```
DeclareAlgebra[U_Symbol, opts__Rule] :=
 Module[{gp, sr, cp, CE, pow,
   gs = Generators /. {opts}, cs = Centrals /. {opts} },
  (#U = U@#) & /@ gs;
  gp = Alternatives @@ gs; gp = gp | gp_;
  (* gen's pattern *)
  sr = Thread[gs → Range@Length@gs]; (* sorting rule *)
  cp = Alternatives @@ cs; (* cent's pattern *)
  CE[ℰ_] := Collect[ℰ, _U,
    (Expand[#] /. ħ^d_ /; d > $TħD :↦ 0) &];
  U_i_[ℰ_] :=
   ℰ /. {t : cp :↦ t_i, u_U :↦ Replace[u, x_ :↦ x_i, 1]};
  U_i_[NCM[]] := U[];
  B[U@(x_)_i_, U@(y_)_i_] :=
   B[U@x_i, U@y_i] = U_i@B[U@x, U@y];
  B[U@(x_)_i_, U@(y_)_j_] /; i =!= j := 0;
  B[U@y_, U@x_] := CE[-B[U@x, U@y]];
  x_ ** U[] := x; U[] ** x_ := x;
  (a_. * x_U) ** (b_. * y_U) :=
   If[a b === 0, 0, CE[a b (x ** y)]];
  U[xx___, x_] ** U[y_, yy___] :=
   If[OrderedQ[{x, y} /. sr], U[xx, x, y, yy],
    U@xx ** (U@y ** U@x + B[U@x, U@y]) ** U@yy];
  U@{c_. * (l : gp)^n_, r___} /; FreeQ[c, gp] :=
   CE[c U@Table[l, {n}] ** U@{r}];
  U@{c_. * l : gp, r___} := CE[c U[l] ** U@{r}];
  U@{c_, r___} /; FreeQ[c, gp] := CE[c U@{r}];
  U@{} = U[];
  U@{l_Plus, r___} := CE[U@{#, r} & /@ l];
  U@{l_, r___} := U@{Expand[l], r};
  U[ℰ_NonCommutativeMultiply] := U /@ ℰ;
  𝔒U[poly_, specs___] := Module[{sp, null, vs, us},
    sp = Replace[{specs}, l_List :↦ l_null, {1}];
    vs = Join @@ (First /@ sp);
    us = Join @@ (sp /. l_s_ :↦ (l /. x_i_ :↦ x_s));
    CE[Total[
       CoefficientRules[poly, vs] /. (p_ → c_) :↦ c U@(us^p)
      ]] /. x_null :↦ x
    ];
  pow[ℰ_, 0] = U[]; pow[ℰ_, n_] := pow[ℰ, n - 1] ** ℰ;
  𝔖U[ℰ_, ss___Rule] := CE@Total[
     CoefficientRules[ℰ, First /@ {ss}] /.
      (p_ → c_) :↦
       c NCM @@ MapThread[pow, {Last /@ {ss}, p}]];
  S_i_[c_. * u_U] :=
   CE[(c /. S_i[U, Centrals])
     DeleteCases[u, _i] **
      U_i[NCM @@ Reverse@Cases[u, x_i :↦ S@U@x]]];
 ]
```