Pensieve header: Zero-co implementation, as a baby 1-co.

```
Print["In \"T before H\" conventions. Internal use symbols: ", {rr, pp}]
```

```
In "T before H" conventions. Internal use symbols: {rr, pp}
```

```
SetDirectory["C:\\drorbn\\AcademicPensieve\\Projects\\OneCo-1604\\Zero"]
```

```
C:\drorbn\AcademicPensieve\Projects\OneCo-1604\Zero
```

Generalities

Generalities

```
Simp[γ_] := Expand[γ];
CF[γ_] := γ /. (λ_β | λ_a) :> MapAt[Simp, λ, 1];
AutoCollecting[λ_] := (λ /: λ[0, ___] = 0;
   λ /: λ[f_, r___] + λ[g_, r___] := λ[Simp[f + g], r];
   λ /: g_ * λ[f_, r___] := λ[Simp[g f], r]);
AutoCollecting /@ {β, a};
UU /: UU[x_] + UU[y_] := UU[x + y];
UU /: a_ * UU[x_] := UU[Expand[a x]];
UU /: D[u_UU, vs__] := CF[u /. (λ_β | λ_a) :> MapAt[D[#, vs] &, λ, 1]];
UU /: Coefficient[u_UU, λ_[js___]] := Total[Cases[u, λ[f_, js] :> f, ∞]];
Kδ /: Kδ_is__ := KroneckerDelta[1, Length[Union[{is}]]];
```

Bases

```
UUBasis[T_List, H_List, f_] := Module[
   {ff, n = 0, h, t},
   ff := f_++n @@ Table[b_t, {t, T}];
   CF /@ UU /@ Flatten@{
        β[ff],
        Table[a[ff, t, h], {t, T}, {h, H}]
       } /. 1_[___] → 1
  ];
UUBasis[S_List, f_] := UUBasis[S, S, f];
UUBasis[n_Integer, m_Integer, f_] := UUBasis[Range@n, Range@m, f];
UUBasis[n_Integer, f_] := UUBasis[Range@n, f];
```

tm, hm, hts, dm

tm-def

```
UU[γ_] // tm[x_, y_, z_] := CF[UU[
    Expand[γ /. a[f_, x | y, j_] :> a[f, z, j] /. b_x|y → b_z]]];
```

hm-def

```
UU[γ_] // hm[x_, y_, z_] := CF[UU[Expand[γ /. a[f_, i_, x | y] :> a[f, i, z]]]];
```

hts-def

```
UU[γ_] // hts[y_, x_] := CF[UU[Expand[γ /.
      a[f_, i_, j_] :> a[f, i, j] - Kδ[ix] Kδ[jy] β[f b[x]]]]];
```

dm-def

```
dm[x_, y_, z_][γ_] := γ // hts[x, y] // tm[x, y, z] // hm[x, y, z]
```

tσ, hσ, dσ on {β, a}

sigma-def

```
tσ[x_List, y_List][γ_] := (rr = Replace[Thread[x → y]];
   CF[γ /. b[i_] :> b[rr@i] /. a[f_, i_, j_] :> a[f, rr@i, j]]);
tσ[x_, y_][γ_] := tσ[{x}, {y}][γ];
hσ[x_List, y_List][γ_] :=
  CF[γ /. a[f_, i_, j_] :> a[f, i, Replace[Thread[x → y]]@j]];
hσ[x_, y_][γ_] := hσ[{x}, {y}][γ];
dσ[x_, y_][γ_] := γ // tσ[x, y] // hσ[x, y];
```

tb, hb, thb, htb, db, bb on {β, a}

tb-def

```
tb[x_][UU[L_], UU[R_]] := UU[0];
```

hb-def

```
hb[y_][UU[L_], UU[R_]] := CF[UU[Expand[Distribute[pp[L, R]] /. {
         pp[0, _] → 0, pp[_, 0] → 0,
         pp[_β, _] → 0, pp[_, _β] → 0
        } /. {
        pp[a[f_, i_, y], u_] :>
         (u /. a[g_, j_, k_] :> Kδ[yk] (a[b[j] f g, i, y] - a[b[i] f g, j, k])),
        _pp → 0
       }]]];
```

thb-def

```
thb[x_, y_][UU[L_], UU[R_]] := CF[UU[Expand[Distribute[pp[L, R]] /. {
        pp[0, _] → 0, pp[_, 0] → 0, pp[_β, _] → 0, pp[_, _β] → 0,
        pp[a[f_, i_, j_], a[g_, k_, l_]] :> Kδ[yl] Kδ[xi] (-a[b[k] f g, i, j] + a[b[i] f g, k, j])
       }]]];
htb[x_, y_][L_UU, R_UU] := -thb[y, x][R, L];
```

$t_1\, h_1\, t_2\, h_2 \to t_1\, t_2\, h_1\, h_2 \to t_2\, t_1\, h_1\, h_2 \to t_2\, t_1\, h_2\, h_1 \to t_2\, h_2\, t_1\, h_1$:

db-def

```
db[x_][u_UU, v_UU] := Module[{t, h}, Plus[
    htb[x, x][u // tσ[x, t], v // hσ[x, h]] // tm[t, x, x] // hm[x, h, x],
    tb[x][u, v // hσ[x, h]] // hm[x, h, x],
    hb[x][u, v // tσ[x, t]] // tm[t, x, x],
    thb[x, x][u // hσ[x, h], v // tσ[x, t]] // tm[t, x, x] // hm[x, h, x] ]];
```

bb-def

```
bb[S_List] := Module[{w, bar, t, n = 0, i, k},
   w = #2 // dσ[S, bar /@ S];
   Sum[t = db[S[[k]]][#1, w // dσ[bar[S[[k]]], S[[k]]]];
    Do[t = t // dm[bar[S[[i]]], S[[i]], S[[i]]], {i, 1, k - 1}];
    Do[t = t // dm[S[[i]], bar[S[[i]]], S[[i]]], {i, k + 1, Length@S}];
    t, {k, Length@S}] ] &
bb[S___] := bb[{S}]
```

ct (contract)

```
ct::usage =
  "ct[h,t][L,R] contracts the head h in L with the tail t in R. ct[s][L,R]
    takes h=t=s, and ct[][L,R] takes s=0. When ambiguous, L is
    placeed below R. It is assumed that L has one h and R has one t.";
```

ct-def

```
ct[s_] := ct[s, s]; ct[] = ct[0, 0];
ct[h_, t_][UU[L_], UU[R_]] := UU[Distribute[pp[L, R]] /. {
      pp[_β, _] → 0,
      pp[a[f_, i_, h], β[g_]] :→ β[f bᵢ g / bₜ],
      pp[a[f_, i_, h], a[g_, t, j_]] :→ a[f g, i, j],
      pp[a[f_, i_, h], a[g_, j_, k_]] :→ a[f bᵢ g / bₜ, j, k],
      pp[a[__], _] → 0}] // CF;
```

dect (de-contract)

```
dect::usage =
  "dect[h,t][uu] returns a pair {L,R} such that ct[h,t][L,R]=uu. Similarly
    for dect[s] and dect[]. uu is assumed to be atomic.";
dect[s_] := dect[s, s];
dect[] = dect[0, 0];
dect[h_, t_][β[f_]] := {};
dect[h_, t_][δβ[f_]] := TBD;
```

Exporting the above as PDF files

The below is adapted from pensieve://2016-04/GaussGassner/GaussGassnerDemo.nb.

```
SetDirectory["C:\\drorbn\\AcademicPensieve\\Projects\\OneCo-1604\\Zero"];

ConditionalExport[fname_String, rest___] := Module[{temp, exists},
  temp = "ConditionalExportTemporary" <> "." <> FileExtension[fname];
  exists = FileExistsQ[fname];
```

```
  Export[temp, rest];
  If[exists && FileByteCount[fname] === FileByteCount[temp],
   DeleteFile[temp],
   (* else *) Print["Exporting " <> fname <> "..."];
   If[exists, DeleteFile[fname]];
   RenameFile[temp, fname]
  ];
  fname
 ]

Button["Export",
 SetOptions[$FrontEndSession, PrintingStyleEnvironment → "Working"];
 TagProperties[_] := {};
 (*TagProperties["ct-def"] = {PageWidth →6/0.65};*)
 Options[CellExport] = {
   PageWidth → 4 / 0.65, CellFilter → Identity,
   ExportDirectory → "Snips", ExportBaseFilename → Automatic,
   ExportFormat → ".pdf", ExportOptions → {}, Split → False
  };
 CellExport[tag_String, opts___Rule] := CellExport[
   NotebookGet[EvaluationNotebook[]],
   tag, opts
  ];
 CellExport[nb_Notebook, tag_String] := CellExport[nb, tag, TagProperties[tag]];
 CellExport[nb_Notebook, tag_String, OptionsPattern[]] := Module[
   {cells, cell, filename, format},
   filename = FileNameJoin[{
      OptionValue[ExportDirectory] /. Automatic → Directory[],
      OptionValue[ExportBaseFilename] /. Automatic → tag
     }];
   format = OptionValue[ExportFormat];
   cells = OptionValue[CellFilter][Cases[
      nb, c_Cell /; FreeQ[List @@ c, Cell] && ! FreeQ[c, CellTags → tag],
      Infinity
     ]];
   If[! OptionValue[Split],
    If[Length[cells] ≥ 1,
     If[Length[cells] == 1,
      cells = Append[First[cells], PageWidth → 1.2 × 72 OptionValue[PageWidth]],
      cells = Cell[CellGroup[cells], PageWidth → 72 OptionValue[PageWidth]]
     ];
     ConditionalExport[
      filename <> format, cells,
```

```
      ImageResolution → 300,
      OptionValue[ExportOptions]
     ]
    ],
    k = 0;
    Table[
     ++k;
     ConditionalExport[
      filename <> "-" <> ToString[k] <> format,
      Append[cell, PageWidth → 72 OptionValue[PageWidth]],
      ImageResolution → 300,
      OptionValue[ExportOptions]
     ],
     {cell, cells}
    ]
   ]
  ];
 nb = NotebookGet[EvaluationNotebook[]];
 tags = Cases[nb, (CellTags → tag_) :> tag, Infinity] // Union;
 CellExport /@ tags;
 Print["Done."]
]
```

Export

```
Exporting Snips\ct-def.pdf...
Done.
Exporting Snips\ct-def.pdf...
Done.
Exporting Snips\ct-def.pdf...
Done.
Exporting Snips\ct-def.pdf...
Done.
```