

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, Locality, R2, CC on scatter level.
- Same, on glow level.
- The scatter of an arbitrary exponential.
- Find an explicit BCH formula.

■ To do

- Implement Hairstyles!*
- Implement strand operations: deletion, addition, and doubling.
 - Solve R4 for F at the scatter level.
 - Verify the pentagon.
 - Check the Hexagons.
 - Solve the θ -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

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

Highlighter marks hair actions

Program

- Bring Hair to Canonical Form

```
SH[expr_] := Factor[expr]
```

- 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, SH[-h]],
    c_*Y[i_, j_, k_, h_] ⇒ Y[i, j, k, SH[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_] ⇒ 1, Infinity]];
      h0 = Limit[h, x[l] → 0];
      h1 = SH[(h - h0) / x[l]];
      Y[i, j, k, h0]
      - Y[j, l, k, SH[h1 x[i]]] - Y[l, i, k, SH[h1 * x[j]]]
    )
  }
  /. Y[i_, j_, k_, h_] ⇒ Y[i, j, k, SH[h]]
  /. 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]
}
];

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, -(Exp[-x[i]] - 1) / x[i]],
  Ar[0, i] -> Ar[0, i] + Y[0, i, j, (Exp[-x[i]] - 1) / x[i]],
  Ar[j, 0] -> Ar[j, 0] + Y[i, j, 0, (Exp[x[i]] - 1) / x[i]]
] ** S[more];

S[sigbar[i_, j_], more___] := S[
  Ar[0, j] -> Ar[0, j] + Y[0, i, j, -(Exp[x[i]] - 1) / x[i]],
  Ar[0, i] -> Ar[0, i] + Y[0, i, j, (Exp[x[i]] - 1) / x[i]],
  Ar[j, 0] -> Ar[j, 0] + Y[i, j, 0, (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, -a],
  Ar[0, i] -> Y[0, i, j, a],
  Ar[0, j] -> Y[i, 0, j, 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, SH[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

```
S[Exp[Der[drules___Rule]]] := Module[
  {k0, ins, outs, k, newout, zero, e, mat},
  k0 = Length[ins = First /@ {drules}];
  outs = {};
  For[k = 1, k ≤ Length[ins], ++k,
    AppendTo[outs, newout = Der[drules][ins[[k]]]];
    ins = ins ~Join~ Complement[
      Union[Cases[{newout}, Y[ijk___, _] => Y[ijk, 1], Infinity]],
      ins
    ]
  ];
  --k;
  zero = Table[0, {k}];
  e[{{i_}}] := ReplacePart[zero, 1, i];
  mat = Replace[
    outs /. Y[ijk___, h_] => -h e[Position[ins, Y[ijk, 1]]],
    0 → zero,
    {1}
  ];
  S @@ Sort[Thread[
    Take[ins, k0] →
    ReducePrimitives[Take[SH[MatrixExp[mat]].ins, k0]]
  ]];
  S[Exp[prims_]] := S[Exp[Der[prims]]];
```

Done line.

■ Extracting Hair

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

■ Strands Operations

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