CS-SL2Invariant on 180623

June 24, 2018 9:14 AM

"Categorify" $\mathbb{E}$! [// has "automatic σ's"]

Define $\sigma_{i\to j}$

Deprecate "Bind"?

Cheat Sheet sl_2-Invariant (the sl_2 portfolio and invariant)

http://drorbn.net/AcademicPensieve/Projects/SL2Invariant/
modified 24/6/18, 09:13

Internal Utilities

Canonical Form:

```
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[ℰ_] :=
  PP_CF@ExpandDenominator@ExpandNumerator@Together[
    Expand[ℰ] //. e^x_ e^y_ :→ e^(x+y) /. e^x_ :→ e^CF[x]];
```

The Kronecker δ:

```
Kδ /: Kδ_{i_,j_} := If[i === j, 1, 0];
```

Equality, multiplication, and degree-adjustment of perturbed Gaussians; $\mathbb{E}[L, Q, P]$ stands for $e^{L+Q}P$:

```
𝔼 /: 𝔼[L1_, Q1_, P1_] ≡ 𝔼[L2_, Q2_, P2_] :=
  CF[L1 == L2] ∧ CF[Q1 == Q2] ∧ CF[Normal[P1 - P2] == 0];
𝔼 /: 𝔼[L1_, Q1_, P1_] 𝔼[L2_, Q2_, P2_] :=
  𝔼[L1 + L2, Q1 + Q2, P1 * P2];
𝔼[L_, Q_, P_]_{$k_} := 𝔼[L, Q, Series[Normal@P, {ϵ, 0, $k}]];
```

Zip and Bind

Variables and their duals:

```
{t*, b*, y*, a*, x*, z*} = {τ, β, η, α, ξ, ζ};
{τ*, β*, η*, α*, ξ*, ζ*} = {t, b, y, a, x, z};
(u_{i_})* := (u*)_i;
```

Finite Zips:

```
collect[sd_SeriesData, ζ_] :=
  MapAt[collect[#, ζ] &, sd, 3];
collect[ℰ_, ζ_] := PP_Collect@Collect[ℰ, ζ];
Zip_{}[P_] := P; Zip_{{ζ_, ζs___}}[P_] := PP_Zip[
  (collect[P // Zip_{{ζs}}, ζ] /. f_. ζ^d_. :→ ∂_{{ζ*,d}} f) /. ζ* → 0]
```

QZip implements the "Q-level zips" on $\mathbb{E}(L, Q, P) = Pe^{L+Q}$. Such zips regard the L variables as scalars.

```
QZip_{ζs_List,simp_}@𝔼[L_, Q_, P_] :=
  PP_QZip@Module[{ζ, z, zs, c, ys, ηs, qt, zrule, Q1, Q2},
    zs = Table[ζ*, {ζ, ζs}];
    c = Q /. Alternatives @@ (ζs ⋃ zs) → 0;
    ys = Table[∂_ζ (Q /. Alternatives @@ zs → 0), {ζ, ζs}];
    ηs = Table[∂_z (Q /. Alternatives @@ ζs → 0), {z, zs}];
    qt = Inverse@Table[Kδ_{z,ζ*} - ∂_{z,ζ}Q, {ζ, ζs}, {z, zs}];
    zrule = Thread[zs → qt.(zs + ys)];
    Q2 = (Q1 = c + ηs.zs /. zrule) /. Alternatives @@ zs → 0;
    simp /@ 𝔼[L, Q2, Det[qt] e^(-Q2) Zip_{ζs}[e^Q1 (P /. zrule)]]];
QZip_{ζs_List} := QZip_{ζs,CF};
```

Upper to lower and lower to Upper:

```
U2l = {B_{i_}^{p_.} → e^(-p ħ γ b_i), B^{p_.} → e^(-p ħ γ b), T_{i_}^{p_.} → e^(p ħ t_i),
   T^{p_.} → e^(p ħ t), 𝒜_{i_}^{p_.} → e^(p γ α_i), 𝒜^{p_.} → e^(p γ α)};
l2U = {e^(c_. b_{i_} + d_.) :→ B_i^(-c/(ħ γ)) e^d, e^(c_. b + d_.) :→ B^(-c/(ħ γ)) e^d,
   e^(c_. t_{i_} + d_.) :→ T_i^(c/ħ) e^d, e^(c_. t + d_.) :→ T^(c/ħ) e^d,
   e^(c_. α_{i_} + d_.) :→ 𝒜_i^(c/γ) e^d, e^(c_. α + d_.) :→ 𝒜^(c/γ) e^d,
   e^ℰ_ :→ e^(Expand@ℰ)};
```

LZip implements the "L-level zips" on $\mathbb{E}(L, Q, P) = Pe^{L+Q}$. Such zips regard all of Pe^Q as a single "P". Here the z's are b and α and the ζ's are β and a.

```
LZip_{ζs_List,simp_}@𝔼[L_, Q_, P_] :=
  PP_LZip@Module[{ζ, z, zs, c, ys, ηs, lt, zrule, L1, L2,
     Q1, Q2},
    zs = Table[ζ*, {ζ, ζs}];
    c = L /. Alternatives @@ (ζs ⋃ zs) → 0;
    ys = Table[∂_ζ (L /. Alternatives @@ zs → 0), {ζ, ζs}];
    ηs = Table[∂_z (L /. Alternatives @@ ζs → 0), {z, zs}];
    lt = Inverse@Table[Kδ_{z,ζ*} - ∂_{z,ζ}L, {ζ, ζs}, {z, zs}];
    zrule = Thread[zs → lt.(zs + ys)];
    L2 = (L1 = c + ηs.zs /. zrule) /. Alternatives @@ zs → 0;
    Q2 = (Q1 = Q /. U2l /. zrule) /. Alternatives @@ zs → 0;
    simp /@
      𝔼[L2, Q2, Det[lt] e^(-L2-Q2)
        Zip_{ζs}[e^(L1+Q1) (P /. U2l /. zrule)]] //. l2U];
LZip_{ζs_List} := LZip_{ζs,CF};
Bind_{}[L_, R_] := L R;
Bind_{{is__}}[L_𝔼, R_𝔼] := PP_Bind@Module[{n},
    Times[
      L /. Table[(v : b | B | t | T | a | x | y)_i → v_{n@i},
        {i, {is}}],
      R /. Table[(v : β | τ | α | 𝒜 | ξ | η)_i → v_{n@i}, {i, {is}}]
    ] // LZip_{Flatten@Table[{β_{n@i}, τ_{n@i}, a_{n@i}}, {i, {is}}]} //
    QZip_{Flatten@Table[{ξ_{n@i}, y_{n@i}}, {i, {is}}]}];
B_{l_List}[L_, R_] := Bind_l[L, R];
B_{is___}[L_, R_] := Bind_{{is}}[L, R];
```

"Define" code

Define[lhs = rhs, ...] defines the lhs to be rhs, except that rhs is computed only once for each value of $k. Fancy Mathematica not for the faint of heart. Most readers should ignore.

```
SetAttributes[Define, HoldAll];
Define[def_, defs__] := (Define[def]; Define[defs];);
Define[op_{is__} = ℰ_] :=
 Module[{SD, ii, jj, kk, isp, nis, nisp, sis},
  Block[{i, j, k},
   ReleaseHold[Hold[
     SD[op_{nisp,$k_Integer}, PP_{Boot@$k}@Block[{i, j, k}, op_{isp,$k} = ℰ;
        op_{nis,$k}]];
     SD[op_{isp}, op_{{is},$k}]; SD[op_{sis__}, op_{{sis}}];
    ] /. {SD → SetDelayed,
     isp → {is} /. {i → i_, j → j_, k → k_},
     nis → {is} /. {i → ii, j → jj, k → kk},
     nisp → {is} /. {i → ii_, j → jj_, k → kk_}
    }] ]]
```

Booting Up

```
$k = 2; ħ = γ = 1;
Define[am_{i,j→k} = 𝔼[(α_i + α_j) a_k, (e^(-γ α_j) ξ_i + ξ_j) x_k, 1]_{$k},
 bm_{i,j→k} = 𝔼[(β_i + β_j) b_k, (η_i + η_j) y_k, e^((e^(-ϵ β_i) - 1) η_j y_k)]_{$k}]
Define[
 R_{i,j} = 𝔼[ħ a_j b_i, ħ x_j y_i, e^(Sum_{k=2}^{$k+1} ((1 - e^(γ ϵ ħ))^k (ħ y_i x_j)^k)/(k (1 - e^(k γ ϵ ħ))))]_{$k},
 P_{i,j} = 𝔼[β_i α_j / ħ, η_i ξ_j / ħ,
   1 + If[$k == 0, 0, Normal@P_{{i,j},$k-1}[[3]] -
     (R_{1,2} ~B_{1,2}~ ((P_{{1,j},0})_{$k} (P_{{i,2},$k-1})_{$k}))[[3]]]]]
```

```
Define[aS_i = 𝔼[-α_i a_j, -ξ_i x_i,
      Sum[Expand[(e^(ξ_i x_i) (-ħ γ ϵ)^k)/(2^k k!) Nest[Expand[x_i^2 ∂_{x_i,2} #] &,
          e^(-ξ_i e^(ħ ϵ a_i) x_i), k]], {k, 0, $k}]]_$k ~B_{i,j}~ am_{i,j→i},
  aS̄_i = 𝔼[-a_i α_i, -x_i 𝒜_i ξ_i,
    1 + If[$k == 0, 0, Normal@aS̄_{{i},$k-1}[[3]] -
        ((aS̄_{{i},0})_$k ~B_i~ aS_i ~B_i~ (aS̄_{{i},$k-1})_$k)[[3]]]]]
Define[bS_i = R_{i,1} ~B_1~ aS_1 ~B_1~ P_{i,1},
  bS̄_i = R_{i,1} ~B_1~ aS̄_1 ~B_1~ P_{i,1},
  aΔ_{i→j,k} = (R_{1,j} R_{2,k}) ~B_{1,2}~ bm_{1,2→3} ~B_3~ P_{3,i},
  bΔ_{i→j,k} = (R_{j,1} R_{k,2}) ~B_{1,2}~ am_{1,2→3} ~B_3~ P_{i,3}]
Define[
  dm_{i,j→k} =
   (𝔼[β_i b_i + α_j a_j, η_i y_i + ξ_j x_j, 1] (aΔ_{i→1,2} ~B_2~ aΔ_{2→2,3} ~B_3~ aS̄_3)
       (bΔ_{j→-1,-2} ~B_{-2}~ bΔ_{-2→-2,-3})) ~B_{-3,-2,-1,1,2,3,i,j}~
    (P_{-1,3} P_{-3,1} am_{2,j→k} bm_{i,-2→k}),
  dS_i = 𝔼[β_i b_1 + α_i a_2, η_i y_1 + ξ_i x_2, 1] ~B_{1,2}~ (bS̄_1 aS_2) ~
    B_{1,2}~ dm_{2,1→i},
  dΔ_{i→j,k} = (bΔ_{i→3,1} aΔ_{i→2,4}) ~B_{1,2,3,4}~ (dm_{3,4→k} dm_{1,2→j})]
Define[R̄_{i,j} = Expand /@ R_{i,j} ~B_j~ dS_j,
  C_i = 𝔼[0, 0, B_i^(1/2) e^(-ħ ϵ a_i/2)]_$k,
  C̄_i = 𝔼[0, 0, B_i^(-1/2) e^(ħ ϵ a_i/2)]_$k,
  Kink_i  =  (R_{1,3} C̄_2) ~B_{1,2}~ dm_{1,2→1} ~B_{1,3}~ dm_{1,3→i},
  Kink̄_i  =  (R̄_{1,3} C_2) ~B_{1,2}~ dm_{1,2→1} ~B_{1,3}~ dm_{1,3→i}]
```

Note. $t == \epsilon \mathrm{a} - \gamma \mathrm{b}$ and $b == -t/\gamma + \epsilon \mathrm{a}/\gamma$.

```
Define[b2t_i = 𝔼[α_i a_i - β_i t_i / γ, ξ_i x_i + η_i y_i, e^(ϵ β_i a_i/γ)]_$k,
  t2b_i = 𝔼[α_i a_i - τ_i γ b_i, ξ_i x_i + η_i y_i, e^(ϵ τ_i a_i)]_$k]
Define[kR_{i,j} = R_{i,j} ~B_{i,j}~ (b2t_i b2t_j)  /. t_{i|j} → t,
  kR̄_{i,j} = R̄_{i,j} ~B_{i,j}~ (b2t_i b2t_j)  /. t_{i|j} → t,
  km_{i,j→k} = (t2b_i t2b_j) ~B_{i,j}~ dm_{i,j→k} ~B_k~ b2t_k  /.
    {t_k → t, T_k → T, τ_{i|j} → 0},
  kC_i = C_i ~B_i~ b2t_i  /. T_i → T,
  kC̄_i = C̄_i ~B_i~ b2t_i  /. T_i → T,
  kKink_i = Kink_i ~B_i~ b2t_i  /. {t_i → t, T_i → T},
  kKink̄_i = Kink̄_i ~B_i~ b2t_i  /. {t_i → t, T_i → T}]
RVK::usage =
  "RVK[xs, rots] represents a Rotational Virtual
    Knot with a list of n Xp/Xm crossings xs and
    a length 2n list of rotation numbers rots.
    Crossing sites are indexed 1 through 2n, and
    rots[[k]] is the rotation between site k-1 and
    site k. RVK is also a casting operator
    converting to the RVK presentation from other
    knot presentations.";
RVK[pd_PD] := Module[{n, xs, x, rots, front, k},
   n = Length[pd];
lb {  xs = List @@ pd /.
      x_X :→ If[PositiveQ[x], Xp[x[[4]], x[[1]]],
        Xm[x[[2]], x[[1]]]];
   rots = Table[0, {2 n}]; }
   front = {0};            } merge
   For[k = 0, k < 2 n, ++k,
    If[k == 0 ∨ FreeQ[front, -k],
     front = Flatten[front /. k → Catch[xs /. {
            Xp[k + 1, l_] | Xm[l_, k + 1] :→ Throw[{l, k + 1, 1 - l}],
            Xp[l_, k + 1] | Xm[k + 1, l_] :→
             (++rots[[l]]; Throw[{1 - l, k + 1, l}])
           }]],
     If[MatchQ[front, {___, k, ___, -k, ___}],
      --rots[[k + 1]]]
    ]
   ];
   RVK[xs, rots]
  ];
RVK[K_] := RVK[PD[K]];
rot[_, 0] = 𝔼[0, 0, 1];
rot[i_, n_Integer] /; n > 0 :=
  rot[i, n] = Module[{j}, (rot[i, n - 1] kC_j) ~B_{i,j}~ km_{i,j→i}];
rot[i_, n_Integer] /; n < 0 :=
  rot[i, n] = Module[{j}, (rot[i, n + 1] kC̄_j) ~B_{i,j}~ km_{i,j→i}];
```

C_i is ur_i ; $\bar{C}_i$ is nr_i ; $ur, nr = 1$.

```
Z[K_] := Z[RVK@K];
Z[rvk_RVK] :=
 Z[rvk] =
  Module[{todo, n, rots, ζ, done, st, x, ζ1, i, j, k,
    k1, k2, k3},
   {todo, rots} = List @@ rvk;
   AppendTo[rots, 0];
   n = Length[todo];
   ζ = 𝔼[0, 0, 1];
   done = {0};
   st = Range[0, 2 n + 1];
   While[todo =!= {},
    {x} = MaximalBy[todo,
      Length[done ⋂ {#[[1]], #[[2]], #[[1]] - 1, #[[2]] - 1}] &,
      1];
    Z$todo = todo; Z$x = x;
    {i, j} = List @@ x;
    ζ1 = Switch[Head[x],
      Xp,
      m_{j,k→j}[
       R⁺_{i,j} (R⁻_{k3,k} nr_{k1} ul_{k2} // m_{k,k1→k} // m_{k,k2→k} // m_{k,k3→k})],
      Xm,
      m_{j,k→j}[R⁻_{i,j} (R⁺_{k,k3} nr_{k1} ul_{k2} // m_{k,k1→k} // m_{k,k2→k} // m_{k,k3→k})]
     ];
    ζ1 = rot[k, rots[[i]]] ζ1 // m_{k,i→i}; rots[[i]] = 0;
    ζ1 = ζ1 rot[k, rots[[i + 1]]] // m_{i,k→i}; rots[[i + 1]] = 0;
    ζ1 = rot[k, rots[[j]]] ζ1 // m_{k,j→j}; rots[[j]] = 0;
    ζ1 = ζ1 rot[k, rots[[j + 1]]] // m_{j,k→j}; rots[[j + 1]] = 0;
    ζ *= ζ1;
    If[MemberQ[done, i], ζ = ζ // m_{i,i+1→i};
     st = st /. st[[i + 2]] → st[[i + 1]]];
    If[MemberQ[done, i - 1], ζ = ζ // m_{st[[i]],i→st[[i]]};
     st = st /. st[[i + 1]] → st[[i]]];
    If[MemberQ[done, j], ζ = ζ // m_{j,j+1→j};
     st = st /. st[[j + 2]] → st[[j + 1]]];
    If[MemberQ[done, j - 1], ζ = ζ // m_{st[[j]],j→st[[j]]};
     st = st /. st[[j + 1]] → st[[j]]];
    done = done ⋃ {i - 1, i, j - 1, j};
    todo = DeleteCases[todo, x]
   ];
   ζ /. {u_0 → u, c_0 → c, w_0 → w}
  ]
Timing@Block[{$k = 1},
  Z = kR_{1,5} kR_{6,2} kR_{3,7} kC̄_4 kKink̄_8 kKink̄_9 kKink̄_10;
  Do[Z = Z~B_{1,r}~km_{1,r→1}, {r, 2, 10}];
  Simplify /@ Z]
```

{4.10938,

$$\mathbb{E}\left[0, 0, \frac{T}{1 - T + T^2} + \frac{1}{(1 - T + T^2)^3} T \left(T\left(-1 + 2T - 3T^2 + 2T^3\right) + 2\left(-1 + T - T^3 + T^4\right) a_1 - 2\left(1 + T^3\right) x_1 y_1\right) \epsilon + O[\epsilon]^2\right]\}$$

```
PrintProfile[]
ProfileRoot is root. Profiled time: 97.468
  (   149)   0.658/ 76.046 above Bind
  (   126)   0.016/  0.016 above CF
  (    16)   0.016/  2.344 above Boot[1]
  (    18)   0.124/  6.123 above Boot[2]
  (     5)   0.063/ 12.939 above Boot[3]
CF: called 34229 times, time in 45.991/51.657
  ( 32777)   5.666/  5.666 under CF
  (   663)  31.682/ 37.348 under LZip
  (   126)   0.016/  0.016 under ProfileRoot
  (   663)   8.627/  8.627 under QZip
  ( 32777)   5.666/  5.666 above CF
Zip: called 1885 times, time in 25.667/112.361
  (   221)   4.333/ 20.942 under LZip
  (   221)   1.954/  9.685 under QZip
  (  1443)  19.380/ 81.734 under Zip
  (  1885)   4.960/  4.960 above Collect
  (  1443)  19.380/ 81.734 above Zip
LZip: called 221 times, time in 16.095/74.385
  (   221)  16.095/ 74.385 under Bind
  (   663)  31.682/ 37.348 above CF
  (   221)   4.333/ 20.942 above Zip
Collect: called 1885 times, time in 4.96/4.96
  (  1885)   4.960/  4.960 under Zip
QZip: called 221 times, time in 3.598/21.91
  (   221)   3.598/ 21.910 under Bind
  (   663)   8.627/  8.627 above CF
  (   221)   1.954/  9.685 above Zip
Bind: called 221 times, time in 0.859/97.154
  (   149)   0.658/ 76.046 under ProfileRoot
  (    29)   0.015/  2.312 under Boot[1]
  (    27)   0.092/  5.999 under Boot[2]
  (    16)   0.094/ 12.797 under Boot[3]
  (   221)  16.095/ 74.385 above LZip
  (   221)   3.598/ 21.910 above QZip
Boot[3]: called 11 times, time in 0.142/21.8
  (     5)   0.063/ 12.939 under ProfileRoot
  (     6)   0.079/  8.861 under Boot[3]
  (    16)   0.094/ 12.797 above Bind
  (     6)   0.079/  8.861 above Boot[3]
Boot[2]: called 22 times, time in 0.124/6.154
  (    18)   0.124/  6.123 under ProfileRoot
  (     4)       0/  0.031 under Boot[2]
  (    27)   0.092/  5.999 above Bind
  (     4)       0/  0.031 above Boot[2]
Boot[1]: called 24 times, time in 0.032/3.095
  (    16)   0.016/  2.344 under ProfileRoot
  (     8)   0.016/  0.751 under Boot[1]
  (    29)   0.015/  2.312 above Bind
  (     2)       0/      0 above Boot[0]
  (     8)   0.016/  0.751 above Boot[1]
Boot[0]: called 2 times, time in 0./0.
  (     2)       0/      0 under Boot[1]
```