MatrixPresentations package

A subpackage for QuantumGroups v2.
Version 2.0, July 30, 2005, Scott Morrison

Introduction

This package gives explicit matrices for the generators of a quantum group, in an arbitrary representation.

Implementation

Start of package

Specify package dependencies:

```
BeginPackage["QuantumGroups`MatrixPresentations`",
  {"QuantumGroups`", "QuantumGroups`Utilities`IntersectSubspaces`",
   "QuantumGroups`Utilities`MatrixWrapper`", "QuantumGroups`Utilities`Debugging`",
   "QuantumGroups`Utilities`DataPackage`", "QuantumGroups`RootSystems`",
   "QuantumGroups`Algebra`", "QuantumGroups`Representations`"}];
```

Usage messages

```
MatrixPresentation::usage = "";
```

```
HighWeightVectors::usage = "";
```

```
HighWeightVectorQ;
```

```
HighWeights::usage = "";
```

```
AllHighWeightVectors::usage = "";
```

```
TensorProductWeightSpaceInclusion;
```

```
ChangeOfBasisMatrix;
```

```
ShortRootBasis;
```

Internals

```
Begin["`Private`"];
```

```
q = Global`q;
```

```
(*MatrixPresentation[Γ_][A_][CircleTimes[V_],β_,λ_]:=
 MatrixPresentation[Γ][A][V,β,λ]*)
```

```
MatrixPresentation[Γ_][A_ ** B_][V_, β_, λ_] /;
   ! MemberQ[Weights[Γ, V], λ + OperatorWeight[Γ][A ** B]] :=
 Matrix[0, WeightMultiplicity[Γ, V, λ]]
```

```
MatrixPresentation[Γ_][A_ ** B_][V_, β_, λ_] :=
 MatrixPresentation[Γ][A][V, β, λ + OperatorWeight[Γ][B]].
  MatrixPresentation[Γ][B][V, β, λ]
```

```
MatrixPresentation[Γ_][0][V_, _, λ_] :=
 With[{m = WeightMultiplicity[Γ, V, λ]}, ZeroesMatrix[m, m]]
```

```
MatrixPresentation[Γ_][1][V_, _, λ_] := identityMatrix[WeightMultiplicity[Γ, V, λ]]
```

```
MatrixPresentation[Γ_][α_?qNumberQ A_][V_, β_, λ_] :=
 α MatrixPresentation[Γ][A][V, β, λ]
```

```
MatrixPresentation[Γ_][A_Plus][V_, β_, λ_] /;
   (Length[Union[OperatorWeight[Γ][#] & /@ (List @@ A)]] == 1) :=
 MatrixPresentation[Γ][#][V, β, λ] & /@ A
```

```
MatrixPresentation[Γ_][K_i_] [V_, _, λ_] :=
 q^CartanFactors[Γ][[i]] λ[[i]] identityMatrix[WeightMultiplicity[Γ, V, λ]]
MatrixPresentation[Γ_][K_i_^-1][V_, _, λ_] :=
 q^-CartanFactors[Γ][[i]] λ[[i]] identityMatrix[WeightMultiplicity[Γ, V, λ]]
```

```
MatrixPresentation[Γ_][A : (SuperPlus[X_] | SuperMinus[X_])][Irrep[Γ_][μ_], β_, λ_] /;
   (ZeroVectorQ[μ] ∧ Length[μ] == Rank[Γ] ∧ Length[λ] == Rank[Γ]) :=
 If[ZeroVectorQ[λ], ZeroesMatrix[0, 1],
  ZeroesMatrix[WeightMultiplicity[Γ, Irrep[Γ][μ], λ + OperatorWeight[Γ][A]], 0]]
```

```
MatrixPresentation[Γ_][A : (SuperPlus[X_] | SuperMinus[X_])][V_, β_, λ_] /;
   MinusculeRepresentationQ[Γ, V] := OnesMatrix[
   WeightMultiplicity[Γ, V, λ + OperatorWeight[Γ][A]], WeightMultiplicity[Γ, V, λ]]
```

```
MatrixPresentation[Γ_][A_⊗B_][V_⊗W_, β_, λ_] /;
  !MemberQ[Weights[Γ, V⊗W], λ+OperatorWeight[Γ][A⊗B]] :=
 Matrix[0, WeightMultiplicity[Γ, V⊗W, λ]]
```

```
MatrixPresentation[Γ_][A_⊗B_][V_⊗W_, β_, λ_] /;
  OperatorLength[Γ, A] > WeightDiameter[Γ, V] :=
 ZeroesMatrix[WeightMultiplicity[Γ, V⊗W, λ+OperatorWeight[Γ][A⊗B]],
  WeightMultiplicity[Γ, V⊗W, λ]]
MatrixPresentation[Γ_][A_⊗B_][V_⊗W_, β_, λ_] /;
  OperatorLength[Γ, B] > WeightDiameter[Γ, W] :=
 ZeroesMatrix[WeightMultiplicity[Γ, V⊗W, λ+OperatorWeight[Γ][A⊗B]],
  WeightMultiplicity[Γ, V⊗W, λ]]
```

```
MatrixPresentation[Γ_][A_⊗B_][V_⊗W_, β_, λ_] :=
 AppendRows @@ Table[
   TensorProductWeightSpaceInclusion[Γ, {V, W}, {λ+OperatorWeight[Γ][A] -
       Weights[Γ, W][[i]], Weights[Γ, W][[i]]+OperatorWeight[Γ][B]}].Together[
     MatrixKroneckerProduct[MatrixPresentation[Γ][A][V, β, λ-Weights[Γ, W][[i]]],
      MatrixPresentation[Γ][B][W, β, Weights[Γ, W][[i]]]]],
   {i, 1, Length[Weights[Γ, W]]}]
```

```
MatrixPresentation[Γ_][A_][V_⊗W_, β_, λ_] /;
  !MemberQ[Weights[Γ, V⊗W], λ+OperatorWeight[Γ][Δ[A]]] :=
 Matrix[0, WeightMultiplicity[Γ, V⊗W, λ]]
```

```
MatrixPresentation[Γ_][A_][V_⊗W_, β_, λ_] := MatrixPresentation[Γ][Δ[A]][V⊗W, β, λ]
```

Watch out-due to the crappiness of my implementation of BlockDiagonalMatrix, this will puke with more than 256 direct summands.

```
MatrixPresentation[Γ_][A_][V_DirectSum, β_, λ_] :=
 BlockDiagonalMatrix @@ (MatrixPresentation[Γ][A][#, β, λ] & /@ (List @@ V))
```

```
PadWithZeroRows[m_Matrix, initial_?NaturalQ, total_?NaturalQ] :=
 With[{rows = Dimensions[m][[1]], cols = Dimensions[m][[2]]},
  Matrix[total, cols, Join[ConstantArray[0, {initial, cols}],
    MatrixData[m], ConstantArray[0, {total-initial-rows, cols}]]]]
```

```
WeightMultiplicityComponents[Γ_, V1_, V2_, λ_] :=
 Table[WeightMultiplicity[Γ, V1, λ-Weights[Γ, V2][[i]]] ×
   WeightMultiplicity[Γ, V2, Weights[Γ, V2][[i]]], {i, 1, Length[Weights[Γ, V2]]}]
```

```
WeightMultiplicityPartialSums[Γ_, V1_, V2_, λ_] :=
 WeightMultiplicityPartialSums[Γ, V1, V2, λ] =
  Drop[FoldList[Plus, 0, WeightMultiplicityComponents[Γ, V1, V2, λ]], -1]
```

Gives the map from $V_\lambda \otimes W_\mu$ into $(V \otimes W)_{\lambda+\mu}$.

```
TensorProductWeightSpaceInclusion[Γ_, {V_, W_}, {λ_, μ_}] :=
 With[{m0 = WeightMultiplicity[Γ, V, λ] × WeightMultiplicity[Γ, W, μ],
   m1 = WeightMultiplicity[Γ, V⊗W, λ+μ], pos = Position[Weights[Γ, W], μ]},
  If[pos == {} ∨ m0 == 0,
   Matrix[m1, m0],
   PadWithZeroRows[identityMatrix[m0],
    WeightMultiplicityPartialSums[Γ, V, W, λ+μ][[pos[[1, 1]]]], m1]
  ]
 ]
```

```
HighWeightVectorQ[Γ_, V_, b_, λ_] :=
 (And @@ Table[ZeroVectorQ[Together[MatrixPresentation[Γ][SuperPlus[X_i]][V, b, λ].#]],
     {i, 1, Rank[Γ]}]) &
```

```
HighWeightVectors[Γ_][Irrep[Γ_][λ_], _, λ_] := {{1}}
HighWeightVectors[Γ_][Irrep[Γ_][λ_], _, _] := {}
```

In order to allow branching calculations, I changed (2007-11-06) the pattern from:

```
(* HighWeightVectors[Γ_][V:(Irrep[Γ_][_]⊗Irrep[Γ_][_]),b_,λ_]:= *)
```

to the less restrictive:

```
HighWeightVectors[Γ_][V_, b_, λ_] := HighWeightVectors[Γ][V, b, λ] = Module[{T, r},
   T = AppendColumns @@
     Table[MatrixPresentation[Γ][SuperPlus[X_i]][V, b, λ], {i, 1, Rank[Γ]}];
   If[Dimensions[T][[1]] == 0, Return[IdentityMatrix[WeightMultiplicity[Γ, V, λ]]]];
   DebugPrint["About to find the null space of a ",
    Dimensions[T][[2]], " by ", Dimensions[T][[2]], " matrix."];
   r = Together[NullSpace[T, Method → "OneStepRowReduction"]];
   DebugPrint["Finished finding null space."];
   r
  ]
```

Changing PositiveWeights to the more sensible HighWeights in the next line seems to break things. Weight ordering issues?

```
AllHighWeightVectors[Γ_][V_, β_] := Flatten[
  Function[{λ}, {λ, #} & /@ HighWeightVectors[Γ][V, β, λ]] /@ PositiveWeights[Γ, V], 1]
```

```
HighWeights[Γ_, V_] := SortWeights[Γ][Union[DecomposeRepresentation[Γ][V] /.
     {DirectSum[v__] :→ {v}, i : Irrep[_][_] :→ {i}} /. Irrep[Γ][λ_] :→ λ]]
```

```
AddOneVectorToSpanningSet[{}, v_?VectorQ] := If[ZeroVectorQ[v], {}, {v}]
```

```
AddOneVectorToSpanningSet[m_?MatrixQ, v_?VectorQ] :=
 If[MatrixRank[m~Join~{v}] == Length[m], m, m~Join~{v}]
```

```
SpanningSet[{}] := {};
```

```
SpanningSet[m_?MatrixQ] := (Global`spanningSetArgument = m;
  Fold[AddOneVectorToSpanningSet, {}, m])
```

```
SpanningSet[Matrix[_, _, data_]] := SpanningSet[data]
```

```
LazyAddOneVectorToSpanningSet[{}, v_?VectorQ] := If[ZeroVectorQ[v], {}, {v}]
```

```
LazyAddOneVectorToSpanningSet[m_?MatrixQ, v_?VectorQ] :=
 If[MatrixRank[m~Join~{v} /. q → 1] == Length[m], m, m~Join~{v}]
```

```
LazySpanningSet[{}] := {};
```

```
LazySpanningSet[m_?MatrixQ] := (Global`spanningSetArgument = m;
  Fold[LazyAddOneVectorToSpanningSet, {}, m])
```

```
LazySpanningSet[Matrix[_, _, data_]] := LazySpanningSet[data]
```

```
CarefulSpanningSetNewTime = CarefulSpanningSetOldTime = 0;
```

```
CarefulSpanningSet[m_] := Module[{tnew, rnew, told, rold},
  {tnew, rnew} = AbsoluteTiming[LazySpanningSet[m]];
  {told, rold} = AbsoluteTiming[SpanningSet[m]];
  If[rnew =!= rold, Print["LazySpanningSet failed!"]];
  CarefulSpanningSetNewTime += tnew;
  CarefulSpanningSetOldTime += told;
  rnew
 ]
```

```
SubIrrepWeightBasis[Γ_][V_, β_, λ_, v_, λ_] := {v}
```

```
SubIrrepWeightBasis[Γ_][V_, β_, λ_, v_, μ_] /; InWeylPolytopeQ[Γ, λ, μ] :=
 SubIrrepWeightBasis[Γ][V, β, λ, v, μ] = Module[{c, r},
   DebugPrintHeld["Calculating ", SubIrrepWeightBasis[Γ][V, β, λ, v, μ]];
   c = Join @@ Table[MatrixPresentation[Γ][SuperMinus[Xi]][V, β,
          μ - OperatorWeight[Γ][SuperMinus[Xi]]].# & /@ SubIrrepWeightBasis[Γ][
        V, β, λ, v, μ - OperatorWeight[Γ][SuperMinus[Xi]]], {i, 1, Rank[Γ]}];
   DebugPrint[" ... prepared spanning set."];
   r = LazySpanningSet[Together[c]];
   DebugPrintHeld["Finished calculating ", SubIrrepWeightBasis[Γ][V, β, λ, v, μ]];
   r
  ]
```

```
SubIrrepWeightBasis[Γ_][V_, β_, λ_, v_, μ_] := {}
```

```
FullSubIrrepWeightBasis[Γ_][V_, β_, μ_] := Flatten[
  SubIrrepWeightBasis[Γ][V, β, #[[1]], #[[2]], μ] & /@ AllHighWeightVectors[Γ][V, β], 1]
```

```
DirectSumDecomposition[Γ_][V_, β_, λ_] := DirectSumDecomposition[Γ][V, β, λ] =
  With[{data = FullSubIrrepWeightBasis[Γ][V, β, λ], d = WeightMultiplicity[Γ, V, λ]},
   If[Length[data] ≠ d,
    Print["Direct sum decomposition failed, in weight space ", λ, " of ", V];
    Return[$Failed]];
   Transpose[Matrix[Length[data], d, data]]
  ]
```

```
InverseDirectSumDecomposition[Γ_][V_, β_, λ_] :=
 InverseDirectSumDecomposition[Γ][V, β, λ] = Inverse[DirectSumDecomposition[Γ][V, β, λ]]
```

```
DirectSumProjection[Γ_][V_DirectSum, index_Integer, λ_] :=
 With[{a = WeightMultiplicity[Γ, Take[V, index - 1], λ], b =
    WeightMultiplicity[Γ, V[[index]], λ], c = WeightMultiplicity[Γ, Drop[V, index], λ]},
  AppendRows[ZeroesMatrix[b, a], identityMatrix[b], ZeroesMatrix[b, c]]
 ]
```

```
DirectSumProjection[Γ_][V_DirectSum, indexes : {___Integer}, λ_] :=
 AppendColumns @@ (DirectSumProjection[Γ][V, #, λ] & /@ indexes)
```

```
DirectSumInclusion[Γ_][V_DirectSum, index_, λ_] :=
 Transpose[DirectSumProjection[Γ][V, index, λ]]
```

```
ChangeOfBasisMatrix[Γ_][V : Irrep[Γ_][_], β_, FundamentalBasis, λ_] :=
 DirectSumDecomposition[Γ][V, β, λ]
ChangeOfBasisMatrix[Γ_][V : Irrep[Γ_][_], FundamentalBasis, β_, λ_] :=
 Inverse[ChangeOfBasisMatrix[Γ][V, β, FundamentalBasis, λ]]
ChangeOfBasisMatrix[Γ_][V_, β_, β_, λ_] := identityMatrix[WeightMultiplicity[Γ, V, λ]]
```

```
MatrixPresentation[Γ_][A : (SuperPlus[X_] | SuperMinus[X_])][Irrep[Γ_][λ_],
    ShortRootBasis, κ_] /; UnitVectorQ[λ] ∧ ShortDominantRootQ[Γ, λ] :=
 MatrixPresentation[Γ][A][Irrep[Γ][λ], ShortRootBasis, μ] =
  Module[{x, h, s, d, kf, action, basis},
   s_i := SimpleRoots[Γ][[i]];
   d_i := CartanFactors[Γ][[i]];
   kf = KillingForm[Γ];
   action[SuperPlus[X_i]][x_μ] := Switch[2 kf[μ, s_i]/kf[s_i, s_i], 2 | 0 | 1, 0, -1, x_(μ+s_i), -2, h_(s_i)];
   action[SuperMinus[X_i]][x_μ] := Switch[2 kf[μ, s_i]/kf[s_i, s_i], -2 | 0 | -1, 0, 1, x_(μ-s_i), 2, h_(s_i)];
   action[SuperPlus[X_i]][h_μ] :=
    Switch[2 kf[μ, s_i]/kf[s_i, s_i], 2, qInteger[2][q^(d_i)], -1, 1, _, 0] x_(s_i);
   action[SuperMinus[X_i]][h_μ] :=
    Switch[2 kf[μ, s_i]/kf[s_i, s_i], 2, qInteger[2][q^(d_i)], -1, 1, _, 0] x_(-s_i);
   basis[μ_] := If[MemberQ[ShortRoots[Γ], μ], {x_μ},
     If[ZeroVectorQ[μ], h_# & /@ ShortSimpleRoots[Γ], {}]];
   With[{β1 = basis[κ], β2 = basis[κ + OperatorWeight[Γ][A]]},
    Matrix[Length[β2], Length[β1], Coefficient[action[A] /@ β1, #] & /@ β2]
   ]
  ]
```

```
General::spell1 : Possible spelling error: new symbol name "action" is similar to existing symbol
   "Action". More…
```

```
MatrixPresentation[Γ_][A : (SuperPlus[X_] | SuperMinus[X_])][V : Irrep[Γ_][λ_],
    FundamentalBasis, κ_] /; UnitVectorQ[λ] ∧ ShortDominantRootQ[Γ, λ] :=
 MatrixPresentation[Γ][A][V, FundamentalBasis, κ] =
  ChangeOfBasisMatrix[Γ][V, FundamentalBasis, ShortRootBasis, κ + OperatorWeight[Γ][A]].
   MatrixPresentation[Γ][A][V, ShortRootBasis, κ].
   ChangeOfBasisMatrix[Γ][V, ShortRootBasis, FundamentalBasis, κ]
```

```
IrrepContainmentRules = {
   Irrep[B_n_][λ_] /; MemberQ[Take[IdentityMatrix[n], {2, -2}], λ] :>
    Irrep[B_n][RotateLeft[λ]] ⊗ Irrep[B_n][UnitVector[n, 1]],
   Irrep[C_n_][λ_] /; MemberQ[Drop[IdentityMatrix[n], 2], λ] :>
    Irrep[C_n][RotateLeft[λ]] ⊗ Irrep[C_n][UnitVector[n, 1]],
   Irrep[D_n_][λ_] /; MemberQ[Take[IdentityMatrix[n], {3, -3}], λ] :>
    If[EvenQ[n - Position[IdentityMatrix[n], λ][[1, 1]]],
     Irrep[D_n][UnitVector[n, n]] ⊗ Irrep[D_n][UnitVector[n, n]],
     Irrep[D_n][UnitVector[n, n - 1]] ⊗ Irrep[D_n][UnitVector[n, n]]],
   Irrep[G_2][{0, 1}] → Irrep[G_2][{1, 0}]⊗^2,
   Irrep[F_4][{1, 0, 0, 0}] → Irrep[F_4][{0, 0, 0, 1}]⊗^2,
   Irrep[F_4][{0, 0, 1, 0}] → Irrep[F_4][{0, 0, 0, 1}]⊗^2,
   Irrep[F_4][{0, 1, 0, 0}] → Irrep[F_4][{1, 0, 0, 0}]⊗^2,
   Irrep[E_6][{0, 0, 1, 0, 0, 0}] → Irrep[E_6][{1, 0, 0, 0, 0, 0}]⊗^2,
   Irrep[E_6][{0, 1, 0, 0, 0, 0}] →
    Irrep[E_6][{1, 0, 0, 0, 0, 0}] ⊗ Irrep[E_6][{0, 0, 0, 0, 0, 1}],
   Irrep[E_6][{0, 0, 0, 0, 1, 0}] → Irrep[E_6][{0, 0, 0, 0, 0, 1}]⊗^2,
   Irrep[E_6][{0, 0, 0, 1, 0, 0}] →
    Irrep[E_6][{1, 0, 0, 0, 0, 0}] ⊗ Irrep[E_6][{0, 0, 1, 0, 0, 0}],
   Irrep[e_7][{0, 0, 1, 0, 0, 0, 0}] → Irrep[E_7][{1, 0, 0, 0, 0, 0, 0}]⊗^2,
   Irrep[e_7][{0, 1, 0, 0, 0, 0, 0}] →
    Irrep[E_7][{1, 0, 0, 0, 0, 0, 0}] ⊗ Irrep[E_7][{0, 0, 0, 0, 0, 0, 1}],
   Irrep[e_7][{0, 0, 0, 0, 0, 1, 0}] → Irrep[E_7][{1, 0, 0, 0, 0, 0, 0}]⊗^2,
   Irrep[e_7][{0, 0, 0, 0, 1, 0, 0}] →
    Irrep[E_7][{1, 0, 0, 0, 0, 0, 0}] ⊗ Irrep[E_7][{0, 1, 0, 0, 0, 0, 0}],
   Irrep[e_7][{0, 0, 0, 1, 0, 0, 0}] → Irrep[E_7][{0, 0, 1, 0, 0, 0, 0}] ⊗
     Irrep[E_7][{1, 0, 0, 0, 0, 0, 0}],
   Irrep[e_8][{0, 0, 0, 0, 0, 0, 1, 0}] -> Irrep[E_8][{0, 0, 0, 0, 0, 0, 0, 1}]⊗^2,
   Irrep[e_8][{1, 0, 0, 0, 0, 0, 0, 0}] -> Irrep[E_8][{0, 0, 0, 0, 0, 0, 0, 1}]⊗^2,
   Irrep[e_8][{0, 1, 0, 0, 0, 0, 0, 0}] ->
    Irrep[E_8][{1, 0, 0, 0, 0, 0, 0, 0}] ⊗ Irrep[E_8][{0, 0, 0, 0, 0, 0, 0, 1}],
   Irrep[e_8][{0, 0, 0, 0, 0, 1, 0, 0}] -> Irrep[E_8][{1, 0, 0, 0, 0, 0, 0, 0}]⊗^2,
   Irrep[e_8][{0, 0, 1, 0, 0, 0, 0, 0}] -> Irrep[E_8][{1, 0, 0, 0, 0, 0, 0, 0}]⊗^2,
   Irrep[e_8][{0, 0, 0, 0, 1, 0, 0, 0}] ->
    Irrep[E_8][{0, 1, 0, 0, 0, 0, 0, 0}] ⊗ Irrep[E_8][{1, 0, 0, 0, 0, 0, 0, 0}],
   Irrep[e_8][{0, 0, 0, 1, 0, 0, 0, 0}] -> Irrep[E_8][{0, 0, 1, 0, 0, 0, 0, 0}] ⊗
     Irrep[E_8][{1, 0, 0, 0, 0, 0, 0, 0}],
   Irrep[Γ_][λ_] /; (!UnitVectorQ[λ] ∧ !ZeroVectorQ[λ]) :>
    Module[{n = Rank[Γ], pos, μ}, pos = Position[λ, _?(# ≠ 0 &)][[1, 1]];
     μ = UnitVector[n, pos];
     Irrep[Γ][λ - μ] ⊗ Irrep[Γ][μ]]
  };

MatrixPresentation[A_1][Z : (SuperPlus[X_1] | SuperMinus[X_1])][
  Irrep[A_1][{λ_}], FundamentalBasis, {μ_}] :=
 fastMatrixPresentation[A_1][Z][Irrep[A_1][{λ}], FundamentalBasis, {μ}]
```

```
MatrixPresentation[Γ_][A : (SuperPlus[X_] | SuperMinus[X_])][
   V : Irrep[Γ_][μ_], FundamentalBasis, λ_] /; ¬ ZeroVectorQ[μ] ∧ ¬ UnitVectorQ[μ] ∨
    (¬ ShortDominantRootQ[Γ, μ] ∧ ¬ MinusculeRepresentationQ[Γ, V]) :=
 Module[{W = V /. IrrepContainmentRules, p, result},
  MatrixPresentation[Γ][A][V, FundamentalBasis, λ] =
   (DebugPrintHeld["Calculating ", MatrixPresentation[Γ][A][V, FundamentalBasis, λ],
     " by looking at ", V, " as a subrep of ", Evaluate[W]];
    If[W == V, Print["Warning, couldn't work out how to find matrix presentations for ",
      V];
     Return[$Failed]];
    p = Position[DecomposeRepresentation[Γ][W], V][[1, 1]];
    result =
     Simplify[Together[DirectSumProjection[Γ][DecomposeRepresentation[Γ][W], p, λ +
           OperatorWeight[Γ][A]].InverseDirectSumDecomposition[Γ][W, FundamentalBasis,
         λ + OperatorWeight[Γ][A]].MatrixPresentation[Γ][A][W, FundamentalBasis, λ].
        DirectSumDecomposition[Γ][W, FundamentalBasis, λ].
        DirectSumInclusion[Γ][DecomposeRepresentation[Γ][W], p, λ]]];
    DebugPrintHeld["Finished calculating ", MatrixPresentation[Γ][A][
      V, FundamentalBasis, λ]];
    result);
  ReportSavedMatrixPresentation[Γ][A][V, FundamentalBasis, λ];
  result
 ]
```

```
numberOfSavedMatrixPresentations[_] := 0
```

```
autopackagingMatrixPresentations = True;
```

```
ReportSavedMatrixPresentation[Γ_][A_][V_, β_, λ_] := Module[{},
  ++numberOfSavedMatrixPresentations[Γ];
  If[
   autopackagingMatrixPresentations ∧ Mod[numberOfSavedMatrixPresentations[Γ], 10] == 0,
   PackageMatrixPresentations[Γ]
  ];
 ]
```

```
fastMatrixPresentation[A1][SuperPlus[X1]][Irrep[A1][{λ_}], FundamentalBasis, {μ_}] /;
  -λ ≤ μ < λ ∧ EvenQ[λ - μ] :=
 Matrix[1, 1, {{Sum[qInteger[ν][q], {ν, Max[μ + 2, -μ], λ, 2}]}}]

fastMatrixPresentation[A1][SuperPlus[X1]][
  Irrep[A1][{λ_}], FundamentalBasis, {λ_}] := Matrix[0, 1, {}]
fastMatrixPresentation[A1][SuperPlus[X1]][Irrep[A1][{λ_}], FundamentalBasis, {μ_}] /;
  μ == -λ - 2 := Matrix[1, 0, {{}}]
fastMatrixPresentation[A1][SuperPlus[X1]][Irrep[A1][{λ_}],
  FundamentalBasis, {μ_}] := Matrix[0, 0]

fastMatrixPresentation[A1][SuperMinus[X1]][Irrep[A1][{λ_}], FundamentalBasis, {μ_}] /;
  -λ < μ ≤ λ ∧ EvenQ[λ - μ] := Matrix[1, 1, {{1}}]
```

```
fastMatrixPresentation[A1][SuperMinus[X1]][Irrep[A1][{λ_}], FundamentalBasis, {μ_}] /;
  λ == -μ := Matrix[0, 1, {}]
fastMatrixPresentation[A1][SuperMinus[X1]][Irrep[A1][{λ_}], FundamentalBasis, {μ_}] /;
  μ == λ + 2 := Matrix[1, 0, {{}}]
fastMatrixPresentation[A1][SuperMinus[X1]][Irrep[A1][{λ_}],
  FundamentalBasis, {μ_}] := Matrix[0, 0]
```

```
End[];
```

End of package