KnotTheory`UniversalKh` package

A subpackage for KnotTheory`, providing access to the "universal mode" of Jeremy Green's JavaKh program, and a program written by Scott Morrison to decompose complexes over Q[t] into indecomposable direct summands.

May 27, 2007, Scott Morrison

```
BeginPackage["KnotTheory`UniversalKh`",
  {"KnotTheory`", "QuantumGroups`Utilities`MatrixWrapper`"}];
```

```
UniversalKh::about =
  "Universal Khovanov homology over Q[t] is calculated using Jeremy Green's
    JavaKh program, interpreted by a wrapper written by Dror
    Bar-Natan, and decomposed into direct summands using a
    program of Scott Morrison and Alexander Shumakovitch.";
```

```
UniversalKh::usage =
  "UniversalKh[K] computes the universal Khovanov homology over Q. (Probably broken
    for links, at present.) See also KhC and KhE for more about the output.";
```

```
sInvariant::usage =
  "sInvariant[K] computes the s-invariant of a knot K, using the UniversalKh
    program. (Probably broken for links, at present.)";
```

```
KhReduced::usage = "KhReduced[K][q,t] gives the reduced
    Khovanov homology of the knot K, using the UniversalKh program.";
```

```
KhE::usage = "KhE denotes a free generator in Khovanov homology, corresponding
   to an exceptional pair. See ?UniversalKh for more information."
KhC::usage = "KhC denotes a torsion generator in Khovanov homology, with the
   differential in KhC[n] being the n-th power of the punctured torus. Thus KhC[1]
   corresponds to a knight's move pair. See ?UniversalKh for more information."
```

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

```
KnotTheory`UniversalKh`Private`
```

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

```
KhN[pd_PD, options___] := KhN[pd, options] = Module[
   {n, pd1,  f,  cl,  out, kh, saveContext, saveContextPath, JavaKhDirectory,
    jarDirectory, classDirectory, classpath, new = True, javaoptions},

   javaoptions = (JavaOptions /. {options} /. Options[Kh]);
```

```
n = Max @@ (Max @@@ pd);
pd1 = pd /. {
   X[n, i_, 1, j_] :> X[n, i, n + 1, j],
   X[i_, 1, j_, n] :> X[i, n + 1, j, n],
   X[1, j_, n, i_] :> X[n + 1, j, n, i],
   X[j_, n, i_, 1] :> X[j, n, i, n + 1]
  };

Print["pd1->", pd1];

new = True; (* This is just an option for Scott,
to allow comparing against Jeremy's program before butchering it. *)
If[new,
 JavaKhDirectory = ToFileName[KnotTheoryDirectory[], "JavaKh-v2"];
 classpath = KnotTheory`FastKh`JavaKhv2ClassPath[];,
 JavaKhDirectory = ToFileName[KnotTheoryDirectory[], "JavaKh-v1"];
 classpath = KnotTheory`FastKh`JavaKhv1ClassPath[];,
];

SetDirectory[JavaKhDirectory];
f = OpenWrite["pd", PageWidth → Infinity];
WriteString[f, ToString[pd1]];
Close[f];

cl = StringJoin["!java -classpath \"", classpath, "\" ",
  javaoptions, " org.katlas.JavaKh.JavaKh -U -Q < pd 2> JavaKh.log"];
f = OpenRead[cl];
out = Read[f, Expression];
Close[f];

If[out == EndOfFile,
 Print["Something went wrong running JavaKh; nothing was returned.
    The command line was: "];
 Print[cl];
 Print["There may have been an error log produced by Java: "];
 FilePrint["JavaKh.log"]; Return[$Failed]];

ResetDirectory[];

out = StringReplace[out, {"q" → "#1", "t" → "#2"}];
(* ToExpression is dangerous! We have to fiddle with the $Context here. *)
saveContext = $Context;
saveContextPath = $ContextPath;
$Context = "KnotTheory`UniversalKh`Private`";
$ContextPath = {"KnotTheory`UniversalKh`Private`"};
kh = ToExpression[out <> "&"][q, t];
$Context = saveContext;
$ContextPath = saveContextPath;
Print["kh->", kh];
minr = Exponent[kh, t, Min];
maxr = Exponent[kh, t, Max];
```

```
  obs = Expand[kh /. h → 0 /. M[_, n_, ___] :> Plus @@ Array[Arc, n]];
  obs = obs /. (q^j_.) * Arc[i_] :> Arc[j, i] /. Arc[i_] :> Arc[0, i];
  mos = Expand[
    h * kh /. {M[0, _] → 0, M[_, 0] → 0, h → H}
     /. M[m_, n_, cs___] :> Plus @@ Flatten[MapIndexed[
          (#1 * Curtain @@ Reverse[#2]) &,
          Partition[{cs}, n],
          {2}
         ]]
   ];
  mos = mos /. (q^j_.) * Curtain[k_, l_] :> Curtain[j, k, l] /.
    Curtain[k_, l_] :> Curtain[0, k, l];
  mos = mos /. (H^g_.) * Curtain[j_, k_, l_] :>
     H^(g - 1) Curtain[j, k, j + 2 (g - 1), l];
  Komplex @@ Table[{r, Coefficient[obs, t, r], Coefficient[mos, t, r]},
    {r, minr, maxr}]
 ]
```

```
ElementaryMatrix[m_, n_, i_, j_] := ElementaryMatrix[m, n, i, j, 1]
```

```
ElementaryMatrix[m_, n_, i_, j_, z_] /; 1 ≤ i ≤ m ∧ 1 ≤ j ≤ n :=
 Module[{data},
  data = Table[0, {m}, {n}];
  data[[i, j]] = z;
  Matrix[data]
 ]
```

```
GradingsList[k : Komplex[{n_, _, _}, ___]] :=
 {n, Cases[{#}, Arc[m_, _] :> m, 2] & /@ (List @@ k)[[All, 2]]}
```

```
AllMatrices[k : Komplex[{n_, _, _}, ___]] := {n,
  Module[{gradings = GradingsList[k][[2]], dimensions, matrix},
    dimensions = Length /@ gradings;
    Table[
     matrix = ZeroesMatrix[dimensions[[i + 1]], dimensions[[i]]];
     matrix = matrix + (k[[i, 3]] /. (Curtain[q1_, m1_, q2_, m2_] :> ElementaryMatrix[
            dimensions[[i + 1]], dimensions[[i]], Position[gradings[[i + 1]], q2][[1, 1]] + m2 - 1,
            Position[gradings[[i]], q1][[1, 1]] + m1 - 1]))
     , {i, 1, Length[k] - 1}]
   ] /. {H → T}
 }
```

```
ZeroVector[n_] := Table[0, {n}]
```

```
Matrix /: α_ Matrix[j_, k_, data_] /; (NumberQ[α /. T → 3.14159`]) := Matrix[j, k, α data]
```

```
FirstRow[Matrix[r_, c_, data_]] := Matrix[1, c, {First[data]}]
FirstRow[Matrix[0, c_, _]] := Matrix[0, c]
FirstColumn[Matrix[r_, c_, data_]] := Matrix[r, 1, {First[#]} & /@ data]
FirstColumn[Matrix[r_, 0, _]] := Matrix[r, 0]
RestColumns[Matrix[r_, c_, data_]] := Matrix[r, c - 1, Rest /@ data]
RestColumns[Matrix[r_, 0 | 1, _]] := Matrix[r, 0]
RestRows[Matrix[r_, c_, data_]] := Matrix[r - 1, c, Rest[data]]
RestRows[Matrix[0 | 1, c_, _]] := Matrix[0, c]
```

```
RotateRows[Matrix[r_, c_, data_]] := Matrix[r, c, RotateLeft[data]]
RotateColumns[Matrix[r_, c_, data_]] := Matrix[r, c, RotateLeft /@ data]
```

```
RotateRows[Matrix[r_, c_, data_], n_] := Matrix[r, c, RotateLeft[data, n]]
RotateColumns[Matrix[r_, c_, data_], n_] := Matrix[r, c, RotateLeft[#, n] & /@ data]
```

```
UniversalKhTimingData = {};
```

```
twist[α_, k_, λ_, μ_, ν_] := ν - (1 / α) T^(-k) μ.λ
```

```
UniversalKh[K : ((Knot | Link | TorusKnot)[_Integer, __]), options___] :=
 UniversalKh[K, options] = Module[{khn, result, components, factor},
   CreditMessage[UniversalKh::about];
   If[Length[Skeleton[K]] > 1,
    Print["Warning: UniversalKh is currently *broken* for links.
         It may be a simple matter of dividing the coefficient of
         KhE by (q+q^{-1}), but we haven't identified the bug."];
   ];
   khn = KhN[PD[K], options];
   result = AbsoluteTiming[DecomposeComplex[GradingsList[khn], AllMatrices[khn]]];
   AppendTo[UniversalKhTimingData, {K, result[[1]] /. Second → 1}];
   result[[2]]
  ]
UniversalKh[d_PD, options___] := With[{khn = KhN[d, options]},
  DecomposeComplex[GradingsList[khn], AllMatrices[khn]]
 ]
```

```
DecomposeComplex[{g0_, gradings0_}, {g0_, matrices0_List}] :=
 Module[{g = g0, gradings = gradings0, matrices = matrices0,
   result = 0, matrix, objects, exponents, i, j, k, α, λ, μ, ν},
  While[Length[matrices] > 0, objects = gradings[[1]]; matrix = matrices[[1]];
   While[
    (exponents = DeleteCases[
        Union[(Exponent[#1, T] &) /@ Flatten[MatrixData[matrix]]], -∞]) ≠ {},
    k = First[exponents];
    If[k == 0,
     Print["Found an isomorphism I wasn't expecting!"];
     Print["Result so far: ", result];
     Print["Remaining objects at this height: ", objects];
     Print["Remaining matrices at this height: ", matrix];
     Print["Other gradings: ", gradings];
     Print["Other matrices: ", matrices];
     Return[$Failed]];
    {i, j} = Position[MatrixData[matrix], e_ /; Exponent[e, T] == k, 2, 1][[1]];
    objects = RotateLeft[objects, j - 1];
    gradings[[2]] = RotateLeft[gradings[[2]], i - 1];
    matrix = RotateRows[matrix, i - 1];
    matrix = RotateColumns[matrix, j - 1];
    If[Length[matrices] > 1, matrices[[2]] = RotateColumns[matrices[[2]], i - 1]];
    α = matrix[[1, 1]] / T^k;
    λ = RestColumns[FirstRow[matrix]];
    μ = RestRows[FirstColumn[matrix]];
    ν = RestRows[RestColumns[matrix]];
    matrix = twist[α, k, λ, μ, ν];
    If[Length[matrices] > 1, matrices[[2]] = RestColumns[matrices[[2]]]];
    result += t^(g+1) q^(2 k+objects[[1]]) KhC[k];
    objects = Rest[objects];
    gradings[[2]] = Rest[gradings[[2]]];];
   If[! ZeroMatrixQ[matrix],
    Print["I was expecting the matrix to be zero now."];
    Print["Result so far: ", result];
    Print["Remaining objects at this height: ", objects];
    Print["Remaining matrices at this height: ", matrix];
    Print["Other gradings: ", gradings];
    Print["Other matrices: ", matrices];
    Return[$Failed]];
   result += KhE Plus @@ (t^g q^#1 &) /@ objects;
   matrices = Rest[matrices];
   gradings = Rest[gradings];
   g++;];
  result += KhE Plus @@ (t^g q^#1 &) /@ gradings[[1]];
  result]
```

```
sInvariant[K_] := With[{ukh = UniversalKh[K]},
  If[Length[Position[ukh, KhE]] == 1,
   Replace[ukh /. {_KhC :> 0, KhE → 1}, {q^s_. :> s, 1 → 0}],
   ukh /. {_KhC :> 0, KhE → 1}
  ]
 ]
```

```
α0rules = {KhE → q + q^-1, KhC[1] → t^-1 q^-3 + q^1, KhC[n_] /; n ≥ 2 :> (q + q^-1) (t^-1 q^-2 n + 1)};
```

```
reducedRules = {KhE → q^-1, KhC[n_] :> t^-1 q^-2 n-1 + q^-1};
```

```
KhReduced[K_] := Function[{q, t}, Evaluate[Expand[UniversalKh[K] /. reducedRules]]]
```

```
End[]
```

```
KnotTheory`UniversalKh`Private`
```

```
EndPackage[]
```