Braiding package

A subpackage for QuantumGroups v2.
Version 2.0, June 22, 2006, Scott Morrison

Introduction

This package calculates braid group actions on tensor products of representations, and restrictions to high weight vectors.

Implementation

Start of package

Specify package dependencies:

```
BeginPackage["QuantumGroups`Braiding`", {"QuantumGroups`", "WikiLink`",
   "QuantumGroups`Utilities`MatrixWrapper`", "QuantumGroups`Utilities`Debugging`",
   "QuantumGroups`Utilities`DataPackage`", "QuantumGroups`Representations`",
   "QuantumGroups`MatrixPresentations`", "QuantumGroups`RepresentationTensors`"}];
```

```
BR = KnotTheory`BR;
```

Usage messages

```
CheckBraidingData::usage = "";
```

```
BraidingData::usage = "";
```

```
PackageBraidingData::usage =
  "PackageBraidingData[Γ] writes currently known braiding data for the
    quantum group Γ into a data directory in the QuantumGroups` package.";
```

```
WriteBraidingDataToWiki::usage =
  "WriteBraidingDataToWiki[V,k] tries to calculate braiding data for the k-fold tensor
    power of a representation V, and saves the results in the Knot Atlas.";
```

```
LoadAllBraidingDataFromWiki::usage =
  "LoadAllBraidingDataFromWiki[] retrieves all currently
    calculated braiding data from the Knot Atlas.";
```

Internals

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

```
BR[2, {1}][Γ_, {V1_, V2_}, β_] := NormalisedBraidingMap[Γ, V1 ⊗ V2, β]
```

```
BR[n_, {1}][Γ_, V_List, β_] := BR[n, {1}][Γ, V, β] = Fold[#1 ⊗ #2 &,
   NormalisedBraidingMap[Γ, V[[1]] ⊗ V[[2]], β], IdentityMap[Γ, #, β] & /@ Drop[V, 2]]
```

```
BR[n_, {-1}][Γ_, V_List, β_] := BR[n, {-1}][Γ, V, β] = Inverse[BR[n, {1}][Γ, V, β]]
```

```
BR[n_, {k_Integer}][Γ_, V_List, β_] /; 1 < k < n :=
 BR[n, {k}][Γ, V, β] = Module[{ib, as, ias, aib, r},
   DebugPrintHeld["Calculating (what a waste!) ", BR[n, {k}][Γ, V, β]];
   ib = IdentityMap[Γ, CircleTimes @@ Take[V, k - 1], β] ⊗
     NormalisedBraidingMap[Γ, V[[k]] ⊗ V[[k + 1]], β];
   as = Associator[Γ, CircleTimes @@ Take[V, k - 1], V[[k]], V[[k + 1]], β];
   ias = InverseAssociator[Γ, CircleTimes @@ Take[V, k - 1], V[[k]], V[[k + 1]], β];
   aib = as.ib.ias;
   r = Fold[#1 ⊗ #2 &, aib, IdentityMap[Γ, #, β] & /@ Drop[V, k + 1]];
   DebugPrint["... finished calculating, result ", ByteCount[r], " bytes"];
   r
  ]
```

```
BR[n_, {k_Integer}][Γ_, V_List, β_] /; 1 < -k < n :=
 BR[n, {k}][Γ, V, β] = Module[{ib, as, ias, aib, r},
   DebugPrintHeld["Calculating (what a waste!) ", BR[n, {k}][Γ, V, β]];
   ib = IdentityMap[Γ, CircleTimes @@ Take[V, (-k) - 1], β] ⊗
     InverseNormalisedBraidingMap[Γ, V[[-k]] ⊗ V[[-k + 1]], β];
   as = Associator[Γ, CircleTimes @@ Take[V, -k - 1], V[[-k]], V[[-k + 1]], β];
   ias = InverseAssociator[Γ, CircleTimes @@ Take[V, -k - 1], V[[-k]], V[[-k + 1]], β];
   aib = as.ib.ias;
   r = Fold[#1 ⊗ #2 &, aib, IdentityMap[Γ, #, β] & /@ Drop[V, -k + 1]];
   DebugPrint["... finished calculating, result ", ByteCount[r], " bytes"];
   r
  ]
```

```
BR[n_Integer, {k1_Integer, k2_Integer}][Γ_, V_List, β_] :=
 BR[n, {k1, k2}][Γ, V, β] = Simplify[BR[n, {k1}][Γ, V, β].BR[n, {k2}][Γ, V, β]]
```

```
BR[n_Integer, k_][Γ_, V : Irrep[Γ_][_], β_] := BR[n, k][Γ, Table[V, {n}], β]
```

```
BR[n_Integer, ks : {__Integer}][Γ_, V_List, β_] :=
 BR[n, ks][Γ, V, β] = Simplify[BR[n, Take[ks, Floor[Length[ks] / 2]]][Γ, V, β].
    BR[n, Drop[ks, Floor[Length[ks] / 2]]][Γ, V, β]]
```

(*For the future; this is a bottleneck. Perhaps we could replace LinearSolve with a hand-written algorithm using interpolation.*)

```
ChangeBasis[map_, basis_] := Module[{},
  DebugPrint["ChangeBasis called with ", Dimensions[map], " ", Length[basis]];
  lastChangeBasisArguments = {map, basis};
  Together[Transpose[LinearSolve[Transpose[basis],
     Together[map.Transpose[basis]], Method → OneStepRowReduction]]]
 ]
```

```
BraidingMatrices[Γ_][V_, n_Integer, λ_] :=
 BraidingMatrices[Γ][V, n, λ] = Module[{a, hwv, matrices, inverses},
   DebugPrintHeld["Calculating ", BraidingMatrices[Γ][V, n, λ]];
   hwv = HighWeightVectors[Γ][V^⊗n, FundamentalBasis, λ];
   DebugPrint["Changing basis ..."];
   matrices = Table[ChangeBasis[
      MatrixData[BR[n, {i}][Γ, V, FundamentalBasis][λ]], hwv], {i, 1, n - 1}];
   DebugPrint["Computing inverses directly ..."];
   inverses = MatrixInverse /@ matrices;
   DebugPrint["Computing inverses again! ..."];
   inverses = Table[ChangeBasis[
      MatrixData[BR[n, {-i}][Γ, V, FundamentalBasis][λ]], hwv], {i, 1, n - 1}];
   DebugPrint["Finished calculating braiding matrices."];
   Together[{matrices, inverses}]
  ]
```

```
LoadBraidingData[Γ_n_] := Module[{},
  Off[Get::noopen, Needs::nocont];
  Needs["QuantumGroups`Data`" <> SymbolName[Γ] <> ToString[n] <> "`BraidingData`"];
  On[Get::noopen, Needs::nocont];
  LoadBraidingData[Γ_n] = False;
  True
 ]
```

```
autosaveBraidingData = True;
```

```
BraidingData[Γ_][V_, n_Integer] := Module[{result},
  BraidingData[Γ][V, n] = result =
    If[LoadBraidingData[Γ], BraidingData[Γ][V, n],
     {qDimension[Γ][Irrep[Γ][#]], BraidingMatrices[Γ][V, n, #]} & /@
      HighWeights[Γ, V^⊗n]
    ];
  If[autosaveBraidingData, PackageBraidingData[Γ]];
  result
 ]
```

```
CheckBraidingData[m : {__?MatrixQ}] :=
 And @@ Table[ZeroMatrixQ[Matrix[Simplify[m[[i]].m[[i + 1]].m[[i]] - m[[i + 1]].m[[i]].m[[i + 1]]]]],
   {i, 1, Length[m] - 1}]
```

```
CheckBraidingData[{m : {__?MatrixQ}, i : {__?MatrixQ}}] := (Length[m] == Length[i]) ∧
   (And @@ Table[ZeroMatrixQ[Matrix[Simplify[m[[k]].i[[k]] - IdentityMatrix[Length[m[[k]]]]]]],
      {k, 1, Length[m]}]) ∧ CheckBraidingData[m]
```

```
CheckBraidingData[d : {{_, {{__?MatrixQ}, {__?MatrixQ}}} ..}] :=
 And @@ (CheckBraidingData /@ (Last /@ d))
```

```
CheckBraidingData[Γ_][V_, k_] := CheckBraidingData[BraidingData[Γ][V, k]]
```

```
PackageBraidingData[Γ_n_] := PackageData[
  BraidingData, BraidingData[Γ_n][_, _],
  {ToString[Γ] <> ToString[n], "BraidingData"},
  "Needs" → {"QuantumGroups`", "QuantumGroups`Braiding`"},
  "ExtraPrivateCode" → "q=Global`q;"
 ]
```

```
listToString[x_List] := listToString[x, ","]
```

```
listToString[x_List, s_String] :=
 StringJoin[Drop[Flatten[Transpose[{ToString /@ x, Table[s, {Length[x]}]}]], -1]]
```

```
WriteBraidingDataToWiki[V_, k_] := WriteBraidingDataToWiki[V, k, 200 * 10^6]
```

```
WriteBraidingDataToWiki[Irrep[Γ_n_][λ_], k_, M_] := MemoryConstrained[Module[{},
    If[CheckBraidingData[Γn][Irrep[Γn][λ], k] =!= True,
     Print["The braiding data for ", Irrep[Γn][λ]^"⊗"<>ToString[k], " is invalid!"];
     Return[$Failed]];
    PackageBraidingData[Γn];
    WikiSetPageText["Data:Braiding/" <> ToString[Γ] <>
      "_" <> ToString[n] <> "/" <> listToString[λ] <> "/" <> ToString[k],
     ToString[BraidingData[Γn][Irrep[Γn][λ], k], InputForm]]
   ],
   M] /. $Aborted :>
   (Print["Aborted because the calculation exceeded the current memory limit: ", M];
    $Aborted)
```

```
LoadBraidingDataFromWiki[Irrep[Γ_n_][λ_], k_] := Module[{text},
  text = WikiGetPageText["Data:Braiding/" <> ToString[Γ] <>
     "_" <> ToString[n] <> "/" <> listToString[λ] <> "/" <> ToString[k]];
  If[text == "" ∨ text == $Failed, Return[$Failed]];
  BraidingData[Γn][Irrep[Γn][λ], k] = ToExpression[text, InputForm]
 ]
```

```
LoadAllBraidingDataFromWiki[] := Module[{targets, irrep, results = {}},
  targets = WikiAllPages[
    "http://katlas.math.toronto.edu/w/index.php", "Braiding", "Data", 100];
  StringCases[#, "Data:Braiding/" ~~ Γ_ ~~ "_" | " " ~~ n : (DigitCharacter ..) ~~
          "/" ~~ λ : ((DigitCharacter .. ~~ ",") ... ~~ DigitCharacter ..) ~~
            "/" ~~ k : (DigitCharacter ..) :>
      (irrep = Irrep[ToExpression["Γ"<>#]][ToExpression["{" <> λ <> "}"]];
       Print["Loading braiding data for ", irrep^"⊗"<>k];
       results = results ~Join~ {{irrep, ToExpression[k]}};
       LoadBraidingDataFromWiki[irrep, ToExpression[k]])
    ] & /@ targets;
  results
 ]
```

```
End[];
```

End of package

```
EndPackage[];
```