddAdj;
certificate = Certify[assumptions, cond2, Q, MaxDeg → 20, MultiLex → True];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond2
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(i) <=> (iv)

(i) => (iv)

In[∘]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond1] // AddAdj;
certificate = Certify[assumptions, cond4Goal, Q, MaxDeg → 15, MultiLex → True];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond4Goal
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(iv) => (i)

In[∘]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond4] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MaxDeg → 15, MultiLex → True];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(i) <=> (v)

(i) => (v)

In[ ]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond1] // AddAdj;
certificate = Certify[assumptions, cond5Goal, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond5Goal
IntegerCoeffQ[certificate]
```

Out[ ]= True

Out[ ]= True

(v) => (i) (&& (ii) - this speeds up the computation due to a different monomial ordering)

In[ ]:=
```
assumptions = Join[PinvA, PinvB, PinvC, PinvM, cond5] // AddAdj;
certificate =
  Certify[assumptions, Join[cond1, cond2], Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[cond1, cond2]
IntegerCoeffQ[certificate]
```

Out[ ]= True

Out[ ]= True

Theorem 2.2

To prove Theorem 2.2, the computations of Theorem 2.1 can be adopted.

Theorem 2.3

Let A, B, C with Moore-Penrose inverses $A^\dagger$, $C^\dagger$ and let $M = A\,B\,C$. Furthermore, let $\tilde{B}$ be such that $Q = C\,C^\dagger\,\tilde{B}\,A^\dagger\,A$ exists and let $P = A^\dagger\,A\,B\,C\,C^\dagger$. Then, the following are equivalent:

i) M is Moore-Penrose invertible and $M^\dagger = C^\dagger\,\tilde{B}\,A^\dagger$;

iv) Q ∈ P{1}, $R(A^*\,A\,P) \supseteq R(Q^*)$, and $R(C\,C^*\,P^*) \subseteq R(Q)$;

v) M is right *-cancellable, $P\,Q = (P\,Q)^2$, $R(A^*\,A\,P) \supseteq R(Q^*)$, and $R(C\,C^*\,P^*) \subseteq R(Q)$;

vi) Q ∈ P{2}, $R(A^*\,A\,P) \supseteq R(Q^*)$, and $R(C\,C^*\,P^*) \subseteq R(Q)$;

```
In[•]:= Q = {{a, 3, 4}, {adj[a], 4, 3}, {aa, 4, 3}, {adj[aa], 3, 4},
          {b, 2, 3}, {adj[b], 3, 2}, {bb, 3, 2}, {adj[bb], 2, 3},
          {c, 1, 2}, {adj[c], 2, 1}, {cc, 2, 1}, {adj[cc], 1, 2},
          {mm, 4, 1}, {adj[mm], 1, 4}, {u, 3, 3}, {adj[u], 3, 3},
      {v, 2, 2}, {adj[v], 2, 2}, {z, 4, 4}, {adj[z], 4, 4}};
```

```
In[•]:= PinvA = Pinv[a, aa];
PinvC = Pinv[c, cc];
p = aa ** a ** b ** c ** cc;
q = c ** cc ** bb ** aa ** a;
m = a ** b ** c;
rangeInclusions = {adj[q] - adj[a] ** a ** p ** v, c ** adj[c] ** adj[p] - q ** u};
cond1 = Pinv[m, cc ** bb ** aa];
cond4 = {p ** q ** p - p};
cond5 = {p ** q ** p ** q - p ** q};
cond6 = {q ** p ** q - q};
u1 = b ** c ** adj[c] ** adj[b] ** adj[a] ** adj[aa];
v2 = bb ** aa ** adj[aa] ** adj[bb] ** adj[cc] ** adj[c];
rangeInclusionsGoal =
  {adj[q] - adj[a] ** a ** p ** v2, c ** adj[c] ** adj[p] - q ** u1};
```

(i) <=> (iv)

(i) => (iv)

```
In[•]:= assumptions = Join[PinvA, PinvC, cond1] // AddAdj;
certificate = Certify[assumptions,
   Join[cond4, rangeInclusionsGoal], Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[cond4, rangeInclusionsGoal]
IntegerCoeffQ[certificate]

Out[•]= True

Out[•]= True
```

(iv) => (i)

```
In[•]:= assumptions = Join[PinvA, PinvC, rangeInclusions, cond4] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(i) <=> (v)

(v) => (i)

In[∘]:=
```
(* First, we show that m̃ = cc**bb**aa is an inner inverse of m *)
(* To this end, we show that our assumptions imply that mm*- mm̃mm* = 0
  holds and then apply the *-cancellability of m *)
assumptions = Join[PinvA, PinvC, rangeInclusions, cond5] // AddAdj;
claim = {m ** adj[m] - m ** cc ** bb ** aa ** m ** adj[m]};
certificate1 = Certify[assumptions, claim, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate1 === claim
IntegerCoeffQ[certificate1]
```

Out[∘]= True

Out[∘]= True

In[∘]:=
```
(* Now, we add the information that m̃
 is an inner inverse of m to our assumptions *)
(* Then, the rest runs through automatically *)
assumptionsNew = Append[assumptions, m - m ** cc ** bb ** aa ** m] // AddAdj;
certificate = Certify[assumptionsNew, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[•]= True

Out[•]= True

(i) => (v)

In[•]:=
```
(* Here, we have to add the information about *
 -cancellability to the assumptions and the claimed properties *)
assumptionCancel = {z ** m ** adj[m]};
claimCancel = {z ** m};
assumptions = Join[PinvA, PinvC, cond1, assumptionCancel] // AddAdj;
certificate =
  Certify[assumptions, Join[claimCancel, cond5, rangeInclusionsGoal],
   Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[claimCancel, cond5, rangeInclusionsGoal]
IntegerCoeffQ[certificate]
```

Out[•]= True

Out[•]= True

(i) <=> (vi)

(i) => (vi)

In[•]:=
```
assumptions = Join[PinvA, PinvC, cond1] // AddAdj;
certificate = Certify[assumptions,
   Join[cond6, rangeInclusionsGoal], Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[cond6, rangeInclusionsGoal]
IntegerCoeffQ[certificate]
```

Out[•]= True

Out[•]= True

(vi) => (i)

In[∘]:=
```
assumptions = Join[PinvA, PinvC, rangeInclusions, cond6] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

Theorem 2.4

Let A, B, C with Moore-Penrose inverses $A^\dagger$, $C^\dagger$ and let $M = A\,B\,C$. Furthermore, let $\tilde{B}$ be such that $Q = C\,C^\dagger\,\tilde{B}\,A^\dagger\,A$ exists and let $P = A^\dagger\,A\,B\,C\,C^\dagger$. Then, the following are equivalent:

i) M is Moore-Penrose invertible and $M^\dagger = C^\dagger\,\tilde{B}\,A^\dagger$;

iv) Q ∈ P{1}, $R(A^*\,A\,P) \subseteq R(Q^*)$, and $R(C\,C^*\,P^*) \supseteq R(Q)$;

v) $C^\dagger\,\tilde{B}\,A^\dagger$ is left *-cancellable, $P\,Q = (P\,Q)^2$, $R(A^*\,A\,P) \subseteq R(Q^*)$, and $R(C\,C^*\,P^*) \supseteq R(Q)$;

vi) Q ∈ P{2}, $R(A^*\,A\,P) \subseteq R(Q^*)$, and $R(C\,C^*\,P^*) \supseteq R(Q)$;

In[∘]:=
```
Q = {{a, 3, 4}, {adj[a], 4, 3}, {aa, 4, 3}, {adj[aa], 3, 4},
       {b, 2, 3}, {adj[b], 3, 2}, {bb, 3, 2}, {adj[bb], 2, 3},
       {c, 1, 2}, {adj[c], 2, 1}, {cc, 2, 1}, {adj[cc], 1, 2},
       {u, 3, 3}, {adj[u], 3, 3},
    {v, 2, 2}, {adj[v], 2, 2}, {z, 4, 4}, {adj[z], 4, 4}};
```

In[∘]:=
```
PinvA = Pinv[a, aa];
PinvC = Pinv[c, cc];
p = aa ** a ** b ** c ** cc;
q = c ** cc ** bb ** aa ** a;
m = a ** b ** c;
rangeInclusions = {adj[q] ** v - adj[a] ** a ** p, c ** adj[c] ** adj[p] ** u - q};
cond1 = Pinv[m, cc ** bb ** aa];
cond4 = {p ** q ** p - p};
cond5 = {p ** q ** p ** q - p ** q};
cond6 = {q ** p ** q - q};
```

In[∘]:=
```
u₂ = adj[bb] ** adj[cc] ** cc ** bb ** aa ** a;
v₁ = adj[b] ** adj[a] ** a ** b ** c ** cc;
rangeInclusionsGoal = {
   adj[a] ** a ** p - adj[q] ** v₁,
   q - c ** adj[c] ** adj[p] ** u₂};
```

(i) <=> (iv)

(i) => (iv)

In[∘]:=
```
assumptions = Join[PinvA, PinvC, cond1] // AddAdj;
certificate = Certify[assumptions,
   Join[cond4, rangeInclusionsGoal], Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[cond4, rangeInclusionsGoal]
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(iv) => (i)

In[∘]:=
```
assumptions = Join[PinvA, PinvC, rangeInclusions, cond4] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(i) <=> (v)

(v) => (i)

In[∘]:=
```
(* First, we show that m̃ = cc**bb**aa is an outer inverse of m *)
(* To this end, we show that our assumptions imply that m̃*m̃ - m̃*m̃mm̃ = 0
  holds and then apply the *-cancellability of m̃ *)
assumptions = Join[PinvA, PinvC, rangeInclusions, cond5] // AddAdj;
claim = {adj[cc ** bb ** aa] ** cc ** bb ** aa -
    adj[cc ** bb ** aa] ** cc ** bb ** aa ** m ** cc ** bb ** aa};
certificate1 = Certify[assumptions, claim, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate1 === claim
IntegerCoeffQ[certificate1]
```

Out[∘]= True

Out[∘]= True

In[∘]:=
```
(* Now, we add the information that m̃
 is an outer inverse of m to our assumptions *)
(* Then, the rest runs through automatically *)
assumptionsNew =
  Append[assumptions, cc ** bb ** aa - cc ** bb ** aa ** m ** cc ** bb ** aa] // AddAdj;
certificate = Certify[assumptionsNew, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(i) => (v)

In[•]:=
```
(* Here, we have to add the information about *
 -cancellability to the assumptions and the claimed properties *)
assumptionCancel = {adj[cc ** bb ** aa] ** cc ** bb ** aa ** z};
claimCancel = {cc ** bb ** aa ** z};
assumptions = Join[PinvA, PinvC, cond1, assumptionCancel] // AddAdj;
certificate =
  Certify[assumptions, Join[claimCancel, cond5, rangeInclusionsGoal],
   Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[claimCancel, cond5, rangeInclusionsGoal]
IntegerCoeffQ[certificate]
```

Out[•]= True

Out[•]= True

(i) <=> (vi)

(i) => (vi)

In[•]:=
```
assumptions = Join[PinvA, PinvC, cond1] // AddAdj;
certificate = Certify[assumptions,
   Join[cond6, rangeInclusionsGoal], Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[cond6, rangeInclusionsGoal]
IntegerCoeffQ[certificate]
```

Out[•]= True

Out[•]= True

(vi) => (i)

In[•]:=
```
assumptions = Join[PinvA, PinvC, rangeInclusions, cond6] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

Theorem 2.6

Let A, B, C with Moore-Penrose inverses $A^\dagger$, $C^\dagger$ and let $M = A\,B\,C$. Furthermore, let $\tilde{B}$ be such that $Q = C\,C^\dagger\,\tilde{B}\,A^\dagger\,A$ exists and let $P = A^\dagger\,A\,B\,C\,C^\dagger$. Then, the following are equivalent:
i) M is Moore-Penrose invertible and $M^\dagger = C^\dagger\,\tilde{B}\,A^\dagger$;
iv) Q ∊ P{1}, $R(A^*\,A\,P) \supseteq R(Q^*)$, and $R(C\,C^*\,P^*) \supseteq R(Q)$;
vi) Q ∊ P{2}, $R(A^*\,A\,P) \subseteq R(Q^*)$, and $R(C\,C^*\,P^*) \subseteq R(Q)$;

```
In[∘]:= Q = {{a, 3, 4}, {adj[a], 4, 3}, {aa, 4, 3}, {adj[aa], 3, 4},
          {b, 2, 3}, {adj[b], 3, 2}, {bb, 3, 2}, {adj[bb], 2, 3},
          {c, 1, 2}, {adj[c], 2, 1}, {cc, 2, 1}, {adj[cc], 1, 2},
          {u, 3, 3}, {adj[u], 3, 3}, {v, 2, 2}, {adj[v], 2, 2}};
```

```
In[∘]:= PinvA = Pinv[a, aa];
PinvC = Pinv[c, cc];
p = aa ** a ** b ** c ** cc;
q = c ** cc ** bb ** aa ** a;
m = a ** b ** c;
cond1 = Pinv[m, cc ** bb ** aa];
cond4 = {p ** q ** p - p, adj[q] - adj[a] ** a ** p ** v, c ** adj[c] ** adj[p] ** u - q};
cond6 = {q ** p ** q - q, adj[q] ** v - adj[a] ** a ** p, c ** adj[c] ** adj[p] - q ** u};
```

```
In[∘]:= u1 = b ** c ** adj[c] ** adj[b] ** adj[a] ** adj[aa];
u2 = adj[bb] ** adj[cc] ** cc ** bb ** aa ** a;
v1 = adj[b] ** adj[a] ** a ** b ** c ** cc;
v2 = bb ** aa ** adj[aa] ** adj[bb] ** adj[cc] ** adj[c];
cond4Goal = {
    p ** q ** p - p,
    adj[q] - adj[a] ** a ** p ** v2,
    c ** adj[c] ** adj[p] ** u2 - q};
cond6Goal = {
    q ** p ** q - q,
    adj[a] ** a ** p - adj[q] ** v1,
    c ** adj[c] ** adj[p] - q ** u1
   };
```

(i) <=> (iv)

(i) => (iv)

```
In[]:= assumptions = Join[PinvA, PinvC, cond1] // AddAdj;
certificate = Certify[assumptions, cond4Goal, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond4Goal
IntegerCoeffQ[certificate]

Out[]= True

Out[]= True
```

(iv) => (i)

```
In[]:= assumptions = Join[PinvA, PinvC, cond4] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]

Out[]= True

Out[]= True
```

(i) <=> (vi)

(i) => (vi)

```
In[]:= assumptions = Join[PinvA, PinvC, cond1] // AddAdj;
certificate = Certify[assumptions, cond6Goal, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond6Goal
IntegerCoeffQ[certificate]

Out[]= True

Out[]= True
```

(vi) => (i)

In[∘]:=
```
assumptions = Join[PinvA, PinvC, cond6] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= `True`

Out[∘]= `True`

Theorem 2.7

Let A, B, C with inverses $A^{\{1,2,3\}}$, $C^{\{1,2,4\}}$ and let $M = A\,B\,C$. Let $\tilde{B}$ be such that $Q = C\,C^{\{1,2,4\}}\,\tilde{B}\,A^{\{1,2,3\}}\,A$ exists and let $P = A^{\{1,2,3\}}\,A\,B\,C\,C^{\{1,2,4\}}$. Furthermore, let ABC be right *-cancellable and let $C^{\{1,2,4\}}\,\tilde{B}\,A^{\{1,2,3\}}$ be left *-cancellable. Then, the following are equivalent:

i) M is Moore-Penrose invertible and $M^{\dagger} = C^{\{1,2,4\}}\,\tilde{B}\,A^{\{1,2,3\}}$;

ii) Q ∈ P{1,2} and both of $A^*\,A\,P\,Q$ and $Q\,P\,C\,C^*$ are Hermitian;

iii) Q ∈ P{1,2} and both of $A^*\,A\,P\,Q$ and $Q\,P\,C\,C^*$ are EP;

iv) $P\,Q = (P\,Q)^2$, $R(A^*\,A\,P) \supseteq R(Q^*)$, and $R(C\,C^*\,P^*) \subseteq R(Q)$;

v) $P\,Q = (P\,Q)^2$, $R(A^*\,A\,P) \subseteq R(Q^*)$, and $R(C\,C^*\,P^*) \supseteq R(Q)$;

In[∘]:=
```
Q = {{a, 3, 4}, {adj[a], 4, 3}, {aa, 4, 3}, {adj[aa], 3, 4},
      {b, 2, 3}, {adj[b], 3, 2}, {bb, 3, 2}, {adj[bb], 2, 3},
      {c, 1, 2}, {adj[c], 2, 1}, {cc, 2, 1}, {adj[cc], 1, 2},
      {u, 3, 3}, {adj[u], 3, 3}, {v, 3, 3}, {adj[v], 3, 3},
      {w, 2, 2}, {adj[w], 2, 2}, {s, 2, 2}, {adj[s], 2, 2}};
```

```
In[•]:= AInv123 = Delete[Pinv[a, aa], {4}];
        CInv124 = Delete[Pinv[c, cc], {3}];
        p = aa ** a ** b ** c ** cc;
        q = c ** cc ** bb ** aa ** a;
        m = a ** b ** c;
        PInv12 = {p ** q ** p - p, q ** p ** q - q};
        cond1 = Pinv[m, cc ** bb ** aa];
        cond2 = {adj[a] ** a ** p ** q - adj[adj[a] ** a ** p ** q],
            q ** p ** c ** adj[c] - adj[q ** p ** c ** adj[c]]};
        cond3 = {adj[a] ** a ** p ** q ** u - adj[adj[a] ** a ** p ** q],
            adj[a] ** a ** p ** q - adj[adj[a] ** a ** p ** q] ** v,
            q ** p ** c ** adj[c] - adj[q ** p ** c ** adj[c]] ** w,
            q ** p ** c ** adj[c] ** s - adj[q ** p ** c ** adj[c]]};
        cond4 = {p ** q ** p ** q - p ** q, adj[q] - adj[a] ** a ** p ** w,
            c ** adj[c] ** adj[p] - q ** v};
        cond5 = {p ** q ** p ** q - p ** q, adj[q] ** s - adj[a] ** a ** p,
            c ** adj[c] ** adj[p] ** u - q};

        u1 = b ** c ** adj[c] ** adj[b] ** adj[a] ** adj[aa];
        u2 = adj[bb] ** adj[cc] ** cc ** bb ** aa ** a;
        v1 = adj[b] ** adj[a] ** a ** b ** c ** cc;
        v2 = bb ** aa ** adj[aa] ** adj[bb] ** adj[cc] ** adj[c];
        cond4Goal = {
            p ** q ** p ** q - p ** q,
            adj[q] - adj[a] ** a ** p ** v2,
            c ** adj[c] ** adj[p] - q ** u1};
        cond5Goal = {
            p ** q ** p ** q - p ** q,
            adj[a] ** a ** p - adj[q] ** v1,
            q - c ** adj[c] ** adj[p] ** u2};
```

(i) <=> (ii)

(i) => (ii)

```
In[•]:= assumptions = Join[AInv123, CInv124, cond1] // AddAdj;
        certificate =
          Certify[assumptions, Join[PInv12, cond2], Q, MultiLex → True, MaxDeg → 15];
        (* Check if certificate is correct and only contains integer coefficients *)
        MultiplyOut /@ certificate === Join[PInv12, cond2]
        IntegerCoeffQ[certificate]

Out[•]= True

Out[•]= True
```

(ii) => (i)

```
In[•]:= assumptions = Join[AInv123, CInv124, PInv12, cond2] // AddAdj;
certificate = Certify[assumptions, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]

Out[•]= True

Out[•]= True
```

(ii) <=> (iii)

(ii) => (iii)

```
In[•]:= (* This trivially holds *)
```

(iii) => (ii)

```
In[•]:= assumptions = Join[AInv123, CInv124, PInv12, cond3] // AddAdj;
certificate =
  Certify[assumptions, Join[PInv12, cond2], Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === Join[PInv12, cond2]
IntegerCoeffQ[certificate]

Out[•]= True

Out[•]= True
```

(i) <=> (iv)

(i) => (iv)

```
In[•]:= assumptions = Join[AInv123, CInv124, cond1] // AddAdj;
certificate = Certify[assumptions, cond4Goal, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond4Goal
IntegerCoeffQ[certificate]

Out[•]= True

Out[•]= True
```

(iv) => (i)

In[∘]:=
```
(* As in the proof of (v) => (i) of Theorem 2.3,
we show that m̃ = cc**bb**aa is an inner inverse of m
  using the right *-cancellability of m *)
assumptions = Join[AInv123, CInv124, cond4] // AddAdj;
claim = {m ** adj[m] - m ** cc ** bb ** aa ** m ** adj[m]};
certificate1 = Certify[assumptions, claim, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate1 === claim
IntegerCoeffQ[certificate1]
```

Out[∘]= True

Out[∘]= True

In[∘]:=
```
(* Now, we add the information that m̃
 is an inner inverse of m to our assumptions *)
(* Then, the rest runs through automatically *)
assumptionsNew =
  Join[AInv123, CInv124, cond4, {m - m ** cc ** bb ** aa ** m}] // AddAdj;
certificate = Certify[assumptionsNew, cond1, Q, MultiLex → True, MaxDeg → 15];
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[∘]= True

Out[∘]= True

(i) <=> (v)

(i) => (v)

```
In[•]:= assumptions = Join[AInv123, CInv124, cond1] // AddAdj;
certificate = Certify[assumptions, cond5Goal, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond5Goal
IntegerCoeffQ[certificate]
```

```
Out[•]= True

Out[•]= True
```

(v) => (i)

```
In[•]:= (* As in the proof of (v) => (i) of Theorem 2.4,
we show that m̃ = cc**bb**aa is an outer inverse of m
  using the left *-cancellability of m̃ *)
assumptions = Join[AInv123, CInv124, cond5] // AddAdj;
claim = {adj[cc ** bb ** aa] ** cc ** bb ** aa -
    adj[cc ** bb ** aa] ** cc ** bb ** aa ** m ** cc ** bb ** aa};
certificate1 = Certify[assumptions, claim, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate1 === claim
IntegerCoeffQ[certificate1]
```

```
Out[•]= True

Out[•]= True
```

```
In[•]:= (* Now, we add the information that m̃
 is an outer inverse of m to our assumptions *)
(* Then, the rest runs through automatically *)
assumptionsNew =
  Append[assumptions, cc ** bb ** aa - cc ** bb ** aa ** m ** cc ** bb ** aa] // AddAdj;
certificate = Certify[assumptionsNew, cond1, Q, MultiLex → True, MaxDeg → 15];
(* Check if certificate is correct and only contains integer coefficients *)
MultiplyOut /@ certificate === cond1
IntegerCoeffQ[certificate]
```

Out[ ]= True

Out[ ]= True

Theorem 2.9

To prove Theorem 2.9, the computations of Theorems 2.1, 2.3, 2.4. 2.6 can be adopted.