Pensieve header: The Drinfel'd-Kohno algebra and associators; now merged into FreeLie` at pensieve://Projects/WKO4/.

## Prolog

```
BeginPackage["FreeLie`"];
Print["DrinfeldKohno` in FreeLie` implements / extends ",
  Sort@{DK, DKS, Morphism, t, σ},
  "."];
Begin["`Private`"];
```

```
DrinfeldKohno` in FreeLie` implements / extends {DK, DKS, Morphism, t, σ}.
```

## DK (Drinfel'd-Kohno Elements)

```
DK[_, 0] = 0;
DK /: DK[k_, x1_] + DK[k_, x2_] := DK[k, x1 + x2];
DK /: c_ * DK[k_, x_] := DK[k, Expand[c x]];
DK /: b[DK[k1_, c_ * x1_LW], DK[k2_, x2_]] :=
  Expand[c b[DK[k1, x1], DK[k2, x2]]];
DK /: b[DK[k1_, x1_LW], DK[k2_, c_ * x2_LW]] := Expand[c b[DK[k1, x1], DK[k2, x2]]];
DK /: b[DK[k1_, x1_Plus], DK[k2_, x2_]] := b[DK[k1, #], DK[k2, x2]] & /@ x1;
DK /: b[DK[k1_, x1_], DK[k2_, x2_Plus]] := b[DK[k1, x1], DK[k2, #]] & /@ x2;
DK /: b[DK[k_, x1_], DK[k_, x2_]] := DK[k, b[x1, x2]];
DK /: b[DK[k1_, x1_], DK[k2_, x2_]] /; k1 > k2 :=
  b[DK[k2, Expand[-x2]], DK[k1, x1]];
DK /: b[DK[k1_, LW@i1_], DK[k2_, LW@i2_]] /; k1 < k2 :=
  b[DK[k1, LW@i1], DK[k2, LW@i2]] = Which[
    i1 == i2, DK[k2, -b[LW@k1, LW@i2]],
    k1 == i2, DK[k2, -b[LW@i1, LW@i2]],
    True, 0
   ];
DK /: b[DK[k1_, w1_LW], DK[k2_, w2_LW]] /; Deg[w1] > 1 :=
  b[DK[k1, w1], DK[k2, w2]] = Module[{x, y},
    {x, y} = LyndonFactorization[w1];
    b[b[DK[k1, x], DK[k2, w2]], DK[k1, y]] +
     b[DK[k1, x], b[DK[k1, y], DK[k2, w2]]]
   ];
DK /: b[DK[k1_, w1_LW], DK[k2_, w2_LW]] /; Deg[w2] > 1 :=
  b[DK[k1, w1], DK[k2, w2]] = Module[{x, y},
    {x, y} = LyndonFactorization[w2];
    b[b[DK[k1, w1], DK[k2, x]], DK[k2, y]] +
     b[DK[k2, x], b[DK[k1, w1], DK[k2, y]]]
   ];
t[i_, j_] := DK[Max[i, j], LW@Min[i, j]];
TopBracketForm[DK[k_, x_]] :=
  x // LieMorphism[Table[LW@i → LW@t_{10 i+k}, {i, k - 1}]];
(*Format[dk_DK] := TopBracketForm[dk];*)
```

## σ

```
σ[lft___, i_Integer, rgt___] := σ[lft, IntegerDigits[i], rgt];
_σ[0] = 0;
x_Plus // s_σ := s[#] & /@ x;
DK[k_, x_Plus] // s_σ := s[DK[k, #]] & /@ x;
DK[k_, c_ * w_LW] // s_σ := Expand[c * s[DK[k, w]]];
DK[k_, LW@i_] // s_σ := Sum[t[α, β], {α, s[[i]]}, {β, s[[k]]}];
DK[k_, w_LW] // s_σ := b @@ (s[DK[k, #]] & /@ LyndonFactorization[w]);
```

# DKSeries

```
DKSeries[ser_Symbol][{dd_Integer}] := TopBracketForm[Append[
    DKS @@ Table[
      ser[d] /.
       DK[k_, x_] :> (x // LieMorphism[Table[LW@i → LW@t_{10 i+k}, {i, k - 1}]])@d,
      {d, dd}
     ],
    "..."]];
Format[dks_DKSeries, StandardForm] := dks[{$SeriesShowDegree}];
DKSeries[ser_Symbol][e___] := ser[e];
b[dk1_DKSeries, dk2_DKSeries] := b[dk1, dk2] = New[DKSeries[ser],
    ser[d_Integer] := ser[d] = Sum[b[dk1[j], dk2[d - j]], {j, 1, d - 1}]
   ];
DKS[dks_DKSeries] := dks;
DKS[expr_] := DKS[expr] = New[DKSeries[ser],
    ser[d_Integer] := ser[d] = Expand[expr /. w_LW /; Deg[w] ≠ d → 0]
   ];
DKS[k_Integer, coefs_] := New[DKSeries[ser],
   ser[setter] = Null;
   ser[d_Integer, UndeterminedCoefficients] := Cases[
     Join @@ Table[
       coefs[j, Sequence @@ #] & /@ AllLyndonWords[d, Range[j - 1]], {j, 2, k}],
     _coefs];
   ser[d_Integer] := If[ser[setter] =!= Null,
     ser[setter][d]; ser[d],
     Sum[
      Plus @@
       ((DK[j, #] * coefs[j, Sequence @@ #]) & /@ AllLyndonWords[d, Range[j - 1]]),
      {j, 2, k}
     ]
    ];
  ];
s1_DKSeries ≡ s2_DKSeries := New[BooleanSequence[bs],
   bs[0] = True;
   bs[d_Integer] :=
    bs[d - 1] && Replace[s1[d] - s2[d] /. DK[_, x_] :> x == 0, 0 → True];
  ];
dks_DKSeries // s_σ := dks // s = New[DKSeries[ser],
    ser[d_Integer] := ser[d] = dks[d] // s
   ];
```

## Morphisms LS → DKS

```
Morphism[mor_][es___] := mor[es];
Morphism[LS, DKS, rules__Rule] := Morphism[LS, DKS, {rules}];
Morphism[LS, DKS, rules_List] := New[Morphism[mor],
    mor[Support] = First /@ rules;
    (mor[w_LW] /; Deg[w] == 1) := (mor[w] = w /. rules);
    mor[w_LW] := (mor[w] = b @@ (mor /@ LyndonFactorization[w]));
    mor[expr_][d_] := Expand[expr /. w_LW :> mor[w][d]];
    mor[ls_LieSeries] := mor[ls] = New[DKSeries[ser],
        ser[d_] := ser[d] = Sum[mor[ls[k]][d], {k, 1, d}]];
  ];
BCH[x_DKSeries, y_DKSeries] :=
  BCH[LW@"x", LW@"y"] // Morphism[LS, DKS, LW@"x" → x, LW@"y" → y];
DKSeries /: x_DKSeries ** y_DKSeries := BCH[x, y];
```

## Epilog

```
End[]; EndPackage[];
```