```
(* Notation of Cup products
   A^{(p)} ⌣_i B^{(q)} ⌣_j C^{(r)}
 = Cup[A[p], B[q], C[r]][{i, j}] *)

Cup[x_, y__][{i__}] := Message[Cup::highercup] /; Length[{x, y}] - 1 ≠ Length[{i}]
Cup::highercup = "Mismatched numbers of cup products";
Del[] = 0; Del[0] = 0;
Cup[x___, 0, y___][{i__}] := 0
Cup[x_, y__][{i___, j_, k___}] := 0 /; j < 0
(* Nilpotent *)
Del[Del[x_[n_]]] := 0
(* Homomorphism *)
Del[x_ + y_] := Del[x] + Del[y]
Del[x_ + y_ + z__] := Del[x] + Del[y] + Del[z]
Cup[x___, y_ + z_, w___][{i__}] := Cup[x, y, w][{i}] + Cup[x, z, w][{i}]

(* Overall factor *)
Del[a_ * x_[n_]] := a * Del[x[n]]
Cup[x___, a_ * y_[n_], z___][{i__}] := a * Cup[x, y[n], z][{i}]

(* Associativity *)
Cup[x__, Cup[y__][{j__}], z___][{i__, k___}] :=
 Cup[x, y, z][{i, j, k}] /; {i, j, k} == Table[0, Length[{i, j, k}]]
Cup[x___, Cup[y__][{j__}], z__][{i___, k__}] :=
 Cup[x, y, z][{i, j, k}] /; {i, j, k} == Table[0, Length[{i, j, k}]]

(* Leibniz rule *)
nform[] = 0; nform[0] = 0;
nform[x_[n_]] := If[x[[0]] =!= Cup, n, Plus @@ (nform /@ x) - Total[n]]
nform[Del[x_]] := nform[x] + 1

Del[Cup[x_, y_][{i_}]] :=
 Cup[Del[x], y][{i}] + (-1)^nform[x] Cup[x, Del[y]][{i}] +
  (-1)^(nform[x] + nform[y] - i) Cup[x, y][{i - 1}] +
  (-1)^(nform[x] × nform[y] + nform[x] + nform[y]) Cup[y, x][{i - 1}]

Del[x_[n_]] := Sum[(-1)^(Plus @@ (nform /@ x[[1 ;; i]])) 
    Flatten[x[[1 ;; i]] ⌣ Del[x[[i + 1]]] ⌣ x[[i + 2 ;; Length[n] + 1]]][n],
   {i, 0, Length[n]}] /; x[[0]] === Cup && n == Table[0, Length[n]]

(* Coefficients *)
Del[x_y_] := 0
```

```
(* Setup of, e.g., Lattice Axion QED *)
dim = 4;
(* x cup-0 y cup-0 z OR {(x cup-i y) cup-j z, x cup-i (y cup-j) z} *)
creatTwoCups[x_, dim_] :=
 If[Length[x] == 3, Module[{formnum, termnum}, termnum = Length[x];
   formnum = Sum[nform[x[[i]]], {i, termnum}];
   If[formnum == dim, {(Cup @@ x)[Table[0, termnum - 1]]},
    Flatten[Table[{Cup[Cup[x[[1]], x[[2]]][{k}], x[[3]]][{formnum - dim - k}], Cup[x[[1]],
          Cup[x[[2]], x[[3]]][{k}]][{formnum - dim - k}]}, {k, 0, formnum - dim}], 1]]]]
(*
E.g., Field components for axion QED
 Axion: ϕ[0], δϕ[1]=∇*ϕ[0]+l[1], Photon: a[1], f[2]=δa[2]=∇*a[1]+z[2]
*)
axions = {ϕ[0], l[1], Del[ϕ[0]], Del[l[1]]};
photons = {a[1], z[2], Del[a[1]], Del[z[2]]};

(* Axion coupling combinations based on the form ϕFF *)
Flatten[Permutations /@ Tuples[{axions, photons, photons}], 1];
DeleteCases[%, {x_, y_, z_} /; (nform[x] + nform[y] + nform[z]) < dim];
DeleteCases[%, {x_, y_, z_} /; x[[0]] === Del && y[[0]] === Del && z[[0]] === Del];
Flatten[creatTwoCups[#, dim] & /@ %, 1];
axionCouplingList =
  DeleteDuplicates[Prepend[%, Cup[Del[ϕ[0]], Cup[∇a[1], ∇a[1]][{1}]][{0}]]];
action = Sum[cᵢ %[[i]], {i, 1, Length[%]}];
Length[axionCouplingList]
```

Out[•]= 562

```
(* Implementation of gauge invariance *)
coeffEquality[s_, gt_, cup_, numcoeff_, dynm_ : 0, printS_ : 0] :=
 Module[{transfS0, transfS, clist},
  transfS0 = Collect[(*Del/@*)Expand[(s /. gt) - s], cup];
  If[dynm == 1, transfS = transfS0 - (transfS0 /. {ϕ[0] → 0, a[1] → 0}),
   transfS = transfS0];
  If[printS == 1, Print[transfS]];
  clist = (List @@ transfS) /. Cup[x__][i__] :> 1;
  Reduce[And @@ Table[clist[[i]] == 0, {i, Length[clist]}],
   Table[cᵢ, {i, numcoeff}]]]
coeffCobdryEquality[s_, gt_, cup_, numcoeff_, dynm_ : 0, printS_ : 0] := Module[
  {transfS0, transfS, clist}, transfS0 = Collect[Del /@ Expand[(s /. gt) - s], cup];
  If[dynm == 1, transfS = transfS0 - (transfS0 /. {ϕ[0] → 0, a[1] → 0}),
   transfS = transfS0];
  If[printS == 1, Print[transfS]];
  clist = (List @@ transfS) /. Cup[x__][i__] :> 1;
  Reduce[And @@ Table[clist[[i]] == 0, {i, Length[clist]}], Table[cᵢ, {i, numcoeff}]]]
```

```
In[ ]:= coeffGenuineEquality[s_, gt_, cup_, wouldbe0_, numcoeff_, dynm_ : 0, printS_ : 0] :=
 Module[{transfS0, transfS, clist},
  transfS0 = Collect[(Expand[(s /. gt) - s] /. wouldbe0 → 0), cup];
  If[dynm == 1, transfS = transfS0 - (transfS0 /. {ϕ[0] → 0, a[1] → 0}),
   transfS = transfS0];
  If[printS == 1, Print[transfS]];
  clist = (List @@ transfS) /. Cup[x__][i__] :> 1;
  Reduce[And @@ Table[clist[[i]] == 0, {i, Length[clist]}], Table[c_i, {i, numcoeff}]]]

(* R 0-form gauge transformation *)
gaugeTransfR0 = {a[1] → a[1] + Del[λ[0]]};
Flatten[axionCouplingList /. gaugeTransfR0 /. Plus → List, 1];
λCouplingList = Complement[DeleteDuplicates[%], axionCouplingList];
(* Z 1-form gauge transformation *)
gaugeTransfZ1 = {a[1] → a[1] + m[1], z[2] → z[2] - Del[m[1]]};
Flatten[axionCouplingList /.
    (gaugeTransfZ1 /. Del[m[1]] → -Del[m[1]]) /. Plus → List, 1];
mCouplingList = Complement[DeleteDuplicates[%], axionCouplingList];
mCobdryCouplingList =
  DeleteDuplicates[(Del /@ mCouplingList) /. Plus → List /. -1 → 1];
(* Z 0-form gauge transformation *)
gaugeTransfZ0 = {ϕ[0] → ϕ[0] + κ[0], l[1] → l[1] - Del[κ[0]]};
Flatten[axionCouplingList /.
    (gaugeTransfZ0 /. Del[κ[0]] → -Del[κ[0]]) /. Plus → List, 1];
κCouplingList = Complement[DeleteDuplicates[%], axionCouplingList];
κCobdryCouplingList =
  DeleteDuplicates[(Del /@ κCouplingList) /. Plus → List /. -1 → 1];
Length /@ {λCouplingList, mCouplingList, κCouplingList}


coeffEqR0List = coeffEquality[action,
   gaugeTransfR0, λCouplingList, Length[axionCouplingList]];

(*coeffEqZ1List=coeffEquality[action,
   gaugeTransfZ1,mCouplingList,Length[axionCouplingList],1,1];*)
coeffEqZ1List = coeffGenuineEquality[action, gaugeTransfZ1,
   mCouplingList, Del[z[2]], Length[axionCouplingList], 1];
(*coeffEqZ1List=coeffCobdryEquality[action,
   gaugeTransfZ1,mCobdryCouplingList,Length[axionCouplingList],1];*)

(*coeffEqZ0List=coeffEquality[action,
   gaugeTransfZ0,κCouplingList,Length[axionCouplingList],1];*)
coeffEqZ0List = coeffCobdryEquality[action,
   gaugeTransfZ0, κCobdryCouplingList, Length[axionCouplingList], 1];
```