Scatter and Glow - Program

By Dror Bar-Natan with Kerene Chu, Zsuzsanna Dancso, Peter Lee and Louis Leung.
December 2008.

Project Goals

Done

- Verify R3, OC, Locallity, R2, CC on scatter level.
- Same, on glow level.
- The scatter of an arbitrary exponential.
- Find an explicit BCH formula.

To do

- Allow multiple hairstyles, implement pertubative hair, allow for mixing hairstyles.
- Split scattering into "light" and "heavy".
- Solve R4 for F at the scatter level.
- Implement strand operations: deletion, addition, and doubling.
- Verify the pentagon.
- Check the Hexagons.
- Solve the \theta-R-F equation.
- Verify the Hexagons.
- The glow of an arbitrary exponential.
- Recover the Alexander polynomial of all knots.
- Recover the multi-variable Alexander polynomial of all links.
- Solve for F at the glow level.
- Pentagon and Hexagons at glow level.
- Recover the Lieberum formulas.

Conventions

- a_{ij} = `Ar[i,j]` is an arrow going from i to j.
- Y_{ijk} = `Y[i,j,k]` $:= [a_{ik}, a_{jk}] = a_{ik}\, a_{jk} - a_{jk}\, a_{ik} = \mathrm{Ad}(a_{ik})(a_{jk}) = -[a_{ij}, a_{jk}]$.
- $x_l\, Y_{ijk} := [a_{lk}, Y_{ijk}]$.
- IHX: $x_i\, Y_{jkl} + x_j\, Y_{kil} + x_k\, Y_{ijl} = 0$.

Program

Hair Stylling

```
If[H[0] =!= 0, H[h_] = AH[h]];
CanonicalForm[0] = 0;
```

AH is "Analytic Hair"

```
CanonicalForm[AH[expr_]] := AH[Factor[expr]];
AH[h_AH] := h;
AH[0] = 0;
AH /: AH[h1_] ** AH[h2_] := AH[h1 ** h2];
AH /: AH[1] ** a_Ar := a;
AH /: AH[h1_] ** Y[ijk__, AH[h2_]] := Y[ijk, AH[h1 ** h2]];
AH /: c_ ** AH[h_] := AH[c ** h];
AH /: AH[h1_] + AH[h2_] := CanonicalForm[AH[h1 + h2]];
AH /: AH[h_] ~Mod~ x[i_] := AH[Limit[h, x[i] -> 0]];
AH /: Coefficient[AH[h_], eps_, 1] := AH[Coefficient[Expand[h], eps, 1]];
AH /: MatrixExp[AH, mat_] := CanonicalForm[Map[AH, MatrixExp[mat], {2}]];
```

- **PH is "Perturbative (Short) Hair"**

```
If[Head[$CutoffDegree] =!= Integer, $CutoffDegree = 5];
PH[h_PH] := h;
PH[0] = 0;
PH[Literal[SeriesData[z, 0, _, n_, _, _]]] /; n ≥ $CutoffDegree := 0;
CanonicalForm[PH[expr_]] := PH[MapAt[Expand, expr + O[z]^$CutoffDegree, 3]];
PH /: PH[h1_] * PH[h2_] := CanonicalForm[PH[h1 * h2]];
PH /: PH[1] * a_Ar := a;
PH /: PH[Literal[SeriesData[z, 0, {1, 0 ...}, 0, n_, _]]] * a_Ar /; n ≥ $CutoffDegree := a;
PH /: PH[h1_] * Y[ijk__, PH[h2_]] := Y[ijk, PH[h1 * h2]];
PH /: c_ * PH[h_] := CanonicalForm[PH[(c /. x[i_] :> z * x[i]) * h]];
PH /: PH[h1_] + PH[h2_] := PH[h1 + h2];
PH /: PH[h_] ~ Mod ~ x[i_] := PH[h /. x[i] → 0];
PH /: Coefficient[PH[h_], eps_, 1] := CanonicalForm[
   PH[Coefficient[h, eps, 1]]
  ];
PH /: MatrixExp[PH, mat_] :=
  Map[PH, Sum[MatrixPower[mat, k] / k!, {k, 0, $CutoffDegree}], {2}];
DeclareSeries[f_[vars___], deg_] := (
   PH[f] = CanonicalForm[PH[
      (f[vars] /. Thread[{vars} → z * {vars}]) + O[z]^deg
     ]];
   PH[f] = PH[f] /. Derivative[ds___][f][0 ...] :> f[ds];
   Cases[PH[f], f[___], Infinity]
  );
DeclareSeries[f_] := DeclareSeries[f, $CutoffDegree];
PHSolve[eqs_, etc___] := Module[{vars},
   vars = Union[Cases[eqs, x[i_], Infinity]];
   If[vars === {}, Solve[eqs, etc],
    Solve[
     Series[Normal[eqs] /. z → 1, Sequence @@ ({#, 0, $CutoffDegree} & /@ vars)],
     etc
    ]
   ]
  ];
```

- **If all else fails ...**

```
CanonicalForm[expr_] := expr /. {
    h_AH :> CanonicalForm[h],
    h_PH :> CanonicalForm[h]
   };
```

Basic Actions with Primitives

AS, IHX

```
ReducePrimitives[prims_] := Module[{l, h0, h1}, prims
    //. {
     (* Anti-symmetry, linearity *)
     Y[i_, i_, __] → 0,
     Y[i_, j_, k_, h_] /; i > j :→ Y[j, i, k, -h],
     c_ * Y[i_, j_, k_, h_] :→ Y[i, j, k, c * h],
     Y[i_, j_, k_, h1_] + Y[i_, j_, k_, h2_] :→ Y[i, j, k, h1 + h2],
     (* IHX *)
     Y[i_, j_, k_, h_] /; ! FreeQ[h, x[l_] /; l < i] :→ (
       l = Min[Cases[{h}, x[l_] :→ l, Infinity]];
       h0 = h ~ Mod ~ x[l];
       h1 = (h - h0) / x[l];
       Y[i, j, k, h0]
        - Y[j, l, k, h1 * x[i]] - Y[l, i, k, h1 * x[j]]
      )
    }
   /. Y[__, 0] → 0
  ];
```

■ Basic Scattering Code

```
(prims_ // S[srules___Rule]) := ReducePrimitives[prims
      /. {
        Ar[i_, j_] :→ Distribute[Ar[Ar[i, 0], Ar[0, j]] /. {srules}],
        Y[i_, j_, k_, h_] :→ Distribute[Y[
            Ar[i, 0] , Ar[j, 0], Ar[0, k], h
          ] /. {srules}]
      }
     /. {Ar[i_, 0] :→ i, Ar[0, j_] :→ j}
    /. {
     Ar[Y[i_, j_, 0, h_], k_] :→ Y[i, j, k, h],
     Y[_Y, _Y, __] :→ 0,
     Y[i_Integer, Y[j_, k_, 0, h_], l_, h1_] :→ Y[j, k, l, x[i] * h * h1],
     Y[Y[j_, k_, 0, h_], i_Integer, l_, h1_] :→ Y[j, k, l, -x[i] * h * h1]
    }
   /. {
    Ar[i_, Y[0, j_, k_, h_]] :→ Y[i, j, k, h],
    Ar[i_, Y[j_, 0, k_, h_]] :→ Y[j, i, k, h],
    Y[i_, j_, Y[0, k_, l_, h_], h1_] :→ Y[i, j, l, -x[k] * h * h1],
    Y[i_, j_, Y[k_, 0, l_, h_], h1_] :→ Y[i, j, l, x[k] * h * h1]
   }
  ];

(prims_ // ts_TS) := prims // (S @@ ts);

S /: S[sr1___Rule] ** S[sr2___Rule] := Module[{elements, rule},
  elements = Union[First /@ {sr1, sr2}];
  S @@ DeleteCases[
     (* The introduction of the symbol "rule" is an ugly hack *)
     Thread[rule[elements, elements // S[sr1] // S[sr2]]],
     rule[e_, e_]
    ] /. rule → Rule
 ]

S[sigma[i_, j_], more___] := S[
    Ar[0, j] → Ar[0, j] + Y[0, i, j, H[-(Exp[-x[i]] - 1) / x[i]]],
    Ar[0, i] → Ar[0, i] + Y[0, i, j, H[(Exp[-x[i]] - 1) / x[i]]],
    Ar[j, 0] → Ar[j, 0] + Y[i, j, 0, H[(Exp[x[i]] - 1) / x[i]]]
   ] ** S[more];
S[sigbar[i_, j_], more___] := S[
    Ar[0, j] → Ar[0, j] + Y[0, i, j, H[-(Exp[x[i]] - 1) / x[i]]],
    Ar[0, i] → Ar[0, i] + Y[0, i, j, H[(Exp[x[i]] - 1) / x[i]]],
    Ar[j, 0] → Ar[j, 0] + Y[i, j, 0, H[(Exp[-x[i]] - 1) / x[i]]]
   ] ** S[more];
```

■ Basic Glow Code

```
SnG[] := SnG[S[], 0];
SnG /: SnG[s1_S, g1_] ** SnG[s2_S, g2_] :=
  SnG[s1 ** s2, ReducePrimitives[(g1 // s2) + g2]];

SnG[sigma[i_, j_], more___] := SnG[S[sigma[i, j]], Ar[i, j]] ** SnG[more];
SnG[sigbar[i_, j_], more___] := SnG[S[sigbar[i, j]], -Ar[i, j]] ** SnG[more];
```

■ Derivations Code

```
Der[drules___Rule][expr_] := Module[{s, eps},
   s = (S[drules] /. (a_ → b_) :→ ReducePrimitives[a → Expand[a + eps * b]]);
   (expr // s) /. {
      _Ar → 0,
      Y[i_, j_, k_, h_] :→ Y[i, j, k, Coefficient[h, eps, 1]]
     } /. Y[__, 0] → 0
  ];

Der[a_Plus] := Der /@ a;
Der /: a_ * Der[drules___Rule] := Der[drules] /. (b_ → c_) :→ (b → a * c);
Der /: Der[dr1___Rule] + Der[dr2___Rule] := Module[{elements},
   elements = Union[First /@ {dr1, dr2}];
   Der @@ DeleteCases[
     Thread[Rule[elements,
        (elements /. {dr1, _Ar → 0}) + (elements /. {dr2, _Ar → 0})
       ]],
     _ → 0
    ]
  ];

Der[a_. * Ar[i_, i_]] := Der[];
Der[a_. * Ar[i_, j_]] /; i ≠ j := Der[
   Ar[j, 0] → Y[i, j, 0, H[-a]],
   Ar[0, i] → Y[0, i, j, H[a]],
   Ar[0, j] → Y[i, 0, j, H[a]]
  ];
Der[a_. * Y[i_, j_, k_, h_]] /; i ≠ j ≠ k := Module[{d1, d2, elements},
   d1 = Der[Ar[i, k]];
   d2 = Der[Ar[j, k]];
   elements = Union[First /@ List @@ d1, First /@ List @@ d2];
   DeleteCases[
    Der @@ Thread[elements → ReducePrimitives[d1@d2@elements - d2@d1@elements]] /.
      Y[ijk__, h1_] :→ Y[ijk, a * h * h1],
    _ → 0
   ]
  ];
Der[_. * Y[i_, i_, __]] = Der[];
Der[a_. * Y[i_, j_, j_, h_]] /; i ≠ j := Der[
   Ar[j, 0] → Y[i, j, 0, x[j] * a * h],
   Ar[0, i] → Y[i, 0, j, x[j] * a * h],
   Ar[0, j] → Y[i, 0, j, -x[j] * a * h]
  ];
Der[a_. * Y[j_, i_, j_, h_]] /; i ≠ j := Der[
   Ar[j, 0] → Y[i, j, 0, -x[j] * a * h],
   Ar[0, i] → Y[i, 0, j, -x[j] * a * h],
   Ar[0, j] → Y[i, 0, j, x[j] * a * h]
  ];
```

Scattering by an Arbitrary Exponential

```
Der /: Exp[Der[], _] = {}; Der /: Exp[Der[drules___Rule], on_] := Module[
  {k0, ins, outs, k, newout, hs, zero, e, mat, expmat},
  If[Head[on] === List,
   k0 = Length[ins = on],
   ins = First /@ {drules};
   If[on === Tails, ins = Cases[ins, Ar[_, 0] | Y[_, _, 0, ___]]];
   k0 = Length[ins]
  ];
  outs = {};
  For[k = 1, k ≤ Length[ins], ++k,
   AppendTo[outs, newout = Der[drules][ins[[k]]]];
   ins = ins ~Join~ Complement[
      Union[Cases[{newout}, Y[ijk__, hs_[_]] :> Y[ijk, hs[1]], Infinity]],
      ins
     ]
  ];
  --k;
  hs = First[Cases[outs, Y[__, hs_[_]] :> hs, Infinity, 1]];
  zero = Table[0, {k}];
  e[{{i_}}] := ReplacePart[zero, 1, i];
  mat = Replace[
    outs /. Y[ijk__, hs[h_]] :> -h e[Position[ins, Y[ijk, hs[1]]]],
    0 → zero,
    {1}
   ];
  expmat = MatrixExp[hs, mat];
  Sort[Thread[
    Take[ins, k0] →
     ReducePrimitives[Take[expmat.ins, k0]]
   ]]
 ];
```

```
S[Exp[d_Der]] := S @@ Exp[d, All];
S[Exp[prims_]] := S[Exp[Der[prims]]];
```

```
TS[Exp[d_Der]] := TS @@ Exp[d, Tails];
TS[Exp[prims_]] := TS[Exp[Der[prims]]];
```

■ Extracting Hair

```
Y /: Coefficient[expr_, Y[i_, j_, k_]] := expr /. {
   _Ar → 0,
   Y[i, j, k, hs_[h_]] :> h,
   _Y → 0
  }
```

■ Strands Operations

```
(*
SOp[op___Rule][S[srules___Rule]] := Module[{op1},
  op1={op} /. (i_ → j_Integer) ⧴ (i → {j});
  expr /. {
    Ar[i_,j_] ⧴ (Ar[i,j] /. op1),
    Y[i_,j_,k_,h_] ⧴ Y[i/.op1, j/.op1, k/.op1, h/.
    *)
```