KnotTheory`KTtoLinKnot`

This file is a subpackage of the KnotTheory` package, whose home is at http://katlas.math.toronto.edu/

This package provides a compatibility layer between KnotTheory` and LinKnot`. You'll need to separately install LinKnot`, available at http://www.mi.sanu.ac.yu/vismath/linknot/.

```
BeginPackage["KnotTheory`"];
```

```
KnotInput::usage =
  "KnotInput[] opens a window in which you can draw a knot or link by hand.
    Right click and select 'Quit' when you're done. This function requires
    the package LinKnots`, and will only run on Windows machines. Sorry!";
KnotInput::about = "The KnotInput program was written by M. Ochiai,
   C. Nakai, Y. Matsuyama and N. Imafuji and is imported to
   KnotTheory` via the package LinKnot by S. Jablan and R. Sazdanovic"
```

```
DrawKnot::usage =
 "DrawKnot[K_] draws a knot (or link!) K. This function requires the package
   LinKnots`, and will only run on Windows machines. Sorry!"
```

```
LinKnotDirectory::usage =
 "LinKnotDirectory[] contains the path to the LinKnot package. It
   must be set correctly in order for all the (Windows only) MathLink
   components of LinKnot to be usable. It can be overriden by the user."
AllConwayNotations::usage = "AllConwayNotations[n_Integer] gives
   a complete list of knots and links with n crossings"
```

```
ConwayNotation::usage =
  "ConwayNotation[s] represents the knot or link whose
    Conway notation is the string s. ConwayNotation[K], where K is a
    knot or a link with up to 12 crossings, returns ConwayNotation[s],
    where s is a string containing the Conway notation of K.";
ConwayNotation::about =
  "The program ConwayNotation relies on code from
    the LinKnot package by Slavik Jablan and Radmila Sazdanovic.";
```

```
Begin["`KTtoLinKnot`"]
```

```
SetAttributes[SwitchDirectories, HoldAll]
SwitchDirectories[e_] := Module[{currentDir = Directory[],
    kbcOnContextPath = MemberQ[$ContextPath, "KnotsByComputer`"], result},
  SetDirectory[LinKnotDirectory[]];
  If[! kbcOnContextPath, AppendTo[$ContextPath, "KnotsByComputer`"]];
  result = e;
  If[! kbcOnContextPath,
   $ContextPath = DeleteCases[$ContextPath, "KnotsByComputer`"]];
  SetDirectory[currentDir];
  result
 ]
```

```
SetAttributes[EnsurePolyBaseVisible, HoldAll]
EnsurePolyBaseVisible[e_] :=
 Module[{pbOnContextPath = MemberQ[$ContextPath, "PolyBase`"], result},
  If[! pbOnContextPath, AppendTo[$ContextPath, "PolyBase`"]];
  result = e;
  If[! pbOnContextPath, $ContextPath = DeleteCases[$ContextPath, "PolyBase`"]];
  result
 ]
```

```
SetAttributes[EnsureKnotLinkBaseVisible, HoldAll]
EnsureKnotLinkBaseVisible[e_] :=
 Module[{klbOnContextPath = MemberQ[$ContextPath, "KnotLinkBase`"], result},
  If[! klbOnContextPath, AppendTo[$ContextPath, "KnotLinkBase`"]];
  result = e;
  If[! klbOnContextPath, $ContextPath = DeleteCases[$ContextPath, "KnotLinkBase`"]];
  result
 ]
```

```
checkArgs[s_, t_] := ListQ[s] && VectorQ[t, IntegerQ[#] && # ≥ 0 &] && Tr[t] ≤ Length[s]
iteratedTake[s_, t_] /; checkArgs[s, t] :=
 iteratedTake[s, t] = With[{w = FoldList[Plus, 0, t]},
   Map[Take[s, #] &, Transpose[{Drop[w, -1] + 1, Rest[w]}]]]
```

```
fContoKTGauss[Ul_String] := Module[{mm, nn, ss, vv, i},
  SwitchDirectories[
   EnsurePolyBaseVisible[
    mm = LinKnots`fGaussExtSigns[Ul];
    nn = LinKnots`fGaussExtSigns[StringReplace[Ul, "-" → ""]];
   ];
   nn = Map[Sign, Flatten[mm]] * Map[Sign, Flatten[nn]];
   vv = Table[nn[[i]] * (-1)^i, {i, Length[nn]}] * Abs[Flatten[mm]];
   ss = Map[Length, mm];
   mm = If[MemberQ[ss, 0], {vv}, iteratedTake[vv, ss]];
   GaussCode @@ If[Length[mm] > 1, mm, mm[[1]]]
  ]
 ]
```

```
PD[cn_ConwayNotation] := PD[GaussCode[cn]]
```

```
InstallLinKnots::failed =
 "The function \"`1`\" requires the LinKnot package, which is not
   distributed as part of KnotTheory. I couldn't seem to load it; try
   downloading it from http://www.mi.sanu.ac.yu/vismath/linknot/,
   and adding the appropriate directory to the $Path."
```

```
InstallLinKnots[symbol_] := Module[{oldContextPath = $ContextPath},
  (*Try to load LinKnots`*)
Needs["LinKnots`"];
   (*If it failed,
   it won't be on the $ContextPath. Try to give a useful error message.*) ×
If[! MemberQ[$ContextPath, "LinKnots`"],
    Message[InstallLinKnots::failed, symbol];
    False,
    LinKnotDirectory[] = DirectoryName[
        File /. Flatten[FileInformation[ToFileName[#, "LinKnots.m"]] & /@ $Path]];
     (*Now clean up the $ContextPath again, removing as much as possible.*) ×
$ContextPath = oldContextPath;
    (InstallLinKnots[s_] := True);
    True
   ]
 ]
```

```
GaussCode[HoldPattern[ConwayNotation[ss_String]]] := Module[{},
  If[InstallLinKnots[ConwayNotation],
   (GaussCode[HoldPattern[ConwayNotation[ss0_String]]] := fContoKTGauss[ss0]);
   CreditMessage["Conway notation (and pdata) to Gauss code
      conversion was written by Radmila Sazdanovic in 2003-2006."];
   GaussCode[ConwayNotation[ss]],
   $Failed
  ]
 ]
```

```
ConwayNotation[x : Except[_String]] := Module[{},
  If[InstallLinKnots[ConwayNotation],
   (* up to 10 crossings D. Rolfsen from Classical notation *)
   ConwayNotation[Knot[n_, k_]] /; (n ≤ 10) :=
    ConwayNotation[LinKnots`fClassicToCon[NameString[Knot[n, k]]]];
   (* up to 12 crossings form ...*)
   ConwayNotation[x0 : Except[_String]] :=
    ConwayNotation[SwitchDirectories[LinKnots`fFindCon[DTtoPData[DTCode[x0]]]]];
   ConwayNotation[x],
   $Failed
  ]
 ]
```

```
(******red are functions from LinKnot*********)
```

```
KnDowToKTGauss[Ul_List] := Module[{ss, gg, sc, i},
  ss = LinKnots`fSignsKL[Abs[Ul]][[2]];
  gg = Map[Sort, Table[{2 i - 1, Abs[Ul[[2, i]]]}, {i, Length[Ul[[2]]]}]];
  sc =
   Flatten[Complement[Table[If[ss[[i]] < 0, gg[[i]], {}], {i, Length[ss]}], {{}}]];
  gg = Map[Last, Sort[Flatten[Table[{{gg[[i, 1]], i}, {gg[[i, 2]], i}},
       {i, Length[gg]}], 1]]];
  gg = Table[gg[[i]] * (-1)^i, {i, Length[gg]}];
  gg = Table[If[MemberQ[sc, i], -gg[[i]], gg[[i]]], {i, Length[gg]}];
  GaussCode @@ If[Length[Ul[[1]]] == 1, gg, iteratedTake[gg, 2 Ul[[1]]]]
 ]
```

```
DowkerToKTGauss[Ul_List] := Module[{ss, ss1, i}, ss = LinKnots`fSignsKL[Abs[Ul]];
  ss1 = Map[Sign, Ul[[2]]] * Map[Sign, ss[[2]]];
  ss = KnDowToKTGauss[{Ul[[1]], ss1 * Ul[[2]]}]]
(*Pdata to Knot theory GaussCode*)
PdataToKTGauss[Ul_List] := Module[{},
  CreditMessage["Conway notation (and pdata) to Gauss code
     conversion was written by Radmila Sazdanovic in 2003-2006."];
  DowkerToKTGauss[LinKnots`fDowfromPD[Ul]]
 ]
```

```
(*DT to Pdata via KnotscapeDow=PD*)
DTtoPData[HoldPattern[DTCode[d__List]]] :=
 LinKnots`fPDataFromDow[{Length /@ {d}, Join[d]}]
DTtoPData[HoldPattern[DTCode[n__Integer]]] :=
 LinKnots`fPDataFromDow[{{Length[{n}]}, {n}}]
```

```
KnotInput[] := Module[{pdata},
  If[InstallLinKnots[KnotInput],
   CreditMessage["Graphical knot input was written
      by M. Ochiai, C. Nakai, Y. Matsuyama and N. Imafuji."];
   SwitchDirectories[PdataToKTGauss[KnotsByComputer`GetPdatabyTracking[]]],
   $Failed]
 ]
```

```
DrawKnot[k_] := Module[{pdata},
  If[InstallLinKnots[DrawKnot],
   CreditMessage["Graphical knot output was written by ???."];
   SwitchDirectories[
    pdata = DTtoPData[DTCode[k]];
    KnotsByComputer`ShowKnotfromPdata[pdata]
   ], $Failed]]
```

```
AllConwayNotations[n : (1 | 2 | 3 | 4 | 5)] := AllConwayNotations[n, Alternating]
AllConwayNotations[n_Integer] /; n ≥ 1 :=
 AllConwayNotations[n, Alternating] ~Join~ AllConwayNotations[n, NonAlternating]
AllConwayNotations[n_Integer, Alternating] /; n ≥ 1 :=
 (InstallLinKnots[AllConwayNotations];
  ConwayNotation /@ ToExpression["KnotLinkBase`a" <> ToString[n]])
AllConwayNotations[n_Integer, NonAlternating] /; n ≥ 1 :=
 (InstallLinKnots[AllConwayNotations];
  ConwayNotation /@ ToExpression["KnotLinkBase`n" <> ToString[n]])
```

EndPackage

```
End[]
```

```
EndPackage[]
```