Initialization

```
Needs["Theorema`"]
```

```
SetGlobals[TraceLevel → 0];
```

```
SetGlobals[JuxtapositionForTimes → True];
```

```
Use[⟨Built-in["Numbers"], Built-in["Operators"], Built-in["Complex Numbers"],
  Built-in["Quantifiers"], Built-in["Tuples"], Built-in["Connectives"], Built-in["Sets"]⟩]
```

```
SetOptions[FlattenKB, ExpandFunctors → "replace"];
```

```
Off[General::"spell1"];
```

```
Unprotect[Or]; ClearAll[Or]; Protect[Or];
```

```
Off[Unset::"write"]
```

```
DeclareAsIdentifiers[□̃]
```

Number Domains

Natural Numbers

```
Definition["ℕ", ℕ = NaturalNumbers[]]
```

```
Definition["Natural Numbers", any[],
 NaturalNumbers[] = Functor[N, any[x, y],
    𝕤 = ⟨⟩
    ─────────────
    ∈_N[x] ⟺ is-natural[x]
    x >_N y ⇔ x > y
    □_N = 0
    x *_N y = x + y
    lcm_N[x, y] = { x ⇐ x > y
                  { y ⇐ True
    ]]
```

Mma Auxiliary Functions

```
Clear[mymult];
```

```
Begin["System`"];
```

```
mymult[x_, y_] := x * y;
```

```
End[];
```

```
Built-in["Mult",
 ™MyMult → mymult]
```

```
Clear[myminus];
```

```
Begin["System`"];
```

```
myminus[x_, y_] := x - y;
```

```
End[];
```

```
Built-in["Minus",
 ™MyMinus → myminus]
```

```
Clear[isnumber];
```

```
Begin["System`"];
```

```
isnumber[™D | L | R | A | B | ⌈x__⌉ | x | x^m_] := False;
```

```
isnumber[™Times[™D, m__] | ™Times[L, m__] | ™Times[R, m__] | ™Times[A, m__] |
    ™Times[B, m__] | ™Times[™Ceiling[m__], y___] | ™Times[x, y___]] := False;
```

```
isnumber[x_] :=
  (x /. {™I → ⅈ, ™E → ⅇ, ™Times → Times, ™Plus → Plus, ™Minus → Minus, ™Divide → Divide, ™Power → Power}) ∈
   Complexes;
```

```
End[];
```

```
Built-in["IsNumber",
 ™IsNumber → isnumber]
```

```
Definition["Mma Auxiliary Functions", any[],
 MmaAuxF[] = Functor[N, any[x, y],
    𝕤 = 〈〉
    ∈_N[x] ⟺ ™IsNumber[x]
    x ·_N y = ™MyMult[x, y]
    x -_N y = ™MyMinus[x, y]
    ]]
```

Reduction Field

```
Definition["Reduction Field", any[D],
 ReductionField[D] = Functor[〈N, extends[D]〉, any[x, y],
    𝕤 = 〈〉
    ∈_N[x] ⇔ ∈_D[x]
    0_N = 0_D
    1_N = 1_D
    x +_N y = x +_D y
    -_N x = -_D x
    x -_N y = x -_D y
    x *_N y = x *_D y
    x /_N y = x /_D y
    x >_N y ⇔ x >_D y
    x >_N y ⇔ ((x ≠ 0_D) ⋀ (y = 0_D))
    rdm_N[x, y] (* the reduction multiplier of x modulo y  *) = { x /_D y   ⇐ x ≠ 0_D ⋀ y ≠ 0_D
                                                              { 0_D       ⇐ otherwise
    lcrd_N[x, y] (* the least common reducible of x and y  *) = { 1_D  ⇐ x ≠ 0_D ⋀ y ≠ 0_D
                                                               { 0_D  ⇐ otherwise
    ]]
```

Rational Number Field

```
Definition["ℚ", ℚ = RationalNumberField[]]
```

```
Definition["Rational Number Field", any[],
 RationalNumberField[] = Functor[N, any[x, y],
    s = ⟨⟩
    ________________________________
    ∈_N[x] ⟺ is-rational[x]
    x >_N y ⇔ x > y
    0_N = 0
    1_N = 1
    x +_N y = x + y
    -_N x = -x
    x -_N y = x - y
    x *_N y = x * y
    ∇_N x = 1 / x
    x /_N y = x / y
    power_N[x, 0] = 1_N
    power_N[x, y] = x *_N power_N[x, y - 1]
    ]]
```

Complex Numbers

IsComplex

```
Clear[IsComplex];
```

```
Begin["System`"];
```

```
IsComplex[x_] := (x /. {™I → ⅈ, ™E → ⅇ}) ∈ Complexes;
```

```
End[];
```

```
Built-in["IsComplex",
 ™IsComplex → IsComplex]
```

```
$TmaOperatorTranslations = DeleteCases[$TmaOperatorTranslations, tr_ /; MemberQ[tr, ™E | ™I, ∞]];
```

ComplexLt

```
Clear[ComplexLt];
```

```
Begin["System`"];
```

```
ComplexLt[x_, x_] := False;
```

```
ComplexLt[x_, y_] := OrderedQ[{x, y} /. {™I → ⅈ, ™E → ⅇ}] ;
```

```
End[];
```

```
Built-in["ComplexLt",
 ™ComplexLt → ComplexLt]
```

ComplexNumberField

```
Definition["ℂ", ℂ = ComplexNumberField[]]
```

```
Definition["Complex Number Field", any[],
 ComplexNumberField[] = Functor[N, any[x, y, z],
    𝕤 = 〈〉
    ─────────────────────
    ∈_N[x] ⟺ ™IsComplex[x]
    x >_N y ⇔ ™ComplexLt[y, x]
    0_N = 0
    1_N = 1
    x +_N y = x + y
    x *_N y = simplify_N[x * y]
    x *_N y *_N z = (x *_N y) *_N z
    -_N x = -x
    x -_N y = x - y
    ∇_N x = 1 / x
    x /_N y = x / y
    lcm_N[x, y] = { x  ⇐  x > y
                  { y  ⇐  True
    power_N[x, y] = Power[x, y]
    exp_N[x] = Exp[x]
    simplify_N[x] = mma-simplify[x]
    PrettyPrint_N[x] = x
    ]]
```

ComplexNumbersMonoid

```
Definition["ℂM", ℂM = ComplexNumbers[]]
```

```
Definition["Complex Numbers", any[],
 ComplexNumbers[] = Functor[N, any[x, y],
    s = ⟨⟩
    ─────────────────────
    ∈[x] ⟺ ™IsComplex[x]
    N
    x > y ⇔ ™ComplexLt[y, x]
      N
    ⬚ = 0
    N
    x * y = x + y
      N
    lcm[x, y] = { x ⇐ x > y
    N           { y ⇐ True
    ]]
```

Word Monoids

```
Definition["Finite Chain", any[L],
 FinChain[L] = Functor[C, any[x, y],
    s = ⟨⟩
    ───────────────────────────────────────────
    ∈[x] ⟺ (x ∈ L)
    C                                                        ]]
    x > y ⇔      ∃             ∃        ((x = L_i) ⋀ (y = L_j))
      C   i=1,…,|L| j=i+1,…,|L|
```

```
Definition["Cartesian Product", any[D, D̄, A, B],
 CartesianProduct[D, D̄] = Functor[N, any[X, Y, X̄, Ȳ],
    s = ⟨⟩
    ────────────────────────────────────────────────────────────────────
    ∈[X] ⟺ (is-tuple[X] ⋀ (|⟨D, D̄⟩| = |X|) ⋀         ∀          (   ∈     [X_j]))
    N                                        j=1,…,|⟨D,D̄⟩|   ⟨D,D̄⟩_j
    ⬚ = ⟨    ⬚        |           ⟩
    N    ⟨D,D̄⟩_j j=1,…,|⟨D,D̄⟩|
                                 ⎧ X_k    >    Y_k
    X > Y ⇔ (       ∃        (⋀ ⎨     ⟨D,D̄⟩_k                       ))      ]
      N     k=1,…,|⟨D,D̄⟩|       ⎩      ∀      (X_l = Y_l)
                                   l=1,…,k-1
    X * Y = ⟨X_m    *    Y_m          |          ⟩
      N        ⟨D,D̄⟩_m       m=1,…,|⟨D,D̄⟩|
    lcm[⟨X, X̄⟩, ⟨Y, Ȳ⟩] = ⟨  lcm  [⟨X, X̄⟩_n, ⟨Y, Ȳ⟩_n]         |         ⟩
    N                      ⟨D,D̄⟩_n                   n=1,…,|⟨X,X̄⟩|
    ]
                  A × B = CartesianProduct[A, B]
```

Definition["Tuples Monoid", any[A],

TuplesMonoid[A] = Functor[N, any[x, y, z, $\bar{x}$, $\bar{y}$, X, Y],

$$\frac{\mathbb{s} = \langle\rangle}{\underset{N}{\in}[X] \Longleftrightarrow \left(\left(\text{is-tuple}[X] \bigwedge \left(\underset{i=1,\ldots,|X|}{\forall} \underset{A}{\in}[X_i]\right) \bigwedge (X_{|X|} \neq 0)\right) \bigvee (X = \langle\rangle)\right)}$$

$\underset{N}{\square} = \langle\rangle$

$\left(\langle x, \bar{x}\rangle \underset{N}{>} \langle\rangle\right) \Leftrightarrow \text{True}$

$\left(\langle\rangle \underset{N}{>} \langle\bar{y}\rangle\right) \Leftrightarrow \text{False}$

$$\left(\langle x, \bar{x}\rangle \underset{N}{>} \langle y, \bar{y}\rangle\right) \Leftrightarrow \left(\bigvee \begin{cases} (x = y) \bigwedge \langle\bar{x}\rangle \underset{N}{>} \langle\bar{y}\rangle \\ x \underset{A}{>} y \end{cases}\right)$$

$x \underset{N}{*} \langle\rangle = x$

$\langle\rangle \underset{N}{*} y = y$

$\langle\bar{x}, x\rangle \underset{N}{*} \langle\bar{y}, y\rangle =$

$$\begin{cases} \left\langle \langle\bar{x}, x\rangle_k \underset{A}{*} \langle\bar{y}, y\rangle_k \Big|_{k=1,\ldots,|\langle\bar{x},x\rangle|} \right\rangle & \Leftarrow \quad (|\langle\bar{y}, y\rangle| = |\langle\bar{x}, x\rangle|) \\ \left\langle \langle\bar{x}, x\rangle_k \underset{A}{*} \langle\bar{y}, y\rangle_k \Big|_{k=1,\ldots,|\langle\bar{x},x\rangle|} \right\rangle \frown \left\langle \langle\bar{y}, y\rangle_k \Big|_{k=|\langle\bar{x},x\rangle|+1,\ldots,|\langle\bar{y},y\rangle|} \right\rangle & \Leftarrow \quad |\langle\bar{y}, y\rangle| > |\langle\bar{x}, x\rangle| \\ \left\langle \langle\bar{x}, x\rangle_k \underset{A}{*} \langle\bar{y}, y\rangle_k \Big|_{k=1,\ldots,|\langle\bar{y},y\rangle|} \right\rangle \frown \left\langle \langle\bar{x}, x\rangle_k \Big|_{k=|\langle\bar{y},y\rangle|+1,\ldots,|\langle\bar{x},x\rangle|} \right\rangle & \Leftarrow \quad \text{otherwise} \end{cases}$$

$\underset{N}{\text{lcm}}[x, \langle\rangle] = x$

$\underset{N}{\text{lcm}}[\langle\rangle, y] = y$

$$\underset{N}{\text{lcm}}[\langle x, \bar{x}\rangle, \langle y, \bar{y}\rangle] = \begin{cases} x \smile \underset{N}{\text{lcm}}[\langle\bar{x}\rangle, \langle\bar{y}\rangle] & \Leftarrow \quad x \underset{A}{>} y \\ y \smile \underset{N}{\text{lcm}}[\langle\bar{x}\rangle, \langle\bar{y}\rangle] & \Leftarrow \quad \text{otherwise} \end{cases}$$

]]

Cartesian Product

Definition["Cartesian Power", any[D, n],

CartesianPower[D, n] = CartesianProduct[$D \big|_{i=1,\ldots,n}$]]

```
Definition["Word Monoid", any[L],
 LexWords[L] = Functor[W, any[u, v, w, ξ, η, ξ̄, η̄, ζ̄],
   s = ⟨⟩
   ─────────────────────────────────────
   ∈_W[w] ⟺ (⋀{ is-tuple[w]
               ∀_{i=1,…,|w|} ∈_L[w_i] })
   ⬚_W (* empty word *) = ⟨⟩
   v *_W w (* multiplication of two words *) = v ⌢ w
   u *_W v *_W w (* multiplication of three words *) = (u *_W v) *_W w
   (⟨η̄⟩ |_W ⟨ξ̄, η̄, ζ̄⟩) (* middle divisibility *) ⇔ True
   (⟨η̄⟩ |_W ⟨ξ̄⟩) ⇔ False
   rquot_W[⟨⟩, v] (* right quotient *) = ⬚_W
   rquot_W[⟨ξ̄, η̄⟩, ⟨η̄⟩] = ⬚_W                                        ]]
   rquot_W[w, v] = rquot_W[⟨w_i |_{i=1,…,|w|-1}⟩, v] ⌢ w_|w|
   lquot_W[w, v] (* left quotient *) = ⟨w_i |_{i=1,…,|w|-|v|-|rquot_W[w,v]|}⟩
   lcm_W[⟨η̄, ξ̄⟩, ⟨ξ̄, ζ̄⟩] (* the least common reducible *) = ⟨η̄, ξ̄, ζ̄⟩
   (⟨η, η̄⟩ >_W ⟨⟩) (* lexicographic ordering *) ⇔ True
   (⟨⟩ >_W ⟨η̄⟩) ⇔ False
   (⟨η, η̄⟩ >_W ⟨ξ, ξ̄⟩) ⇔ (⋁{ (η = ξ) ⋀ ⟨η̄⟩ >_W ⟨ξ̄⟩
                              η >_L ξ })
```

```
Definition["Word Monoid with Degree Ordering", any[L],
 DegWords[L] = where[B = LexWords[L],
   Functor[W, any[u, v, w, η, η̄],
   s = ⟨⟩
   ─────────────────────────────
   ∈_W[w] ⟺ ∈_B[w]
   ⬚_W = ⬚_B
   v *_W w = v *_B w
   u *_W v *_W w = u *_B v *_B w
   (v |_W w) ⇔ (v |_B w)
   rquot_W[w, v] = rquot_B[w, v]
   lquot_W[w, v] = lquot_B[w, v]                  ]]]
   lcm_W[w, v] = lcm_B[w, v]
   (⟨η, η̄⟩ >_W ⟨⟩) (* degree ordering *) ⇔ True
   (⟨⟩ >_W ⟨η̄⟩) ⇔ False
   (v >_W w) ⇔ (⋁{ |v| > |w|
                   ⋀{ |v| = |w|
                      v >_B w } })
```

```
Free Module
```

Definition["Free Module", any[K, B],

FreeModule[K, B] = Functor[V, any[c, d, x, y, ξ, η, A, $\bar{x}$, $\bar{y}$],

$$\mathbb{s} = \langle\rangle$$

$$\underset{V}{\in}[x] \Longleftrightarrow \left(\text{where}\left[z = |x|,\ \bigwedge\begin{cases}\text{is-tuple}[x] \\ \underset{i=1,\ldots,z}{\forall}\left(\bigwedge\begin{cases}\text{is-tuple}[x_i] \\ |x_i| = 2 \\ \underset{V}{\text{is-coeff}}[(x_i)_1] \\ \underset{V}{\text{is-bvec}}[(x_i)_2]\end{cases}\right) \\ \underset{i=1,\ldots,z-1}{\forall}\ (x_i)_2 \underset{B}{>} (x_{i+1})_2\end{cases}\right]\right)$$

$$\underset{V}{\text{is-bvec}}[\xi] \Leftrightarrow \underset{B}{\in}[\xi]$$

$$\underset{V}{\text{is-coeff}}[c] \Leftrightarrow \left(\underset{K}{\in}[c] \bigwedge c \neq \underset{K}{0}\right)$$

$$\underset{V}{\text{bvec}}[\xi] = \left\langle\left\langle \underset{K}{1}, \xi\right\rangle\right\rangle$$

$$\underset{V}{\text{to-bvec}}[\langle\langle c, \xi\rangle\rangle] = \begin{cases}\xi & \Leftarrow\ c = \underset{K}{1} \\ \underset{B}{\square} & \Leftarrow\ \text{otherwise}\end{cases}$$

$$\underset{V}{\text{to-bvec}}[A] = \underset{B}{\square}$$

$$\underset{V}{0} = \langle\rangle$$

$$\langle\rangle \underset{V}{+} y = y$$

$$x \underset{V}{+} \langle\rangle = x$$

$$\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle \underset{V}{+} \left\langle\langle d, \eta\rangle, \bar{y}\right\rangle = \begin{cases}\langle c, \xi\rangle \smile \left(\langle\bar{x}\rangle \underset{V}{+} \left\langle\langle d, \eta\rangle, \bar{y}\right\rangle\right) & \Leftarrow\ \xi \underset{B}{>} \eta \\ \langle d, \eta\rangle \smile \left(\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle \underset{V}{+} \langle\bar{y}\rangle\right) & \Leftarrow\ \eta \underset{B}{>} \xi \\ \left\langle c \underset{K}{+} d, \xi\right\rangle \smile \left(\langle\bar{x}\rangle \underset{V}{+} \langle\bar{y}\rangle\right) & \Leftarrow\ (\xi = \eta) \bigwedge c \underset{K}{+} d \neq \underset{K}{0} \\ \langle\bar{x}\rangle \underset{V}{+} \langle\bar{y}\rangle & \Leftarrow\ \text{otherwise}\end{cases}$$

$$V[+][x, \bar{x}] = x \underset{V}{+} V[+][\bar{x}]$$

$$\underset{V}{-}\langle\rangle = \langle\rangle$$

$$\underset{V}{-}\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle = \left\langle \underset{K}{-}c, \xi\right\rangle \smile \left(\underset{V}{-}\langle\bar{x}\rangle\right)$$

$$x \underset{V}{-} y = x \underset{V}{+} \left(\underset{V}{-}y\right)$$

$$0 \underset{V}{\cdot} y \text{ (* scalar multiplication *) } = \langle\rangle$$

$$c \underset{V}{\cdot} \langle\rangle = \langle\rangle$$

$$c \underset{V}{\cdot} \left\langle\langle d, \eta\rangle, \bar{y}\right\rangle = \left\langle c \underset{K}{*} d, \eta\right\rangle \smile c \underset{V}{\cdot} \langle\bar{y}\rangle$$

$$y \underset{V}{\cdot} 0 = \langle\rangle$$

$$\langle\rangle \underset{V}{\cdot} c = \langle\rangle$$

$$\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle \underset{V}{\cdot} d = \left\langle c \underset{K}{*} d, \xi\right\rangle \smile \langle\bar{x}\rangle \underset{V}{\cdot} d$$

$$\underset{V}{\text{coord}}[\langle\rangle, x] \text{ (* coordinate wrt a basis *) } = \langle\rangle$$

$$\underset{V}{\text{coord}}\left[\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle, \xi\right] = c$$

$$\underset{V}{\text{coord}}\left[\left\langle\langle c, \xi\rangle, x, \bar{x}\right\rangle, \eta\right] = \underset{V}{\text{coord}}\left[\langle x, \bar{x}\rangle, \eta\right]$$

$$\underset{V}{\text{coord}}[\langle\langle c, \xi\rangle\rangle, \eta] = \underset{K}{0}$$

$$\left(\langle\rangle \underset{V}{>} y\right) \text{ (* ordering *) } \Leftrightarrow \text{False}$$

$$\left(\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle \underset{V}{>} \langle\rangle\right) \Leftrightarrow \text{True}$$

$$\left(\left\langle\langle c, \xi\rangle, \bar{x}\right\rangle \underset{V}{>} \left\langle\langle d, \eta\rangle, \bar{y}\right\rangle\right) \Leftrightarrow \left(\bigvee\begin{cases}\xi \underset{B}{>} \eta \\ \bigwedge\begin{cases}\xi = \eta \\ c \underset{K}{>} d\end{cases} \\ \bigwedge\begin{cases}\xi = \eta \\ c = d \\ \langle\bar{x}\rangle \underset{V}{>} \langle\bar{y}\rangle\end{cases}\end{cases}\right)$$

]]

Monoid Algebra

```
Definition["Monoid Algebra", any[K, W],
 MonoidAlgebra[K, W] = where[V = FreeModule[K, W],
   Functor[P, any[c, d, f, g, ξ, η, f̄, m̄, n̄],
     𝔰 = ⟨⟩
     ─────────────────────────────
     ∈_P[f] ⟺ ∈_V[f]
     is-bvec_P[ξ] ⇔ is-bvec_V[ξ]
     is-coeff_P[c] ⇔ is-coeff_V[c]
     0_P = 0_V
     P[+][f̄] = V[+][f̄]
     -_P f = -_V f
     f -_P g = f -_V g
     ew_P[ξ] = bvec_V[ξ]
     c ·_P f = c ·_V f
     coord_P[f, ξ] = coord_V[f, ξ]
     ─────────────────────────────
     1_P = ⟨⟨1_K, ⬚_W⟩⟩
     ⟨⟩ *_P g (* multiplication of two polynomials *) = ⟨⟩
     f *_P ⟨⟩ = ⟨⟩
     ⟨⟨c, ξ⟩, m̄⟩ *_P ⟨⟨d, η⟩, n̄⟩ = (⟨⟨c *_K d, ξ *_W η⟩⟩ +_P ⟨⟨c, ξ⟩⟩ *_P ⟨n̄⟩) +_P ⟨m̄⟩ *_P ⟨⟨d, η⟩, n̄⟩
     P[*][f, f̄] = f *_P P[*][f̄]
     (⟨⟩ >_P g) (* ordering *) ⇔ False
     (⟨⟨c, ξ⟩, m̄⟩ >_P ⟨⟩) ⇔ True
     (⟨⟨c, ξ⟩, m̄⟩ >_P ⟨⟨d, η⟩, n̄⟩) ⇔ ( ⋁{ ξ >_W η
                                           ⋀{ ξ = η
                                              c >_K d
                                           ⋀{ ξ = η
                                              c = d
                                              ⟨m̄⟩ >_P ⟨n̄⟩ )
     rrdm_P[⟨⟩, g] (* right reduction multiplier *) = 0_P
     rrdm_P[⟨⟨c, ξ⟩, m̄⟩, ⟨⟩] = 0_P
     rrdm_P[⟨⟨c, ξ⟩, m̄⟩, ⟨⟨d, η⟩, n̄⟩] = { ⟨⟨rdm_K[c, d], rquot_W[ξ, η]⟩⟩  ⇐ rdm_K[c, d] ≠ 0_K ⋀ η |_W ξ
                                          0_P                              ⇐ otherwise
     lrdm_P[⟨⟩, f] (* left reduction multiplier *) = 0_P
     lrdm_P[⟨⟨c, ξ⟩, m̄⟩, ⟨⟩] = 0_P
     lrdm_P[⟨⟨c, ξ⟩, m̄⟩, ⟨⟨d, η⟩, n̄⟩] = { ⟨⟨1_K, lquot_W[ξ, η]⟩⟩  ⇐ rdm_K[c, d] ≠ 0_K ⋀ η |_W ξ
                                          0_P                     ⇐ otherwise
     lcrd_P[⟨⟨c, ξ⟩, m̄⟩, ⟨⟨d, η⟩, n̄⟩] (* least common reducible *) = ⟨⟨lcrd_K[c, d], lcm_W[ξ, η]⟩⟩
   ]]]
```

```
Definition["Groebner Extension", any[P],
 GB[P] = Functor[⟨G, extends[P]⟩, any[c, f, g, k, l, r, t, ξ, lhs1, lhs2, rhs1, rhs2, f̄, m̄, R, S],
    s = ⟨⟩
    ─────────────────────────────
    f >_G g ⇔ f >_P g
    0_G = 0_P
    1_G = 1_P
    f +_G g ⇔ f +_P g
    G[*][f, f̄] = P[*][f, f̄]
    hredp_G[f, g] (* head reduction modulo polynomial *) = f -_P lrdm_P[f, g] *_P g *_P rrdm_P[f, g]
    hredp_G[f, g, l, r] = f -_P l *_P g *_P r
    hred_G[f, S] (* head reduction modulo system *) = where[q = S[f],
      hredp_G[f, q2, q1, q3]]
    sred_G[⟨⟩, S, t] (* step reduction modulo system at given position *) = 0_P
    sred_G[⟨⟨c, ξ⟩, m̄⟩, S, 1] = hred_G[⟨⟨c, ξ⟩, m̄⟩, S]
    sred_G[⟨⟨c, ξ⟩, m̄⟩, S, t] = ⟨⟨c, ξ⟩⟩ +_G sred_G[⟨m̄⟩, S, t - 1]
    sred_G[f, S] (* step reduction modulo system *) = cred_G[f, S, 1]
    cred_G[f, S, k] (* complete reduction modulo system at given position *) =
     where[f1 = sred_G[f, S, k],
      { cred_G[f, S, k + 1]   ⇐  (f1 = f) ⋀ (k < |f|)
      { f1                    ⇐  otherwise                    ]]]
     tred_G[f, S] (* total reduction modulo system *) = where[q = sred_G[f, S],
      { tred_G[q, S]   ⇐  q ≠ f
      { f              ⇐  otherwise      ]
    spol_G[f, g] (* the S-polynomial of f and g *) = where[L = lcrd_P[f, g],
      hredp_G[L, f] -_G hredp_G[L, g]]
    spolred_G[⟨f, g⟩, S] = tred_G[spol_G[f, g], S]
    Gb_G[R, S] = where[pairs = ⟨⟨R_i, R_j⟩ | _{i=1,..,|R|; j=1,..,|R|}  R_i ≠ R_j⟩,
      Gb_G[R, pairs, S]]
    Gb_G[R, ⟨⟩, S] = R
    Gb_G[R, ⟨⟨f, g⟩, m̄⟩, S] = where[h = spolred_G[⟨f, g⟩, S],
      {              Gb_G[R, ⟨m̄⟩, S]                         ⇐    h = 0_P
      { Gb_G[R ⌣ h, (⟨h ⇸ R⟩ ⌢ ⟨m̄⟩) ⌢ ⟨R ⇷ h⟩, S]           ⇐  otherwise
     ]
```

```
Definition["Theorema general functions", any[F, l, r],
 l ⇸ F = ⟨⟨l, F_i⟩ | _{i=1,..,|F|}⟩
 F ⇷ r = ⟨⟨F_i, r⟩ | _{i=1,..,|F|}⟩ ]
```

Exponential Polynomials

MakeBoxes&MakeExpressions (Formatting&Parsing)

```
MakeExpression[RowBox[{SubscriptBox["∂", Q_], p_}], StandardForm] :=
  MakeExpression[RowBox[{RowBox[{Q, "[", "™Der", "]"}], "[", p, "]"}], StandardForm];
```

```
MakeExpression[RowBox[{SubscriptBox["∫", Q_], p_}], StandardForm] :=
  MakeExpression[RowBox[{RowBox[{Q, "[", "™Int", "]"}], "[", p, "]"}], StandardForm];
```

```
MakeBoxes[Q_[™Der][p_], StandardForm] :=
  RowBox[{SubscriptBox["∂", MakeBoxes[Q, StandardForm]], MakeBoxes[p, StandardForm]}];
```

```
MakeBoxes[Q_[™Int][p_], StandardForm] :=
  RowBox[{SubscriptBox["∫", MakeBoxes[Q, StandardForm]], MakeBoxes[p, StandardForm]}];
```

```
MakeExpression[RowBox[{"⌈", x_, "⌉"}], StandardForm] :=
  MakeExpression[RowBox[{"™Ceiling", "[", x, "]"}], StandardForm];
```

```
MakeBoxes[™Ceiling[x_], StandardForm] := RowBox[{"⌈", MakeBoxes[x, StandardForm], "⌉"}];
```

```
MakeExpression[RowBox[{"⌊", x_, "⌋"}], StandardForm] :=
  MakeExpression[RowBox[{"™Floor", "[", x, "]"}], StandardForm];
```

```
MakeBoxes[™Floor[x_], StandardForm] := RowBox[{"⌊", MakeBoxes[x, StandardForm], "⌋"}];
```

Built-ins

```
Begin["System`"];
```

```
MmaSimplify[p_] := FullSimplify[p];
```

```
End[];
```

```
Built-in["MmaSimplify",
 mma-simplify → MmaSimplify]
```

Exponential Polynomials

```
Definition["Exp Polys", any[K],
 ExpPolys[K] = where[P = MonoidAlgebra[K, ℕ×ℂM],
   Functor[E, any[c, d, m, p, q, λ, ξ, m̄, n̄],
     𝕤 = ⟨⟩
     ─────────────────────────────────────────
     ∈_E[p] ⟺ ∈_P[p]
     0_E = 0_P
     1_E = 1_P
     p >_E q ⇔ p >_P q
     p +_E q = p +_P q
     -_E p = -_P p
     p -_E q = p -_P q
     c ·_E p = c ·_P p
     p *_E q = p *_P q
     ew_E[p] = ew_P[p]
     coord_E[p, ξ] = coord_P[p, ξ]
     is-bvec_E[ξ] ⇔ is-bvec_P[ξ]
     is-coeff_E[c] ⇔ is-coeff_P[c]
     ─────────────────────────────────────────
     lcrd_E[⟨⟨c, λ⟩⟩, ⟨⟨d, ξ⟩⟩] = ⟨⟨lcrd_K[c, d], lcm_{ℕ×ℂM}[λ, ξ]⟩⟩
     is-scal_E[⟨⟨c, ⬚_{ℕ×ℂM}⟩⟩] ⇔ is-coeff_P[c]
     is-scal_E[p] ⇔ False
     to-scal_𝓕[⟨⟩] = 0_K
     to-scal_E[⟨⟨c, ⬚_{ℕ×ℂM}⟩⟩] = c
     eval_E[⟨⟩, d] = 0_K
     eval_E[⟨⟨c, ⟨⟩⟩⟩, d] = c
     eval_E[⟨⟨c, ⟨0, λ⟩⟩⟩, d] = c *_K exp_K[λ *_K d]
     eval_E[⟨⟨c, ⟨m, λ⟩⟩, m̄⟩, d] = c *_K power_K[d, m] *_K exp_K[λ *_K d] +_K eval_E[⟨m̄⟩, d]
     ∂_E⟨⟩ = 0_P
     ∂_E⟨⟨c, ⟨⟩⟩⟩ = 0_P
     ∂_E⟨⟨c, ⟨0, λ⟩⟩, n̄⟩ = λ ·_P ⟨⟨c, ⟨0, λ⟩⟩⟩ +_P ∂_E⟨n̄⟩
     ∂_E⟨⟨c, ⟨m, λ⟩⟩, n̄⟩ = ⟨⟨c *_K m, ⟨m - 1, λ⟩⟩⟩ +_P λ ·_P ⟨⟨c, ⟨m, λ⟩⟩⟩ +_P ∂_E⟨n̄⟩
     ∫_E⟨⟩ = 0_P
     ∫_E⟨⟨c, ⟨⟩⟩⟩ = ⟨⟨c, ⟨1, 0⟩⟩⟩
     ∫_E⟨⟨c, ⟨m, 0⟩⟩, n̄⟩ = ⟨⟨c /_K (m + 1), ⟨m + 1, 0⟩⟩⟩ +_P ∫_E⟨n̄⟩
     ∫_E⟨⟨c, ⟨0, λ⟩⟩, n̄⟩ = (⟨⟨c /_K λ, ⟨0, λ⟩⟩⟩ -_P ⟨⟨c /_K λ, ⟨0, 0⟩⟩⟩) +_P ∫_E⟨n̄⟩
     ∫_E⟨⟨c, ⟨m, λ⟩⟩, n̄⟩ = (⟨⟨c /_K λ, ⟨m, λ⟩⟩⟩ -_P ∫_E m ·_P ⟨⟨c /_K λ, ⟨m - 1, λ⟩⟩⟩) +_P ∫_E⟨n̄⟩
     PrettyPrint_E[p] = ™Plus[PrettyPrintM_E[p_i] | _{i=1,…,|p|}]
     PrettyPrint_E[⟨m⟩] = PrettyPrintM_E[m]
     PrettyPrint_E[⟨⟩] = 0
     PrettyPrintM_E[⟨c, ⟨m, λ⟩⟩] = ™Times[c, ™Times[x^m, ™E^{λ x}]]
 ]]]
```

Built-ins

```
Begin["System`"];

Clear[CoeffQ, Mon, Coeff, SplitPoly, FuncOrder];

CoeffQ[expr_] := FreeQ[expr, x];

Mon[t1_, t2___] := Mon[t1 * t2];

Mon[Times[c_?CoeffQ, t__]] := Mon[t];

Mon[Times[c_, t__]] := Mon[c] * Mon[t];

Mon[Power[ⅇ, Plus[λ_?NumericQ, μ_]]] := Power[ⅇ, μ];

Mon[c_?CoeffQ] := 1;

Mon[s_] := s;

Coeff[t1_, t2___] := Coeff[t1 * t2];

Coeff[Times[c_, t__]] := Coeff[c] * Coeff[t];

Coeff[ⅇ^Plus[λ_?NumericQ,μ_]] := ⅇ^λ;

Coeff[s_?CoeffQ] := s ;

Coeff[s_] := 1 ;

SplitPoly[c_?CoeffQ] := {{c, mma[1]}};

SplitPoly[Plus[t1_, t2___]] := Join[SplitPoly[t1], Flatten[SplitPoly /@ {t2}, 1]];

SplitPoly[0] := ™Tuple[];

SplitPoly[s_] := {{FullSimplify[Coeff[s]], mma[Mon[s]]}};
```

```
SplitPoly[t1_, t2___] := SplitPoly /@ {t1, t2};
```

```
MySplitPoly[A_] := SplitPoly[Expand[A]] /. List → ™Tuple;
```

```
FuncOrder[p_, p_] := False;
```

```
FuncOrder[p_, q_] := OrderedQ[{p, q}];
```

```
FuncQ[mma[t___]] := True;
```

```
FuncQ[s_] := False;
```

```
Clear[ToMma];
```

```
ToMma[〈m_, r___〉] := Plus[ToMma[m], ToMma[r]];
```

```
ToMma[〈c_, mma[m_]〉] := Times[c, m];
```

```
ToMma[t1_, t2__] := Plus[ToMma[t1], ToMma[t2]];
```

```
ToMma[〈〉] := 0;
```

```
ToMma[] := 0;
```

```
ToMma[c_] := c;
```

```
Clear[MmaPlus, MmaMinus, MmaTimes, MmaDif, MmaInt];
```

```
MmaPlus[p_, q_] := MySplitPoly[ToMma[p] + ToMma[q]];
```

```
MmaMinus[p_] := MySplitPoly[-ToMma[p]];
```

```
MmaMinus[p_, q_] := MySplitPoly[ToMma[p] - ToMma[q]];
```

```
MmaTimes[p_, q_] := MySplitPoly[ToMma[p] * ToMma[q]];
```

```
MmaDiff[p_] := MySplitPoly[Expand[D[ToMma[p], x]]];
```

```
MmaInt[p_] := MySplitPoly[Expand[TrigToExp[Integrate[ToMma[p], {x, 0, x}]]]];
```

```
MmaSubs[p_, c_] := Subs[ToMma[p], c];
```

```
Subs[p_, c_] := p /. x → c;
```

```
MmaSimplify[p_] := FullSimplify[p];
```

```
SetOptions[TransformResultFromMma, PlusNegativeAsMinus → True,
  TimesNegPowerAsFraction → False, PowerRationalAsRoot → False, NegPowerAsReciprocal → False];
```

```
TmaTRFM[p_] := TransformResultFromMma[p /. {™Times → Times, ™Plus → Plus, ™Minus → Minus, ™Power → Power}];
```

```
End[];
```

```
Built-in["FuncOrder",
 func-less → FuncOrder]
```

```
Built-in["FuncQ",
 is-func → FuncQ]
```

```
Built-in["MmaDiff",
 mma-diff → MmaDiff]
```

```
Built-in["MmaInt",
 mma-int → MmaInt]
```

```
Built-in["ToMma",
 ™toMma → ToMma]
```

```
Built-in["MmaPlus",
 mma-plus → MmaPlus]
```

```
Built-in["MmaMinus",
 mma-minus → MmaMinus]
```

```
Built-in["MmaTimes",
 mma-times → MmaTimes]
```

```
Built-in["MmaSubs",
 mma-subs → MmaSubs]
```

```
Built-in["MmaSimplify",
 mma-simplify → MmaSimplify]
```

```
Built-in["TmaTRFM",
 tma-trfm → TmaTRFM]
```

```
Built-in["TmaSplit",
 tma-split → MySplitPoly]
```

mytimes

```
Clear[mytimes];
```

```
mytimes /: MakeBoxes[mytimes[x_], StandardForm] := MakeBoxes[x, StandardForm];
```

```
mytimes /: MakeBoxes[mytimes[x_, y_], StandardForm] :=
  RowBox[{MakeBoxes[x, StandardForm], MakeBoxes[y, StandardForm]}];
```

```
mytimes /: MakeBoxes[mytimes[x_, y_, z__], StandardForm] := RowBox[
   {MakeBoxes[x, StandardForm], MakeBoxes[y, StandardForm], MakeBoxes[mytimes[z], StandardForm]}];
```

```
mytimes[1, x__] := x;
```

```
mytimes[-1, x__] := -x;
```

```
mytimes[x__, 1] := x;
```

```
mytimes[x__, -1] := -x;
```

Mathematica Functions

```
Definition["Mma Functions", any[K],
 MmaFct[K] = Functor[E, any[c, d, l, p, q, ξ],
    s = ⟨⟩
    ────────────────────────────────────────────
    ∈_E[p] ⟺ ⋀ { is-tuple[p]
                 ∀_{i=1,…,|p|} ( ⋀ { is-tuple[p_i]
                                      |p_i| = 2
                                      is-coeff_E[(p_i)_1]
                                      is-bvec_E[(p_i)_2] } )
                 ∀_{i=1,…,|p|-1} func-less[(p_{i+1})_2, (p_i)_2] }
    0_E = ⟨⟩
    1_E = ⟨⟨1, mma[1]⟩⟩
    p >_E q ⇔ func-less[q, p]
    p +_E q = mma-plus[p, q]
    -_E p = mma-minus[p]
    p -_E q = mma-minus[p, q]
    c ·_E q = mma-times[c, q]
    p *_E q = mma-times[p, q]
    is-bvec_E[ξ] ⇔ is-func[ξ]
    is-coeff_E[c] ⇔ (∈_K[c] ⋀ c ≠ 0_K)
    is-scal_E[⟨⟨c, mma[1]⟩⟩] ⇔ is-coeff_E[c]
    is-scal_E[p] ⇔ False
    to-scal_E[⟨⟩] = 0_K
    to-scal_E[⟨⟨c, mma[1]⟩⟩] = c
    eval_E[p, d] = mma-subs[p, d]
    ∂_E p = mma-diff[p]
    ∫_E p = mma-int[p]
    wronmin_E[l] = tma-wronmin[l]
    simplify_E[p] = mma-simplify[p]                    ]]
```

Integro-Differential Operators

```
Definition["Basis", any[𝓕],
 Basis[𝓕] = Functor[B, any[ξ, η],
    s = ⟨⟩
    ──────────────────────────────────
    ∈_B[⟨"⌈⌉", ξ⟩] ⟺ is-bvec_𝓕[ξ]
    ⟨"⌈⌉", ξ⟩ >_B ⟨"⌈⌉", η⟩ ⇔ ξ >_𝓕 η          ]]
```

```
Definition["Basis for Characters", any[K],
 CharBasis[K] = Functor[C, any[c, d],
     𝕊 = ⟨⟩
     ─────────────────────────────
     ∈_C[⟨"⌊⌋", c⟩] ⟺ (∈_K[c])
     ⟨"⌊⌋", c⟩ >_C ⟨"⌊⌋", d⟩ ⇔ c >_K d        ]]
     𝔼 = ⟨"⌊⌋", 0⟩
```

```
Definition["Double Basis", any[B, C],
 DoubleBasis[B, C] = Functor[D, any[d, e],
     𝕊 = ⟨⟩
     ─────────────────────────────────────────────
     ∈_D[d] ⟺ ((∈_B[d]) ⋁ (∈_C[d]))
     ⟨"⌈⌉", d⟩ >_D ⟨"⌈⌉", e⟩ ⇔ ⟨"⌈⌉", d⟩ >_B ⟨"⌈⌉", e⟩
     ⟨"⌊⌋", d⟩ >_D ⟨"⌊⌋", e⟩ ⇔ ⟨"⌊⌋", d⟩ >_C ⟨"⌊⌋", e⟩ ]]
     ⟨"⌈⌉", d⟩ >_D ⟨"⌊⌋", e⟩ ⇔ True
     ⟨"⌊⌋", d⟩ >_D ⟨"⌈⌉", e⟩ ⇔ False
```

```
Definition["↦", any[l, L],
 l ↦ L = Functor[E, any[x, y],
     𝕊 = ⟨⟩
     ──────────────────────────────
     ∈_E[x] ⟺ (∈_L[x] ⋁ (x = l))
     l >_E x ⇔ (l ≠ x)                 ]]
     y >_E l ⇔ False
     x >_E y ⇔ x >_L y
```

```
Definition["Green System", any[ℱ, K],
 GreenSystem[ℱ, K] = where[𝒜 = FreeIntDiffOp[ℱ, K],
   Functor[Gr, any[c, f, g, φ, ψ, ξ, q̄, ū, v̄, A],
```

```
𝔰 = ⟨⟩
────────────────────────────────────────────
rule_Gr[⟨ū, ⟨"⌈⌉", f⟩, ⟨"⌈⌉", g⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨⟨"⌈⌉", f⟩, ⟨"⌈⌉", g⟩⟩⟩⟩ -_𝒜
    fun-to-op_𝒜[ew_𝒜[f *_ℱ g]], ⟨v̄⟩⟩
rule_Gr[⟨ū, ⟨"⌊⌋", φ⟩, ⟨"⌊⌋", ψ⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩, ⟨"⌊⌋", ψ⟩⟩⟩⟩ -_𝒜
    ⟨⟨1_K, ⟨⟨"⌊⌋", ψ⟩⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, ⟨"⌊⌋", φ⟩, ⟨"⌈⌉", f⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩, ⟨"⌈⌉", f⟩⟩⟩⟩ -_𝒜
    (⟨⟨"⌊⌋", φ⟩⟩ ⋄_𝒜 ew_𝒜[f]) *_𝒜 ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∂", ⟨"⌈⌉", f⟩, v̄⟩] = ⟨⟨ū⟩, (⟨⟨1_K, ⟨"∂", ⟨"⌈⌉", f⟩⟩⟩⟩ -_𝒜
      ⟨"∂"⟩ ⋄_𝒜 ew_𝒜[f]) -_𝒜 ⟨⟨1_K, ⟨⟨"⌈⌉", f⟩, "∂"⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∂", ⟨"⌊⌋", φ⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∂", ⟨"⌊⌋", φ⟩⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∂", "∫", v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∂", "∫"⟩⟩⟩ -_𝒜 1_𝒜, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", ⟨"⌈⌉", f⟩, "∫", v̄⟩] =
  ⟨⟨ū⟩, (⟨⟨1_K, ⟨"∫", ⟨"⌈⌉", f⟩, "∫"⟩⟩⟩ -_𝒜 ⟨"∫"⟩ ⋄_𝒜 ew_𝒜[f] *_𝒜 ⟨⟨1_K, ⟨"∫"⟩⟩⟩) +_𝒜 ⟨⟨1_K, ⟨"∫"⟩⟩⟩ *_𝒜 ⟨"∫"⟩ ⋄_𝒜 ew_𝒜[f], ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", ⟨"⌈⌉", f⟩, "∂", v̄⟩] =
  ⟨⟨ū⟩, (((⟨⟨1_K, ⟨"∫", ⟨"⌈⌉", f⟩, "∂"⟩⟩⟩ -_𝒜 ⟨⟨1_K, ⟨⟨"⌈⌉", f⟩⟩⟩⟩) +_𝒜 ⟨⟨1_K, ⟨"∫"⟩⟩⟩ *_𝒜 ⟨"∂"⟩ ⋄_𝒜 ew_𝒜[f]) +_𝒜
    (⟨𝔼⟩ ⋄_𝒜 ew_𝒜[f]) *_𝒜 ⟨⟨1_K, ⟨𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", ⟨"⌈⌉", f⟩, ⟨"⌊⌋", φ⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∫", ⟨"⌈⌉", f⟩, ⟨"⌊⌋", φ⟩⟩⟩⟩ -_𝒜
    ⟨"∫"⟩ ⋄_𝒜 ew_𝒜[f] *_𝒜 ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", "∫", v̄⟩] = ⟨⟨ū⟩, (⟨⟨1_K, ⟨"∫", "∫"⟩⟩⟩ -_𝒜 ⟨"∫"⟩ ⋄_𝒜 ew_𝒜[f] *_𝒜 ⟨⟨1_K, ⟨"∫"⟩⟩⟩) +_𝒜 ⟨⟨1_K, ⟨"∫"⟩⟩⟩ *_𝒜 ⟨"∫"⟩ ⋄_𝒜
      ew_𝒜[f], ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", "∂", v̄⟩] = ⟨⟨ū⟩, (⟨⟨1_K, ⟨"∫", "∂"⟩⟩⟩ -_𝒜 1_𝒜) +_𝒜 ⟨⟨1_K, ⟨𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", ⟨"⌊⌋", φ⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∫", ⟨"⌊⌋", φ⟩⟩⟩⟩ -_𝒜 (⟨"∫"⟩ ⋄_𝒜 ew_𝒜[f]) *_𝒜 ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, 𝔼, ⟨"⌊⌋", φ⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨𝔼, ⟨"⌊⌋", φ⟩⟩⟩⟩ -_𝒜 ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, ⟨"⌊⌋", φ⟩, 𝔼, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨⟨"⌊⌋", φ⟩, 𝔼⟩⟩⟩ -_𝒜 ⟨⟨1_K, ⟨𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, 𝔼, ⟨"⌈⌉", f⟩, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨𝔼, ⟨"⌈⌉", f⟩⟩⟩⟩ -_𝒜 (⟨𝔼⟩ ⋄_𝒜 ew_𝒜[f]) *_𝒜 ⟨⟨1_K, ⟨𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, ⟨"⌊⌋", 0⟩, "∫", v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨𝔼, "∫"⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∂", 𝔼, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∂", 𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", ⟨"⌈⌉", f⟩, 𝔼, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∫", ⟨"⌈⌉", f⟩, 𝔼⟩⟩⟩ -_𝒜 ⟨"∫"⟩ ⋄_𝒜 ew_𝒜[f] *_𝒜 ⟨⟨1_K, ⟨𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, 𝔼, 𝔼, v̄⟩] = ⟨⟨ū⟩, ⟨⟨1_K, ⟨𝔼, 𝔼⟩⟩⟩ -_𝒜 ⟨⟨1_K, ⟨𝔼⟩⟩⟩, ⟨v̄⟩⟩
rule_Gr[⟨ū, "∫", ⟨"⌈⌉", ⟨⟨c, f⟩⟩⟩, v̄⟩] =
  { ⟨⟨ū⟩, ⟨⟨1_K, ⟨"∫", ⟨"⌈⌉", ⟨⟨c, f⟩⟩⟩⟩⟩⟩ -_𝒜 ⟨⟨1_K, ⟨⟨"⌈⌉", ⟨⟨c, f⟩⟩⟩, "∫"⟩⟩⟩, ⟨v̄⟩⟩   ⇐ is-const_ℱ[c]
  { ⟨⟨⟩, ⟨⟩, ⟨⟩⟩                                                                   ⇐ otherwise
rule_Gr[A] = ⟨0_𝒜, 0_𝒜, 0_𝒜⟩
𝒢_Gr[⟨⟨c, ξ⟩, q̄⟩] = where[l = ⟨⟨1_K, (rule_Gr[ξ])_1⟩⟩, m = c ·_𝒜 (rule_Gr[ξ])_2, r = ⟨⟨1_K, (rule_Gr[ξ])_3⟩⟩,
    { ⟨l, m, r⟩     ⇐   m ≠_𝒜 0
    { ⟨l, 0_𝒜, r⟩   ⇐  otherwise ]
```

```
Definition["Groebner Basis Normal Form", any[G, S],
 GBNF[G, S] = Functor[C, any[a, b, c, ā, ℬ, F],
    𝕤 = 〈〉
    ─────────────────────────
    NF_C[a] = tred_G[a, S]
    0_C = 0_G
    1_C = 1_G
    C[+][ā] = G[+][ā]
    -_C a = -_G a
    a -_C b = a -_G b
    C[*][ā] = G[*][ā]
    c ._C a = c ._G a
    ew_C[a] = ew_G[a]
    EvalMat_C[ℬ, F] = EvalMat_G[ℬ, F]
  ]]
```

```
Definition["Quotient Algebra", any[G],
 QuotAlg[G] = Functor[Q, any[a, b, c, ā, ℬ, F],
    𝕤 = 〈〉
    ─────────────────────────
    ∈_Q[a] ⟺ (NF_G[a] = a)
    0_Q = 0_G
    1_Q = 1_G
    Q[+][ā] = NF_G[G[+][ā]]
    -_Q a = NF_G[-_G a]
    a -_Q b = NF_G[a -_G b]
    Q[*][ā] = NF_G[G[*][ā]]
    c ._Q a = NF_G[c ._G a]
    ew_Q[a] = ew_G[a]
    EvalMat_Q[ℬ, F] = EvalMat_G[ℬ, F]
  ]]
```

```
Definition["FreeIntDiffOp", any[ℱ, K],
 FreeIntDiffOp[ℱ, K] =
  where[ℳ = MonoidAlgebra[K, DegWords["∂" ↦ ("∫" ↦ (DoubleBasis[Basis[ℱ], CharBasis[K]]))]],
   Functor[𝒜, any[b, c, d, f, g, α, β, ξ, w, f̄, ḡ, m̄, w̄, ℬ, F, J],
```

```
𝔰 = 〈〉
--------------------------------------------------------------------------------
∈_𝒜[f̄] ⟺ ∈_𝓜[f̄]
0_𝒜 = 0_𝓜
𝒜[+][f̄] = 𝓜[+][f̄]
-_𝒜 f = -_𝓜 f
f -_𝒜 g = f -_𝓜 g
c ·_𝒜 f = c ·_𝓜 f
𝒜[⋆][f̄] = 𝓜[⋆][f̄]
is-bvec_𝒜[b] ⇔ is-bvec_𝓜[b]
is-scal_𝒜[〈〈c, 〈〉〉〉] ⇔ (∈_K [c])
is-scal_𝒜[b] ⇔ False
lrdm_𝒜[f, g] = lrdm_𝓜[f, g]
--------------------------------------------------------------------------------
rrdm_𝒜[f, g] = rrdm_𝓜[f, g]
f >_𝒜 g ⇔ f >_𝓜 g
lcm_𝒜[f, g] = lcm_{DegWords["∂"↦("∫"↦(DoubleBasis[Basis[ℱ],CharBasis[K]]))]}[f, g]
lcrd_𝒜[〈〈c, f〉, f̄〉, 〈〈d, g〉, ḡ〉] (* least common reducible *) = 〈〈lcrd_K[c, d], lcm_𝒜[f, g]〉〉
1_𝒜 = 〈〈1_K, 〈〉〉〉
ew_𝒜[〈〉] = 〈〉
ew_𝒜[〈〈〈〉, g〉〉] = 〈〉
ew_𝒜[〈〈〈〈〈〈c, w〉〉, 〈〈α〉, β, J〉〉, w̄〉, g〉, m̄〉] = (α ⋆_K c) ·_𝓜 ew_𝓜[〈〈〈〈〈〈1, w〉〉, 〈〈1〉, β, J〉〉〉, g〉〉] +_{MonoidAlgebra[K,ℱ]}
   ew_𝒜[〈〈〈w̄〉, g〉〉] +_{MonoidAlgebra[K,ℱ]} ew_𝒜[〈m̄〉]
ew_𝒜[〈〈〈〈〈〈c, w〉〉, ξ〉, w̄〉, g〉, m̄〉] = c ·_𝓜 ew_𝓜[〈〈〈〈〈〈1, w〉〉, ξ〉〉, g〉〉] +_{MonoidAlgebra[K,ℱ]} ew_𝒜[〈〈〈w̄〉, g〉〉] +_{MonoidAlgebra[K,
   ew_𝒜[〈m̄〉]
to-scal_𝒜[〈〉] = 0
to-scal_𝒜[〈〈c, ⬚_{DegWords["∂"↦("∫"↦(DoubleBasis[Basis[ℱ],CharBasis[K]]))]}〉〉] = c
op-to-fun_𝒜[〈〉] (* integro-diff operators to functions *) = 〈〉
op-to-fun_𝒜[〈〈c, 〈〉〉〉] = c ·_ℱ 1_ℱ
op-to-fun_𝒜[〈〈c, 〈〈"⌈⌉", ξ〉〉〉, m̄〉] = 〈c, ξ〉 ⌣ op-to-fun_𝒜[〈m̄〉]
fun-to-op_𝒜[〈〉] (* functions to integro-diff operators *) = 〈〉
fun-to-op_𝒜[〈〈c, ξ〉, m̄〉] = { 〈〈to-scal_ℱ[〈〈c, ξ〉〉], 〈〉〉〉 +_𝓜 fun-to-op_𝒜[〈m̄〉]   ⇐  is-scal_𝒜[〈〈c, ξ〉〉]
                              { 〈〈c, 〈〈"⌈⌉", ξ〉〉〉〉 +_𝓜 fun-to-op_𝒜[〈m̄〉]           ⇐  otherwise
〈〉 ⋄_𝒜 f (* the action of int-diff word on function *) = fun-to-op_𝒜[f]
〈w̄, "∂"〉 ⋄_𝒜 〈〈c, f〉〉 = c ·_𝒜 〈w̄〉 ⋄_𝒜 ew_𝒜[∂_ℱ f]
〈w̄, "∫"〉 ⋄_𝒜 〈〈c, f〉〉 = c ·_𝒜 〈w̄〉 ⋄_𝒜 ew_𝒜[∫_ℱ f]
〈w̄, 〈"⌊⌋", d〉〉 ⋄_𝒜 〈〈c, f〉〉 = c ·_𝒜 〈w̄〉 ⋄_𝒜 ew_𝒜[eval_ℱ[f, d]]
〈w̄, 〈"⌈⌉", b〉〉 ⋄_𝒜 〈〈c, f〉〉 = c ·_𝒜 〈w̄〉 ⋄_𝒜 ew_𝒜[b ⋆_ℱ f]
〈〉 ⊙_𝒜 f (* the action of int-diff op on function *) = 〈〉
〈〈c, ξ〉, m̄〉 ⊙_𝒜 f = c ·_𝒜 ξ ⋄_𝒜 f +_𝒜 〈m̄〉 ⊙_𝒜 f
EvalMat_𝒜[ℬ, F] = 〈〈to-scal_𝒜[ℬ_i ⊙_𝒜 op-to-fun_𝒜[F_j]] | j=1,..,|F|〉 | i=1,..,|ℬ|〉
]]
```

```
Definition["PrettyPrinting for IntDiffPol", any[𝓕, K],
 PrettyPrintingIDP[𝓕, K] = Functor[P, any[b, c, m, n, p, r, w, α, β, λ, γ, η, J, η̄, ξ̄, J̄],

  𝕤 = ⟨⟩
  ──────────────────────────────────────────
  PrettyPrintW_P[⟨⟩] = ""

  PrettyPrintW_P[⟨λ, η, η̄⟩] = { ™Times[λ², PrettyPrintW_P[⟨η̄⟩]]                          ⇐ λ = η
                               { mytimes[PrettyPrintL_P[λ], PrettyPrintW_P[⟨η, η̄⟩]]     ⇐ otherwise

  PrettyPrintW_P[⟨η⟩] = PrettyPrintL_P[η]

  PrettyPrintL_P[⟨"⌈⌉", ⟨⟨⟩, ⟨⟩, J⟩⟩] = PrettyPrintJ_P[J]

  PrettyPrintL_P[⟨"⌈⌉", ⟨⟨⟩, β, J⟩⟩] = mytimes[™Power[u, β], PrettyPrintJ_P[J]]

  PrettyPrintL_P[⟨"⌈⌉", ⟨α, ⟨⟩, J⟩⟩] = mytimes[™Power[u[0], α], PrettyPrintJ_P[J]]

  PrettyPrintL_P[⟨"⌈⌉", ⟨α, β, J⟩⟩] = mytimes[™Power[u[0], α], ™Power[u, β], PrettyPrintJ_P[J]]

  PrettyPrintJ_P[⟨⟩] = ""

  PrettyPrintJ_P[⟨⟨b, γ⟩, J̄⟩] = mytimes[™Times["∫", PrettyPrint_𝓕[b]], ™Power[u, γ], PrettyPrintJ_P[⟨J̄⟩]]]]

  PrettyPrintL_P["∂"] = "∂"

  PrettyPrintL_P["∫"] = "∫"

  PrettyPrintL_P[⟨⟩] = ""

  PrettyPrintL_P[η] = η

  PrettyPrintM_P[⟨c, w⟩] = ™Times[PrettyPrint_K[c], PrettyPrintW_P[w]]

  PrettyPrint_P[⟨⟨⟨c, η⟩, ξ̄⟩, η̄⟩] = ⟨PrettyPrint_P[⟨⟨⟨c, η⟩, ξ̄⟩, η̄⟩_i]  |  i=1,…,|⟨⟨⟨c,η⟩,ξ̄⟩,η̄⟩|⟩

  PrettyPrint_P[p] = ™Plus[PrettyPrintM_P[p_i]  |  i=1,…,|p|]

  PrettyPrint_P[⟨m⟩] = PrettyPrintM_P[m]

  PrettyPrint_P[⟨⟩] = 0
```

```
Definition["IntDiffOp", any[𝓕, K],
 IntDiffOp[𝓕, K] = where[𝓐 = FreeIntDiffOp[𝓕, K],
  QuotAlg[GBNF[GB[𝓐], 𝓖_GreenSystem[𝓕,K]]]]
]
```

Integro-Differential Polynomials

```
Definition["Differential Polynomials", any[𝓕, K],
 DiffPolys[𝓕, K] = where[M = MonoidAlgebra[𝓕, TuplesMonoid[ℕ]],
   Functor[V, any[c, d, f, k, m, n, w, x, y, λ, ξ, η, A, η̄, m̄, x̄, ȳ],
    𝕤 = ⟨⟩
    ─────────────────────────────
    ∈_V[x] ⟺ (∈_M[x])
    ─────────────────────────────
    is-bvec_V[ξ] ⇔ is-bvec_M[ξ]
    is-coeff_V[c] ⇔ is-coeff_M[c]
    bvec_V[ξ] = bvec_M[ξ]
    ─────────────────────────────
    to-exp_V[⟨⟩] = ⟨⟩
    to-exp_V[⟨⟨c, ξ⟩, x̄⟩] = { ⟨⟨c ·_𝓕 1_𝓕, ξ⟩⟩ +_V to-exp_V[⟨x̄⟩]              ⇐ is-coeff_V[⟨⟨c, ⟨0, 0⟩⟩⟩]
                            { ⟨⟨⟨⟨c, ⟨0, 0⟩⟩⟩ ·_𝓕 1_𝓕, ξ⟩⟩ +_V to-exp_V[⟨x̄⟩]   ⇐ otherwise
    0_V = 0_M
    1_V = ⟨⟨1_𝓕, ⟨⟩⟩⟩
    x +_V y = x +_M y
    ─────────────────────────────
    V[+][x, x̄] = x +_M M[+][x̄]
    -_V x = -_M x
    x -_V y = x -_M y
    ─────────────────────────────
    c ·_V y (* scalar multiplication *) = c ·_M y
    x *_V y = x *_M y
    D_V[⟨⟩] = ⟨⟩
    D_V[⟨m̄, 0⟩] = D_V[⟨m̄⟩]
    D_V[⟨k⟩] = ⟨⟨k, ⟨k - 1, 1⟩⟩⟩
    D_V[⟨m̄, n⟩] = D_V[⟨m̄⟩] *_MonoidAlgebra[K,TuplesMonoid[ℕ]] ⟨⟨1_K, ⟨0 | i=1,…,|⟨m̄⟩|, n⟩⟩⟩ +_MonoidAlgebra[K,TuplesMonoid[ℕ]]
      ⟨⟨n, ⟨m̄, n - 1, 1⟩⟩⟩
    ─────────────────────────────
    ∂_V⟨⟩ = ⟨⟩
    ∂_V⟨⟨c, ξ⟩, x̄⟩ = { ∂_𝓕 c ·_V ⟨⟨1_𝓕, ξ⟩⟩ +_V c ·_V to-exp_V[D_V[ξ]] +_V ∂_V⟨x̄⟩   ⇐ ∂_𝓕 c ≠ 0_𝓕
                     { c ·_V to-exp_V[D_V[ξ]] +_V ∂_V⟨x̄⟩                              ⇐ otherwise
    PrettyPrint_V[x] = ™Plus[PrettyPrintM_V[x_i] | i=1,…,|x|]
    PrettyPrint_V[⟨m⟩] = PrettyPrintM_V[m]
    PrettyPrint_V[⟨⟩] = 0
    PrettyPrintM_V[⟨c, w⟩] = tma-trfm[™Times[PrettyPrintC_V[c], PrettyPrintW_V[w]]]
    PrettyPrintC_V[c] = PrettyPrint_𝓕[c]
    PrettyPrintW_V[⟨⟩] = 1
    PrettyPrintW_V[⟨η, η̄⟩] = ™Power[u, ⟨η, η̄⟩]
   ]]]
```

```
Definition["Term Monoid for IDP", any[N, 𝓕],
 TermMonoid[𝓕, N] =
  CartesianProduct[TuplesMonoid[N], TuplesMonoid[N], TuplesMonoid[𝓕 × TuplesMonoid[N]]]]
```

```
Definition["Integro-Differential Polys", any[K, 𝓕],
```

IntDiffPolys[$\mathcal{F}$, K] = where[M = FreeModule[$\mathcal{F}$, TermMonoid[$\mathcal{F}$, $\mathbb{N}$]],

Functor[V, any[a, b, c, f, g, j, k, p, w, α, β, γ, η, $\bar{\eta}$, A, J, $\tilde{b}$, $\tilde{\alpha}$, $\tilde{\beta}$, $\tilde{\gamma}$, $\tilde{J}$, $\bar{a}$, $\bar{\alpha}$, $\bar{\gamma}$, $\bar{J}$, $\bar{\bar{J}}$],

$\mathbb{s} = \langle\rangle$

$\underset{V}{\in}[\langle\langle f, J\rangle, \bar{J}\rangle] \Leftrightarrow (\underset{M}{\in}[\langle\langle f, J\rangle, \bar{J}\rangle])$

$f \underset{V}{>} g \Leftrightarrow f \underset{M}{>} g$

$\text{is-}\underset{V}{\text{bvec}}[\langle\langle c, f\rangle\rangle] \Leftrightarrow (\underset{\mathcal{F}}{\in}[c] \bigwedge \text{is-}\underset{M}{\text{bvec}}[f])$

$\text{is-}\underset{V}{\text{coeff}}[c] \Leftrightarrow \text{is-}\underset{M}{\text{coeff}}[c]$

$\underset{V}{\text{ew}}[f] = \underset{M}{\text{bvec}}[f]$

$\underset{V}{\text{lcrd}}[\langle\langle a, J\rangle\rangle, \langle\langle b, \tilde{J}\rangle\rangle] = \langle\langle \underset{\mathcal{F}}{\text{lcrd}}[a, b], \underset{\text{TermMonoid}[\mathcal{F},\mathbb{N}]}{\text{lcm}}[J, \tilde{J}]\rangle\rangle$

$\text{is-}\underset{V}{\text{const}}[\langle\langle c, \langle\alpha, \langle\rangle, J\rangle\rangle\rangle] \Leftrightarrow (\alpha \neq \underset{\text{TuplesMonoid}[\mathbb{N}]}{\square} \bigvee J \neq \underset{\text{TuplesMonoid}[\mathcal{F}\times\text{TuplesMonoid}[\mathbb{N}]]}{\square})$

$\text{is-}\underset{V}{\text{const}}[A] \Leftrightarrow \text{False}$

$\text{is-}\underset{V}{\text{scal}}[\langle\langle c, \underset{\text{TermMonoid}[\mathcal{F},\mathbb{N}]}{\square}\rangle\rangle] \Leftrightarrow \text{is-}\underset{V}{\text{coeff}}[c]$

$\text{is-}\underset{V}{\text{scal}}[\langle\langle c, \langle\rangle\rangle\rangle] \Leftrightarrow \text{is-}\underset{V}{\text{coeff}}[c]$

$\text{is-}\underset{V}{\text{scal}}[a] \Leftrightarrow \text{False}$

$\text{to-}\underset{V}{\text{scal}}[\langle\rangle] = \underset{K}{0}$

$\text{to-}\underset{V}{\text{scal}}[\langle\langle c, \underset{\text{TermMonoid}[\mathcal{F},\mathbb{N}]}{\square}\rangle\rangle] = c$

$\text{to-}\underset{V}{\text{scal}}[\langle\langle c, \langle\rangle\rangle\rangle] = c$

$\underset{V}{0} = \langle\rangle$

$\underset{V}{1} = \langle\langle \underset{\mathcal{F}}{1}, \langle\langle\rangle, \langle\rangle, \langle\rangle\rangle\rangle\rangle$

$\underset{V}{-} f = \underset{M}{-} f$

$f \underset{V}{+} g = f \underset{M}{+} g$

$f \underset{V}{-} g = f \underset{M}{-} g$

$\langle\rangle \underset{V}{\cdot} f = \langle\rangle$

$c \underset{V}{\cdot} f = c \underset{M}{\cdot} f$

$$\underset{V}{\text{eval}}[\langle\langle f, \langle\alpha, \beta, \langle\rangle\rangle\rangle\rangle, 0] = \begin{cases} \langle\langle \underset{\mathcal{F}}{\text{eval}}[f, 0] \underset{\mathcal{F}}{\cdot} \underset{\mathcal{F}}{1}, \langle \alpha \underset{\text{TuplesMonoid}[\mathbb{N}]}{*} \beta, \langle\rangle, \langle\rangle\rangle\rangle\rangle & \Leftarrow \text{is-}\underset{\mathcal{F}}{\text{coeff}}[\underset{\mathcal{F}}{\text{eval}}[f, 0]] \\ \langle\langle \underset{\mathcal{F}}{\text{eval}}[f, 0], \langle \alpha \underset{\text{TuplesMonoid}[\mathbb{N}]}{*} \beta, \langle\rangle, \langle\rangle\rangle\rangle\rangle & \Leftarrow \underset{\mathcal{F}}{\text{eval}}[f, 0] \neq \underset{K}{0} \\ \langle\rangle & \Leftarrow \text{otherwise} \end{cases}$$

$\underset{V}{\text{eval}}[\langle\langle f, \langle\alpha, \beta, J\rangle\rangle\rangle, 0] = \langle\rangle$

$\text{to-}\underset{V}{\text{idp}}[\langle\rangle] = \langle\rangle$

$\text{to-}\underset{V}{\text{idp}}[\langle\langle c, J\rangle, \bar{J}\rangle] = \langle\langle c, \langle\langle\rangle, \langle\rangle, J\rangle\rangle\rangle \underset{V}{+} \text{to-}\underset{V}{\text{idp}}[\langle\bar{J}\rangle]$

$\text{dp-to-}\underset{V}{\text{idp}}[\langle\rangle] = \langle\rangle$

$\text{dp-to-}\underset{V}{\text{idp}}[\langle\langle b, \gamma\rangle, \bar{J}\rangle] = \langle\langle b, \langle\langle\rangle, \gamma, \langle\rangle\rangle\rangle\rangle \underset{V}{+} \text{dp-to-}\underset{V}{\text{idp}}[\langle\bar{J}\rangle]$

$\text{ls-to-}\underset{V}{\text{idp}}[\langle\rangle] = \langle\rangle$

$\text{ls-to-}\underset{V}{\text{idp}}[\langle J, \bar{J}\rangle] = \text{to-}\underset{V}{\text{idp}}[J] \underset{V}{+} \text{ls-to-}\underset{V}{\text{idp}}[\langle\bar{J}\rangle]$

$\underset{V}{\text{exp}}[\langle\rangle] = \langle\rangle$

$\underset{V}{\text{exp}}[a] = \langle\langle \underset{\mathcal{F}}{1}, a\rangle\rangle$

$a \underset{V}{\times} \langle\rangle = \langle\rangle$

$a \underset{V}{\times} \langle\langle c, \langle\alpha, \bar{\alpha}\rangle\rangle, \bar{a}\rangle = \langle\langle c, \langle a, \alpha, \bar{\alpha}\rangle\rangle\rangle \underset{\text{FreeModule}[\mathcal{F},\text{FreeModule}[\mathcal{F},\text{TuplesMonoid}[\mathbb{N}]]]}{+} a \underset{V}{\times} \langle\bar{a}\rangle$

$a \underset{V}{\circ} \langle\rangle = \langle\rangle$

$\langle a\rangle \underset{V}{\circ} \langle\langle c, \langle\alpha, \beta, \langle\rangle\rangle\rangle, \bar{a}\rangle = \langle\langle c, \langle\alpha, \beta, \langle a\rangle\rangle\rangle\rangle \underset{V}{+} \langle a\rangle \underset{V}{\circ} \langle\bar{a}\rangle$

$$\langle a\rangle \underset{V}{\circ} \langle\langle c, \langle \alpha, \beta, \langle\langle b, \gamma\rangle, \bar{J}\rangle\rangle\rangle, \bar{a}\rangle = \langle\langle c, \langle \alpha, \beta, \langle a, \langle b, \gamma\rangle, \bar{J}\rangle\rangle\rangle\rangle \underset{V}{+} \langle a\rangle \underset{V}{\circ} \langle \bar{a}\rangle$$

$$\langle\rangle \underset{V}{\diamond} \alpha = \underset{V}{\mathrm{exp}}[\alpha]$$

$$a \underset{V}{\diamond} \langle\rangle = \underset{V}{\mathrm{exp}}[a]$$

$$\langle a, \bar{J}\rangle \underset{V}{\diamond} \langle \alpha, \bar{\bar{J}}\rangle \text{(* shuffle product *)} = a \underset{V}{\times} \left(\langle \bar{J}\rangle \underset{V}{\diamond} \langle \alpha, \bar{\bar{J}}\rangle\right) \underset{\mathrm{FreeModule}[\mathcal{F},\mathrm{FreeModule}[\mathcal{F},\mathrm{TuplesMonoid}[\mathbb{N}]]]}{+} \alpha \underset{V}{\times} \left(\langle \bar{\bar{J}}\rangle \underset{V}{\diamond} \langle a, \bar{J}\rangle\right)$$

$$\langle\rangle \underset{V}{*} J = \langle\rangle$$

$$J \underset{V}{*} \langle\rangle = \langle\rangle$$

$$\langle\langle b, \langle \alpha, \beta, \langle\rangle\rangle\rangle, \bar{J}\rangle \underset{V}{*} \langle\langle \tilde{b}, \langle \tilde{\alpha}, \tilde{\beta}, \tilde{J}\rangle\rangle, \bar{\bar{J}}\rangle = \left(\left\langle\left\langle b \underset{\mathcal{F}}{*} \tilde{b}, \left\langle \alpha \underset{\mathrm{TuplesMonoid}[\mathbb{N}]}{*} \tilde{\alpha}, \beta \underset{\mathrm{TuplesMonoid}[\mathbb{N}]}{*} \tilde{\beta}, \tilde{J}\right\rangle\right\rangle\right\rangle \underset{V}{+} \langle\langle b, \langle \alpha, \beta, \langle\rangle\rangle\rangle\rangle \underset{V}{*} \langle \bar{\bar{J}}\rangle\right) \underset{V}{+} \langle \bar{J}\rangle \underset{V}{*} \langle\langle \tilde{b}, \langle \tilde{\alpha}, \tilde{\beta}, \tilde{J}\rangle\rangle, \bar{\bar{J}}\rangle$$

$$\langle\langle b, \langle \alpha, \beta, J\rangle\rangle, \bar{J}\rangle \underset{V}{*} \langle\langle \tilde{b}, \langle \tilde{\alpha}, \tilde{\beta}, \tilde{J}\rangle\rangle, \bar{\bar{J}}\rangle = \left(\left(\langle\langle b, \langle \alpha, \beta, \langle\rangle\rangle\rangle\rangle \underset{V}{*} \langle\langle \tilde{b}, \langle \tilde{\alpha}, \tilde{\beta}, \langle\rangle\rangle\rangle\rangle\right) \underset{V}{*} \underset{V}{\text{to-idp}}[J \underset{V}{\diamond} \tilde{J}] \underset{V}{+} \langle\langle b, \langle \alpha, \beta, J\rangle\rangle\rangle \underset{V}{*} \langle \bar{\bar{J}}\rangle\right) \underset{V}{+} \langle \bar{J}\rangle \underset{V}{*} \langle\langle \tilde{b}, \langle \tilde{\alpha}, \tilde{\beta}, \tilde{J}\rangle\rangle, \bar{\bar{J}}\rangle$$

$$\underset{V}{\text{exp-basis}}[\langle\rangle] = \langle\rangle$$

$$\underset{V}{\text{exp-basis}}[\langle\langle c, \langle \alpha, \beta, \langle\rangle\rangle\rangle, \bar{J}\rangle] = \langle\langle c, \langle \alpha, \beta, \langle\rangle\rangle\rangle\rangle \underset{V}{+} \underset{V}{\text{exp-basis}}[\langle \bar{J}\rangle]$$

$$\underset{V}{\text{exp-basis}}[\langle\langle c, \langle \alpha, \beta, \langle\langle\langle\rangle, \gamma\rangle, \bar{J}\rangle\rangle\rangle\rangle] = \langle\rangle$$

$$\underset{V}{\text{exp-basis}}[\langle\langle c, \langle \alpha, \beta, \langle\langle f, \langle 0, 1\rangle\rangle, \bar{J}\rangle\rangle\rangle\rangle] =$$
$$c \underset{V}{\cdot} \underset{V}{\text{exp-basis}}\left[\left\langle\left\langle 1_{\mathcal{F}}, \langle \alpha, \beta, \langle \bar{J}\rangle\rangle\right\rangle\right\rangle \underset{V}{*} \left(\left(\langle\langle f, \langle\langle\rangle, \langle 1\rangle, \langle\rangle\rangle\rangle\rangle \underset{V}{-} \int_V \langle\langle \partial_{\mathcal{F}} f, \langle\langle\rangle, \langle 1\rangle, \langle\rangle\rangle\rangle\rangle\right) \underset{V}{-} \underset{V}{\mathrm{eval}}[\langle\langle f, \langle\langle\rangle, \langle 1\rangle, \langle\rangle\rangle\rangle\rangle, 0]\right)\right]$$

$$\underset{V}{\text{exp-basis}}[\langle\langle c, \langle \alpha, \beta, \langle\langle f, \langle \bar{\alpha}, k, 1\rangle\rangle, \bar{J}\rangle\rangle\rangle\rangle] =$$
$$\left(1 \underset{K}{/} (k+1) \underset{\mathcal{F}}{\cdot} c\right) \underset{V}{\cdot} \underset{V}{\text{exp-basis}}\left[\left\langle\left\langle 1_{\mathcal{F}}, \langle \alpha, \beta, \langle \bar{J}\rangle\rangle\right\rangle\right\rangle \underset{V}{*} \left(\left(\langle\langle f, \langle\langle\rangle, \langle \bar{\alpha}, k+1\rangle, \langle\rangle\rangle\rangle\rangle \underset{V}{-} \int_V \left(\partial_V \left\langle\left\langle 1_{\mathcal{F}}, \langle\langle\rangle, \langle \bar{\alpha}\rangle, \langle\rangle\rangle\right\rangle\right\rangle \underset{V}{*} \left\langle\left\langle 1_{\mathcal{F}}, \left\langle\langle\rangle, \left\langle 0 \,\Big|_{i=0,\ldots,|\langle\bar{\alpha}\rangle|-1}, k+1\right\rangle, \langle\rangle\right\rangle\right\rangle\right\rangle\right)\right) \underset{V}{-} \underset{V}{\mathrm{eval}}\left[\left\langle\left\langle 1_{\mathcal{F}}, \langle\langle\rangle, \langle \bar{\alpha}, k+1\rangle, \langle\rangle\rangle\right\rangle\right\rangle, 0\right]\right)\right]$$

$$\underset{V}{\text{exp-basis}}[\langle\langle c, \langle \alpha, \beta, \langle\langle\langle\langle b, a\rangle\rangle, \gamma\rangle\rangle\rangle\rangle\rangle] = \left\langle\left\langle c \underset{\mathcal{F}}{\cdot} b, \left\langle \alpha, \beta, \left\langle\left\langle\left\langle\left\langle 1_K, a\right\rangle\right\rangle, \gamma\right\rangle\right\rangle\right\rangle\right\rangle\right\rangle$$

$$\underset{V}{\text{exp-basis}}\left[\left\langle\left\langle c, \langle \alpha, \beta, \langle\langle\langle\langle b, a\rangle, \bar{a}\rangle, \gamma\rangle, \bar{J}\rangle\rangle\right\rangle, \bar{\bar{J}}\right\rangle\right] = \left(\left\langle\left\langle\left\langle\left\langle 1_K, a\right\rangle\right\rangle, \gamma\right\rangle\right\rangle \underset{V}{\circ} \underset{V}{\text{exp-basis}}\left[\left\langle\left\langle c \underset{\mathcal{F}}{\cdot} b, \langle \alpha, \beta, \langle \bar{J}\rangle\rangle\right\rangle\right\rangle\right] \underset{V}{+} \underset{V}{\text{exp-basis}}\left[\left\langle\left\langle c, \langle \alpha, \beta, \langle\langle\langle \bar{a}\rangle, \gamma\rangle, \bar{J}\rangle\rangle\right\rangle\right\rangle\right]\right) \underset{V}{+} \underset{V}{\text{exp-basis}}[\langle \bar{\bar{J}}\rangle]$$

$$\partial_V \langle\rangle = \langle\rangle$$

$$\partial_V \langle f, \langle\langle\rangle, \langle\rangle, \langle\rangle\rangle\rangle = \begin{cases} \langle\rangle & \Leftarrow \ \text{is-}\underset{\mathcal{F}}{\text{scal}}[f] \\ \partial_{\mathcal{F}} f \underset{V}{\cdot} 1_V & \Leftarrow \ \text{otherwise} \end{cases}$$

$$\partial_V \langle f, \langle \alpha, \beta, \langle\rangle\rangle\rangle =$$
$$\begin{cases} \underset{V}{\text{dp-to-idp}}\left[\partial_{\mathrm{DiffPolys}[\mathcal{F},K]} \left\langle\left\langle 1_{\mathcal{F}}, \beta\right\rangle\right\rangle\right] \underset{V}{*} \langle\langle f, \langle \alpha, \langle\rangle, \langle\rangle\rangle\rangle\rangle & \Leftarrow \ \text{is-}\underset{\mathcal{F}}{\text{scal}}[f] \\ \partial_{\mathcal{F}} f \underset{V}{\cdot} \left\langle\left\langle 1_{\mathcal{F}}, \langle \alpha, \beta, \langle\rangle\rangle\right\rangle\right\rangle \underset{V}{+} \underset{V}{\text{dp-to-idp}}\left[\partial_{\mathrm{DiffPolys}[\mathcal{F},K]} \left\langle\left\langle 1_{\mathcal{F}}, \beta\right\rangle\right\rangle\right] \underset{V}{*} \langle\langle f, \langle \alpha, \langle\rangle, \langle\rangle\rangle\rangle\rangle & \Leftarrow \ \text{otherwise} \end{cases}$$

$$\partial_V \langle f, \langle \alpha, \beta, \langle\langle b, \gamma\rangle, \bar{J}\rangle\rangle\rangle =$$
$$\begin{cases} \underset{V}{\text{dp-to-idp}}\left[\partial_{\mathrm{DiffPolys}[\mathcal{F},K]} \left\langle\left\langle 1_{\mathcal{F}}, \beta\right\rangle\right\rangle\right] \underset{V}{*} \langle\langle f, \langle \alpha, \langle\rangle, \langle\langle b, \gamma\rangle, \bar{J}\rangle\rangle\rangle\rangle \underset{V}{+} \left\langle\left\langle f \underset{\mathcal{F}}{*} b, \left\langle \alpha, \beta \underset{\mathrm{TuplesMonoid}[\mathbb{N}]}{*} \gamma, \langle \bar{J}\rangle\right\rangle\right\rangle\right\rangle & \Leftarrow \ \text{is-}\underset{\mathcal{F}}{\text{scal}}[f] \\ \left(\langle\langle \partial_{\mathcal{F}} f, \langle \alpha, \beta, \langle\langle b, \gamma\rangle, \bar{J}\rangle\rangle\rangle\rangle \underset{V}{+} \underset{V}{\text{dp-to-idp}}\left[\partial_{\mathrm{DiffPolys}[\mathcal{F},K]} \left\langle\left\langle 1_{\mathcal{F}}, \beta\right\rangle\right\rangle\right] \underset{V}{*} \langle\langle f, \langle \alpha, \langle\rangle, \langle\langle b, \gamma\rangle, \bar{J}\rangle\rangle\rangle\rangle\right) \underset{V}{+} \left\langle\left\langle f \underset{\mathcal{F}}{*} b, \left\langle \alpha, \beta \underset{\mathrm{TuplesMonoid}[\mathbb{N}]}{*} \gamma, \langle \bar{J}\rangle\right\rangle\right\rangle\right\rangle & \Leftarrow \ \text{otherwise} \end{cases}$$

$$\partial_V \left\langle\langle f, \langle \alpha, \beta, J\rangle\rangle, \bar{\bar{J}}\right\rangle = \partial_V \langle f, \langle \alpha, \beta, J\rangle\rangle \underset{V}{+} \partial_V \langle \bar{\bar{J}}\rangle$$

$$\int_V \langle\rangle = \langle\rangle$$

```
∫_V⟨⟨f, ⟨α, ⟨⟩, ⟨⟩⟩⟩⟩ = (∫_𝓕 f) ·_V ⟨⟨1_𝓕, ⟨α, ⟨⟩, ⟨⟩⟩⟩⟩
∫_V⟨⟨f, ⟨α, ⟨⟩, ⟨⟨b, γ⟩, J̄⟩⟩⟩⟩ = (* quasiconstant *) (∫_𝓕 f) ·_V ⟨⟨1_𝓕, ⟨α, ⟨⟩, ⟨⟨b, γ⟩, J̄⟩⟩⟩⟩ -_V
  (∫_V⟨⟨b *_𝓕 ∫_𝓕 f, ⟨α, γ, ⟨J̄⟩⟩⟩⟩)
∫_V⟨⟨f, ⟨α, ⟨0, 1⟩, ⟨⟩⟩⟩⟩ = (* quasilinear base case *)
  (⟨⟨f, ⟨α, ⟨1⟩, ⟨⟩⟩⟩⟩ -_V ∫_V((∂_𝓕 f) ·_V ⟨⟨1_𝓕, ⟨α, ⟨1⟩, ⟨⟩⟩⟩⟩)) -_V eval_V[⟨⟨f, ⟨α, ⟨1⟩, ⟨⟩⟩⟩⟩, 0]
∫_V⟨⟨f, ⟨α, ⟨0, 1⟩, ⟨⟨b, γ⟩, J̄⟩⟩⟩⟩ = (* quasilinear base case *)
  (⟨⟨f, ⟨α, ⟨1⟩, ⟨⟨b, γ⟩, J̄⟩⟩⟩⟩ -_V ∫_V((∂_𝓕 f) ·_V ⟨⟨1_𝓕, ⟨α, ⟨1⟩, ⟨⟨b, γ⟩, J̄⟩⟩⟩⟩)) -_V ∫_V⟨⟨f *_𝓕 b, ⟨α, ⟨1⟩ *_TuplesMonoid[ℕ] γ, ⟨J̄⟩⟩
∫_V⟨⟨f, ⟨α, ⟨ᾱ, k, 1⟩, ⟨⟩⟩⟩⟩ = (* quasilinear *)
  (1 /_K (k + 1) ·_𝓕 1_𝓕) ·_V ((⟨⟨f, ⟨α, ⟨ᾱ, k + 1⟩, ⟨⟩⟩⟩⟩ -_V ∫_V(∂_V⟨⟨f, ⟨α, ⟨ᾱ⟩, ⟨⟩⟩⟩⟩ *_V ⟨⟨1_𝓕, ⟨⟨⟩, ⟨0 | i=0,..,|⟨ᾱ⟩|-1, k + 1⟩, ⟨⟩⟩⟩⟩)) -_V
    eval_V[⟨⟨f, ⟨α, ⟨ᾱ, k + 1⟩, ⟨⟩⟩⟩⟩, 0])
∫_V⟨⟨f, ⟨α, ⟨ᾱ, k, 1⟩, J⟩⟩⟩ = (* quasilinear *)
  (1 /_K (k + 1) ·_𝓕 1_𝓕) ·_V (⟨⟨f, ⟨α, ⟨ᾱ, k + 1⟩, J⟩⟩⟩ -_V ∫_V(∂_V⟨⟨f, ⟨α, ⟨ᾱ⟩, J⟩⟩⟩ *_V ⟨⟨1_𝓕, ⟨⟨⟩, ⟨0 | i=0,..,|⟨ᾱ⟩|-1, k + 1⟩, ⟨⟩⟩⟩⟩))
∫_V⟨⟨f, ⟨α, β, ⟨J̄⟩⟩⟩⟩ = (* functional *) ⟨⟨1_𝓕, ⟨α, ⟨⟩, ⟨⟨f, β⟩, J̄⟩⟩⟩⟩
∫_V⟨J, J̄⟩ = ∫_V⟨J⟩ +_V ∫_V⟨J̄⟩
PrettyPrint_V[p] = ™Plus[PrettyPrintM_V[p_i] | i=1,..,|p|]
PrettyPrintV_V[p] = ™Plus[PrettyPrintMV_V[p_i] | i=1,..,|p|]
PrettyPrint_V[⟨p⟩] = PrettyPrintM_V[p]
PrettyPrintV_V[⟨p⟩] = PrettyPrintMV_V[p]
PrettyPrint_V[⟨⟩] = 0
PrettyPrintV_V[⟨⟩] = 0
PrettyPrintM_V[⟨c, w⟩] = tma-trfm[™Times[PrettyPrint_𝓕[c], PrettyPrintW_V[w]]]
PrettyPrintMV_V[⟨c, w⟩] = tma-trfm[™Times[PrettyPrint_𝓕[c], PrettyPrintWV_V[w]]]
PrettyPrintW_V[⟨⟩] = 1
PrettyPrintWV_V[⟨⟩] = 1
PrettyPrintW_V[⟨⟨⟩, ⟨⟩, J⟩] = PrettyPrintL_V[J]
PrettyPrintWV_V[⟨⟨⟩, ⟨⟩, J⟩] = PrettyPrintLV_V[J]
PrettyPrintW_V[⟨⟨⟩, β, J⟩] = mytimes[™Power[u, β], PrettyPrintL_V[J]]
PrettyPrintW V_V[⟨⟨⟩, β, J⟩] = mytimes[™Power[v, β], PrettyPrintLV_V[J]]
PrettyPrintW_V[⟨α, ⟨⟩, J⟩] = mytimes[™Power[u[0], α], PrettyPrintL_V[J]]
PrettyPrintWV_V[⟨α, ⟨⟩, J⟩] = mytimes[™Power[v[0], α], PrettyPrintLV_V[J]]
PrettyPrintW_V[⟨α, β, J⟩] = mytimes[™Power[u[0], α], ™Power[u, β], PrettyPrintL_V[J]]
PrettyPrintWV_V[⟨α, β, J⟩] = mytimes[™Power[v[0], α], ™Power[v, β], PrettyPrintLV_V[J]]
PrettyPrintL_V[⟨⟩] = ""
PrettyPrintLV_V[⟨⟩] = ""
PrettyPrintL_V[⟨⟨b, γ⟩, J̄⟩] = mytimes[mytimes["∫", PrettyPrint_𝓕[b]], ™Power[u, γ], PrettyPrintL_V[⟨J̄⟩]]
PrettyPrintLV_V[⟨⟨b, γ⟩, J̄⟩] = mytimes[mytimes["∫", PrettyPrint_𝓕[b]], ™Power[v, γ], PrettyPrintLV_V[⟨J̄⟩]]]
```

```
Definition["PrettyPrinting", any[ℱ, K],
 PrettyPrinting[ℱ, K] = Functor[P, any[b, c, m, n, p, r, w, α, β, λ, γ, η, J, L, η̄, ξ̄, J̄],

   𝕤 = ⟨⟩
   ──────────────────────────────────────────
   PrettyPrintW_P[⟨⟩] = ""

   PrettyPrintW_P[⟨λ, η, η̄⟩] = mytimes[PrettyPrintL_P[λ], PrettyPrintW_P[⟨η, η̄⟩]]

   PrettyPrintW_P[⟨η⟩] = PrettyPrintL_P[η]

   PrettyPrintF_P[⟨⟨⟩, ⟨⟩, ⟨⟩⟩] = ""

   PrettyPrintF_P[⟨α, ⟨⟩, ⟨⟩⟩] = ™Power[v[0], α]

   PrettyPrintF_P[⟨⟨⟩, β, ⟨⟩⟩] = mytimes[™Power[v, β]]

   PrettyPrintF_P[⟨α, β, ⟨⟩⟩] = mytimes[™Power[v[0], α], ™Power[v, β]]

   PrettyPrintF_P[⟨⟨⟩, ⟨⟩, J⟩] = PrettyPrintJ_P[J]

   PrettyPrintF_P[⟨α, β, J⟩] = mytimes[™Power[v[0], α], ™Power[v, β], PrettyPrintJ_P[J]]

   PrettyPrintL_P[⟨"⌈⌉", ⟨⟨c, ⟨α, β, J⟩⟩⟩⟩] = mytimes[ PrettyPrint_IntDiffPolys[𝔼xp,𝕂] [c], PrettyPrintF_P[⟨α, β, J⟩]]

   PrettyPrintL_P[⟨"⌊⌋", 0⟩] = "E"

   PrettyPrintJ_P[⟨⟩] = ""

   PrettyPrintJ_P[⟨⟨b, γ⟩, J̄⟩] = mytimes[mytimes["∫", PrettyPrintV_IntDiffPolys[𝔼xp,𝕂] [b]], ™Power[v, γ], PrettyPrintJ_P[⟨J̄⟩]]

   PrettyPrintL_P["∂"] = "∂"

   PrettyPrintL_P["∫"] = "∫"

   PrettyPrintL_P[⟨⟩] = ""

   PrettyPrintL_P[η] = η

   PrettyPrintM_P[⟨⟩] = 0

   PrettyPrintM_P[⟨c, ⟨⟩⟩] = c

   PrettyPrintM_P[⟨c, w⟩] = mytimes[PrettyPrint_K[c], PrettyPrintW_P[w]]

   PrettyPrintRule_P[⟨m, n⟩] = PrettyPrint_P[m] → PrettyPrint_P[n]

   PrettyPrintRules_P[L] = ⟨PrettyPrintRule_P[L_i] | i=1,..,|L|⟩

   PrettyPrint_P[⟨⟨⟨c, η⟩, ξ̄⟩, η̄⟩] = ⟨PrettyPrint_P[⟨⟨⟨c, η⟩, ξ̄⟩, η̄⟩_i] | i=1,..,|⟨⟨⟨c,η⟩,ξ̄⟩,η̄⟩|⟩

   PrettyPrint_P[p] = ™Plus[PrettyPrintM_P[p_i] | i=1,..,|p|]

   PrettyPrint_P[⟨m⟩] = PrettyPrintM_P[m]

   PrettyPrint_P[⟨⟩] = 0

 ]]
```

Translator IDP

Translator Exponential Polynomials

```
Use[⟨Built-in["Tuples"], Built-in["Quantifiers"],
  Built-in["Connectives"], Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"]⟩]
```

```
Definition["Parsing Exp", any[𝓕],
 Parsing[𝓕] = Functor[P, any[c, m, n, r, λ, m̄, n̄],

    𝕤 = ⟨⟩
    ─────────────────────────────────────────────
    ParsePol_P[] = ⟨⟩

    ParsePol_P[™Tuple[m̄]] = ⟨ParsePol_P[⟨m̄⟩_i]  |  i=1,…,|⟨m̄⟩| ⟩

    ParsePol_P[™Plus[]] = ⟨⟩

    ParsePol_P[™Plus[m, n̄]] = ParsePol_P[m] ⌣ ParsePol_P[™Plus[n̄]]

    ParsePol_P[™Times[™Minus[c], m̄]] = ParseTerm_P[™Times[™Minus[c], m̄]]

    ParsePol_P[™Times[c, m̄]] = ParseTerm_P[™Times[c, m̄]]

    ParsePol_P[™Minus[™Times[™Minus[c], m̄]]] = ParseTerm_P[™Times[c, m̄]]

    ParsePol_P[™Minus[™Times[c, m̄]]] = ParseTerm_P[™Times[™Minus[c], m̄]]

    ParsePol_P[™Minus[m, n]] = ParsePol_P[m] ⌣ ParsePol_P[™Minus[n]]

    ParsePol_P[™Minus[m, n, r, m̄]] = ParsePol_P[m] ⌣ ParsePol_P[™Minus[™Minus[n], r, m̄]]

    ParsePol_P[™Minus[c]] = { ⟨⟨−c, ⟨⟩⟩⟩                  ⇐ ™IsNumber[c]
                            { ParseTerm_P[™Minus[c]]     ⇐ otherwise

    ParsePol_P[c] = { ⟨⟨c, ⟨⟩⟩⟩        ⇐ ™IsNumber[c]
                    { ParseTerm_P[c]   ⇐ otherwise

    ParseTerm_P[™Minus[c]] = { ⟨⟨−c, ⟨⟩⟩⟩              ⇐ ™IsNumber[c]
                             { −1 ·_𝓕 ParseMon_P[c]   ⇐ otherwise

    ParseTerm_P[™Times[™Times[m̄]]] = ParseTerm_P[™Times[m̄]]

    ParseTerm_P[™Times[™Minus[c], m̄]] = { −c ·_𝓕 ParseTerm_P[™Times[m̄]]      ⇐ ™IsNumber[c]
                                        { −1 ·_𝓕 ParseTerm_P[™Times[c, m̄]]   ⇐ otherwise

    ParseTerm_P[™Times[c, m̄]] = { c ·_𝓕 ParseTerm_P[™Times[m̄]]   ⇐ ™IsNumber[c]
                                { ParseMon_P[™Times[c, m̄]]      ⇐ otherwise

    ParseTerm_P[c] = { ⟨⟨c, ⟨⟩⟩⟩        ⇐ ™IsNumber[c]
                     { ParseMon_P[c]    ⇐ otherwise

    ParseTerm_P[™Times[]] = ⟨⟩

    ParseMon_P[™Times[™Power[x, n], ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨n, λ⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Times[™Power[x, n], ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨n, λ⟩⟩⟩⟩⟩
                                                                                            ]]
    ParseMon_P[™Times[™Power[x, n], ™Power[E, x]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨n, 1⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Times[™Power[x, n], ™Power[E, x]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨n, 1⟩⟩⟩⟩⟩

    ParseMon_P[™Times[x, ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨1, λ⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Times[x, ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨1, λ⟩⟩⟩⟩⟩

    ParseMon_P[™Power[E, ™Times[λ, x]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨0, λ⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨0, λ⟩⟩⟩⟩⟩

    ParseMon_P[™Power[x, n]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨n, 0⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Power[x, n]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨n, 0⟩⟩⟩⟩⟩

    ParseMon_P[™Power[E, x]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨0, 1⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Power[E, x]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨0, 1⟩⟩⟩⟩⟩

    ParseMon_P[™Power[E, ™Minus[x]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨0, −1⟩⟩⟩⟩⟩

    ParseMon_P[™Times[™Power[E, ™Minus[x]]]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨0, −1⟩⟩⟩⟩⟩

    ParseMon_P[x] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨1, 0⟩⟩⟩⟩⟩

    ParseMon_P[™Times[x]] = ⟨⟨1, ⟨⟨"⌈⌉", ⟨1, 0⟩⟩⟩⟩⟩

    ParseMon_P[™Floor[c]] = ⟨⟨1, ⟨⟨"⌊⌋", c⟩⟩⟩⟩

    ParseMon_P[™Times[™Floor[c]]] = ⟨⟨1, ⟨⟨"⌊⌋", c⟩⟩⟩⟩
```

```
ParseMon_P[™D] = ⟨⟨1, ⟨"∂"⟩⟩⟩
ParseMon_P[™Times[™D]] = ⟨⟨1, ⟨"∂"⟩⟩⟩
ParseMon_P[™Power[m, 0]] = ⟨⟩
ParseMon_P[™Power[m, 1]] = ParseMon_P[m]
ParseMon_P[™Power[m, n]] = ParseMon_P[m] ⋆_𝓕 ParseMon_P[™Power[m, n -_MmaAuxF[] 1]]
ParseMon_P[™Times[™Power[m, n]]] = ParseMon_P[™Power[m, n]]
ParseMon_P[™Times[™Times[m̄]]] = ParseMon_P[™Times[m̄]]
ParseMon_P[™Times[m, n, n̄]] = ParseMon_P[m] ⋆_𝓕 ParseMon_P[™Times[n, n̄]]
ParseMon_P[c] = ⟨⟨c, ⟨⟩⟩⟩
ParseMon_P[] = ⟨⟩
```

```
Definition["Parsing", any[𝓕],
 Parsing[𝓕] = Functor[P, any[c, m, n, r, λ, m̄, n̄],
    𝕤 = ⟨⟩
    ─────────────────────────────────────────
    ParsePol_P[] = ⟨⟩
    ParsePol_P[™Tuple[m̄]] = ⟨ParsePol_P[⟨m̄⟩_i] | i=1,..,|⟨m̄⟩| ⟩
    ParsePol_P[™Plus[]] = ⟨⟩
    ParsePol_P[™Plus[m, n̄]] = ParsePol_P[m] ⊨ ParsePol_P[™Plus[n̄]]
    ParsePol_P[™Times[™Minus[c], m̄]] = ParseTerm_P[™Times[™Minus[c], m̄]]
    ParsePol_P[™Times[c, m̄]] = ParseTerm_P[™Times[c, m̄]]
    ParsePol_P[™Minus[™Times[™Minus[c], m̄]]] = ParseTerm_P[™Times[c, m̄]]
    ParsePol_P[™Minus[™Times[c, m̄]]] = ParseTerm_P[™Times[™Minus[c], m̄]]
    ParsePol_P[™Minus[m, n]] = ParsePol_P[m] ≍ ParsePol_P[™Minus[n]]
    ParsePol_P[™Minus[m, n, r, m̄]] = ParsePol_P[m] ≍ ParsePol_P[™Minus[™Minus[n], r, m̄]]
    ParsePol_P[™Minus[c]] = { ⟨⟨-c, ⟨⟩⟩⟩                 ⇐ ™IsNumber[c]
                            { ParseTerm_P[™Minus[c]]    ⇐ otherwise
    ParsePol_P[c] = { ⟨⟨c, ⟨⟩⟩⟩         ⇐ ™IsNumber[c]
                    { ParseTerm_P[c]    ⇐ otherwise
    ParseTerm_P[™Minus[c]] = { ⟨⟨-c, ⟨⟩⟩⟩              ⇐ ™IsNumber[c]
                             { -1 ·_𝓕 ParseMon_P[c]    ⇐ otherwise
    ParseTerm_P[™Times[™Times[m̄]]] = ParseTerm_P[™Times[m̄]]
    ParseTerm_P[™Times[™Minus[c], m̄]] = { -c ·_𝓕 ParseTerm_P[™Times[m̄]]      ⇐ ™IsNumber[c]
                                        { -1 ·_𝓕 ParseTerm_P[™Times[c, m̄]]   ⇐ otherwise
    ParseTerm_P[™Times[c, m̄]] = { c ·_𝓕 ParseTerm_P[™Times[m̄]]   ⇐ ™IsNumber[c]
                                { ParseMon_P[™Times[c, m̄]]       ⇐ otherwise
    ParseTerm_P[c] = { ⟨⟨c, ⟨⟩⟩⟩         ⇐ ™IsNumber[c]
                     { ParseMon_P[c]     ⇐ otherwise
    ParseTerm_P[™Times[]] = ⟨⟩                                            ]]
    ParseMon_P[™Times[™Power[x, n], ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨n, λ⟩⟩⟩
    ParseMon_P[™Times[™Times[™Power[x, n], ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨n, λ⟩⟩⟩
    ParseMon_P[™Times[™Power[x, n], ™Power[E, x]]] = ⟨⟨1, ⟨n, 1⟩⟩⟩
    ParseMon_P[™Times[™Times[™Power[x, n], ™Power[E, x]]]] = ⟨⟨1, ⟨n, 1⟩⟩⟩
    ParseMon_P[™Times[x, ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨1, λ⟩⟩⟩
    ParseMon_P[™Times[™Times[x, ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨1, λ⟩⟩⟩
    ParseMon_P[™Power[E, ™Times[λ, x]]] = ⟨⟨1, ⟨0, λ⟩⟩⟩
```

```
ParseMon_P[™Times[™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨0, λ⟩⟩⟩
ParseMon_P[™Power[x, n]] = ⟨⟨1, ⟨n, 0⟩⟩⟩
ParseMon_P[™Times[™Power[x, n]]] = ⟨⟨1, ⟨n, 0⟩⟩⟩
ParseMon_P[™Power[E, x]] = ⟨⟨1, ⟨0, 1⟩⟩⟩
ParseMon_P[™Times[™Power[E, x]]] = ⟨⟨1, ⟨0, 1⟩⟩⟩
ParseMon_P[™Power[E, ™Minus[x]]] = ⟨⟨1, ⟨0, -1⟩⟩⟩
ParseMon_P[™Times[™Power[E, ™Minus[x]]]] = ⟨⟨1, ⟨0, -1⟩⟩⟩
ParseMon_P[x] = ⟨⟨1, ⟨1, 0⟩⟩⟩
ParseMon_P[™Times[x]] = ⟨⟨1, ⟨1, 0⟩⟩⟩
ParseMon_P[™Power[m, 0]] = ⟨⟩
ParseMon_P[™Power[m, 1]] = ParseMon_P[m]
ParseMon_P[™Power[m, n]] = ParseMon_P[m] *_F ParseMon_P[™Power[m, n -_MmaAuxF[] 1]]
ParseMon_P[™Times[™Power[m, n]]] = ParseMon_P[™Power[m, n]]
ParseMon_P[™Times[™Times[m̄]]] = ParseMon_P[™Times[m̄]]
ParseMon_P[™Times[m, n, n̄]] = ParseMon_P[m] *_F ParseMon_P[™Times[n, n̄]]
ParseMon_P[c] = ⟨⟨c, ⟨⟩⟩⟩
ParseMon_P[] = ⟨⟩
```

```
Definition["ParsingF", any[F],
 ParsingF[F] = Functor[P, any[c, m, n, r, λ, m̄, n̄],
    s = ⟨⟩
    ─────────────────────────────────────────
    ParsePolF_P[] = ⟨⟩
    ParsePolF_P[0] = ⟨⟩
    ParsePolF_P[™Tuple[m̄]] = ⟨ParsePolF_P[⟨m̄⟩_i] | i=1,…,|⟨m̄⟩| ⟩
    ParsePolF_P[™Plus[]] = ⟨⟩
    ParsePolF_P[™Plus[m, n̄]] = ParsePolF_P[m] ⍭ ParsePolF_P[™Plus[n̄]]
    ParsePolF_P[™Times[™Minus[c], m̄]] = ParseTermF_P[™Times[™Minus[c], m̄]]
    ParsePolF_P[™Times[c, m̄]] = ParseTermF_P[™Times[c, m̄]]
    ParsePolF_P[™Minus[™Times[™Minus[c], m̄]]] = ParseTermF_P[™Times[c, m̄]]
    ParsePolF_P[™Minus[™Times[c, m̄]]] = ParseTermF_P[™Times[™Minus[c], m̄]]
    ParsePolF_P[™Minus[m, n]] = ParsePolF_P[m] ⍭ ParsePolF_P[™Minus[n]]
    ParsePolF_P[™Minus[m, n, r, m̄]] = ParsePolF_P[m] ⍭ ParsePolF_P[™Minus[™Minus[n], r, m̄]]
    ParsePolF_P[™Minus[c]] = { ⟨⟨-c, ⟨⟩⟩⟩                 ⇐ ™IsNumber[c]
                             { ParseTermF_P[™Minus[c]]  ⇐ otherwise
    ParsePolF_P[c] = { ⟨⟨c, ⟨⟩⟩⟩          ⇐ ™IsNumber[c]
                     { ParseTermF_P[c]   ⇐ otherwise
    ParseTermF_P[™Minus[c]] = { ⟨⟨-c, ⟨⟩⟩⟩                 ⇐ ™IsNumber[c]
                              { -1 ·_F ParseMonF_P[c]     ⇐ otherwise
    ParseTermF_P[™Times[™Times[m̄]]] = ParseTermF_P[™Times[m̄]]
    ParseTermF_P[™Times[™Minus[c], m̄]] = { -c ·_F ParseTermF_P[™Times[m̄]]     ⇐ ™IsNumber[c]
                                         { -1 ·_F ParseTermF_P[™Times[c, m̄]]  ⇐ otherwise
    ParseTermF_P[™Times[c, m̄]] = { c ·_F ParseTermF_P[™Times[m̄]]   ⇐ ™IsNumber[c]        ]]
                                 { ParseMonF_P[™Times[c, m̄]]       ⇐ otherwise
```

```
                      ⟨⟨c, ⟨⟩⟩⟩        ⇐  ™IsNumber[c]
ParseTermF_P[c] = {
                      ParseMonF_P[c]   ⇐  otherwise

ParseTermF_P[™Times[]] = ⟨⟩

ParseMonF_P[™Times[™Power[x, m], ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩

ParseMonF_P[™Times[™Power[E, ™Times[λ, x]], ™Power[x, m]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩

ParseMonF_P[™Times[™Power[x, m], ™Power[E, x]]] = ⟨⟨1, ⟨m, 1⟩⟩⟩

ParseMonF_P[™Times[x, ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨1, λ⟩⟩⟩

ParseMonF_P[™Times[™Times[™Power[x, m], ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩

ParseMonF_P[™Times[™Times[™Power[E, ™Times[λ, x]], ™Power[x, m]]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩

ParseMonF_P[™Times[™Times[™Power[x, m], ™Power[E, x]]]] = ⟨⟨1, ⟨m, 1⟩⟩⟩

ParseMonF_P[™Times[™Times[x, ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨1, λ⟩⟩⟩

ParseMonF_P[™Power[x, m]] = ⟨⟨1, ⟨m, 0⟩⟩⟩

ParseMonF_P[™Times[™Power[x, m]]] = ⟨⟨1, ⟨m, 0⟩⟩⟩

ParseMonF_P[™Power[E, ™Times[λ, x]]] = ⟨⟨1, ⟨0, λ⟩⟩⟩

ParseMonF_P[™Times[™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨0, λ⟩⟩⟩

ParseMonF_P[x] = ⟨⟨1, ⟨1, 0⟩⟩⟩

ParseMonF_P[™Times[x]] = ⟨⟨1, ⟨1, 0⟩⟩⟩

ParseMonF_P[™Power[E, x]] = ⟨⟨1, ⟨0, 1⟩⟩⟩

ParseMonF_P[™Times[™Power[E, x]]] = ⟨⟨1, ⟨0, 1⟩⟩⟩

ParseMonF_P[] = ⟨⟩
```

```
Definition["Parsing Diff Polys", any[ℱ],
 ParsingDP[ℱ] = Functor[P, any[c, k, l, m, n, r, λ, c̄, d̄, m̄, n̄],

    𝕤 = ⟨⟩
    ______________________________________________________________
    ParsePolF_P[] = ⟨⟩

    ParsePolF_P[0] = ⟨⟩

    ParsePolF_P[™Tuple[m̄]] = ⟨ParsePolF_P[⟨m̄⟩_i] | i=1,..,|⟨m̄⟩| ⟩

    ParsePolF_P[™Plus[]] = ⟨⟩

    ParsePolF_P[™Plus[m, n̄]] = ParsePolF_P[m] ≍ ParsePolF_P[™Plus[n̄]]

    ParsePolF_P[™Times[™Minus[c], m̄]] = ParseTermF_P[™Times[™Minus[c], m̄]]

    ParsePolF_P[™Times[c, m̄]] = ParseTermF_P[™Times[c, m̄]]

    ParsePolF_P[™Minus[™Times[™Minus[c], m̄]]] = ParseTermF_P[™Times[c, m̄]]

    ParsePolF_P[™Minus[™Times[c, m̄]]] = ParseTermF_P[™Times[™Minus[c], m̄]]
                                                                              ]]
    ParsePolF_P[™Minus[m, n]] = ParsePolF_P[m] ≍ ParsePolF_P[™Minus[n]]

    ParsePolF_P[™Minus[m, n, r, m̄]] = ParsePolF_P[m] ≍ ParsePolF_P[™Minus[™Minus[n], r, m̄]]

                                  ⟨⟨-c, ⟨0, 0⟩⟩⟩               ⇐  ™IsNumber[c]
    ParsePolF_P[™Minus[c]] = {
                                  ParseTermF_P[™Minus[c]]      ⇐  otherwise

                         ⟨⟨c, ⟨0, 0⟩⟩⟩      ⇐  ™IsNumber[c]
    ParsePolF_P[c] = {
                         ParseTermF_P[c]    ⇐  otherwise

                                    ⟨⟨-c, ⟨0, 0⟩⟩⟩             ⇐  ™IsNumber[c]
    ParseTermF_P[™Minus[c]] = {
                                    -1 ·_ℱ ParseMonF_P[c]      ⇐  otherwise

    ParseTermF_P[™Times[™Times[m̄]]] = ParseTermF_P[™Times[m̄]]
```

```
ParseTermF_P[™Times[™Minus[c], m̄]] = { -c ·_F ParseTermF_P[™Times[m̄]]    ⇐ ™IsNumber[c]
                                       { -1 ·_F ParseTermF_P[™Times[c, m̄]] ⇐ otherwise

ParseTermF_P[™Times[c, m̄]] = { c ·_F ParseTermF_P[™Times[m̄]] ⇐ ™IsNumber[c]
                               { ParseMonF_P[™Times[c, m̄]]   ⇐ otherwise

ParseTermF_P[c] = { ⟨⟨c, ⟨0, 0⟩⟩⟩ ⇐ ™IsNumber[c]
                  { ParseMonF_P[c] ⇐ otherwise

ParseTermF_P[™Times[]] = ⟨⟩

ParseMonF_P[™Times[™Power[x, m], ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩
ParseMonF_P[™Times[™Power[E, ™Times[λ, x]], ™Power[x, m]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩
ParseMonF_P[™Times[™Power[x, m], ™Power[E, x]]] = ⟨⟨1, ⟨m, 1⟩⟩⟩
ParseMonF_P[™Times[x, ™Power[E, x]]] = ⟨⟨1, ⟨1, 1⟩⟩⟩
ParseMonF_P[™Times[x, ™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨1, λ⟩⟩⟩
ParseMonF_P[™Times[™Times[™Power[x, m], ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩
ParseMonF_P[™Times[™Times[™Power[E, ™Times[λ, x]], ™Power[x, m]]]] = ⟨⟨1, ⟨m, λ⟩⟩⟩
ParseMonF_P[™Times[™Times[™Power[x, m], ™Power[E, x]]]] = ⟨⟨1, ⟨m, 1⟩⟩⟩
ParseMonF_P[™Times[™Times[x, ™Power[E, ™Times[λ, x]]]]] = ⟨⟨1, ⟨1, λ⟩⟩⟩
ParseMonF_P[™Power[x, m]] = ⟨⟨1, ⟨m, 0⟩⟩⟩
ParseMonF_P[™Times[™Power[x, m]]] = ⟨⟨1, ⟨m, 0⟩⟩⟩
ParseMonF_P[™Times[c, ™Power[x, m]]] = ⟨⟨c, ⟨m, 0⟩⟩⟩
ParseMonF_P[™Power[E, ™Times[λ, x]]] = ⟨⟨1, ⟨0, λ⟩⟩⟩
ParseMonF_P[™Times[™Power[E, ™Times[λ, x]]]] = ⟨⟨1, ⟨0, λ⟩⟩⟩
ParseMonF_P[x] = ⟨⟨1, ⟨1, 0⟩⟩⟩
ParseMonF_P[E] = ⟨⟨1, ⟨0, 1⟩⟩⟩
ParseMonF_P[™Times[x]] = ⟨⟨1, ⟨1, 0⟩⟩⟩
ParseMonF_P[™Power[E, x]] = ⟨⟨1, ⟨0, 1⟩⟩⟩
ParseMonF_P[™Times[™Power[E, x]]] = ⟨⟨1, ⟨0, 1⟩⟩⟩
ParseMonF_P[™Times[c]] = ParsePolF_P[c]
ParseMonF_P[c] = ⟨⟨c, ⟨0, 0⟩⟩⟩
ParseMonF_P[] = ⟨⟩

ParseDP_P[] = ⟨⟩
ParseDP_P[™Plus[]] = ⟨⟩
ParseDP_P[™Plus[m, n̄]] = ParseMonDP_P[m] ≍ ParseDP_P[™Plus[n̄]]
ParseDP_P[™Power[u, k]] = ⟨⟨1_F, k⟩⟩
ParseDP_P[™Times[c̄, ™Power[u, k]]] = ParseMonDP_P[™Times[c̄, ™Power[u, k]]]
ParseMonDP_P[™Times[c̄, ™Power[u, k]]] = where[p = ParsePolF_P[™Times[c̄]],
  ⟨⟨p, k⟩⟩]
ParseMonDP_P[™Power[u, k]] = ⟨⟨1_F, k⟩⟩

ParseIDP_P[] = ⟨⟩
ParseIDP_P[™Plus[]] = ⟨⟩
ParseIDP_P[™Plus[m, n̄]] = ParseIDP_P[m] ≍ ParseIDP_P[™Plus[n̄]]
ParseIDP_P[™Times[™Minus[c], m̄]] = ParseMonIDP_P[™Times[™Minus[c], m̄]]
ParseIDP_P[™Minus[™Times[™Minus[c], m̄]]] = ParseMonIDP_P[™Times[c, m̄]]
ParseIDP_P[™Minus[™Times[c, m̄]]] = ParseMonIDP_P[™Times[™Minus[c], m̄]]
```

```
ParseIDP_P[™Minus[m, n]] = ParseIDP_P[m] ⊕ ParseIDP_P[™Minus[n]]
ParseIDP_P[™Minus[m, n, r, m̄]] = ParseIDP_P[m] ⊕ ParseIDP_P[™Minus[™Minus[n], r, m̄]]
ParseIDP_P[™Minus[c]] = { ⟨⟨ParsePolF_P[-c], ⟨0, 0⟩⟩⟩   ⇐ ∈_𝓕[c]
                        { ParseMonIDP_P[™Minus[c]]      ⇐ otherwise
ParseMonIDP_P[™Minus[c]] = { ⟨⟨ParsePolF_P[-c], ⟨0, 0⟩⟩⟩   ⇐ ∈_𝓕[c]
                           { (-1_IntDiffPolys[𝓕,𝕂]) ·_IntDiffPolys[𝓕,𝕂] ParseMonIDP_P[c]   ⇐ otherwise
ParseIDP_P[™Times[c, d̄]] = ParseMonIDP_P[™Times[c, d̄]]
ParseIDP_P[™Power[u, k]] = ⟨⟨1_𝓕, ⟨⟨⟩, k, ⟨⟩⟩⟩⟩
ParseIDP_P[u] = ⟨⟨1_𝓕, ⟨⟨⟩, ⟨1⟩, ⟨⟩⟩⟩⟩
ParseIDP_P[™Power[u[0], k]] = ⟨⟨1_𝓕, ⟨k, ⟨⟩, ⟨⟩⟩⟩⟩
ParseIDP_P[u[0]] = ⟨⟨1_𝓕, ⟨⟨1⟩, ⟨⟩, ⟨⟩⟩⟩⟩
ParseIDP_P[™Times[c, ™Power[u, k]]] = ParseMonIDP_P[™Times[c, ™Power[u, k]]]
ParseIDP_P[c] = ⟨⟨ParsePolF_P[c], ⟨⟨⟩, ⟨⟩, ⟨⟩⟩⟩⟩
ParseMonIDP_P[™Minus[c]] = { ⟨⟨ParsePolF_P[-c], ⟨0, 0⟩⟩⟩   ⇐ ∈_𝓕[c]
                           { −_IntDiffPolys[𝓕,𝕂] ParseMonIDP_P[c]   ⇐ otherwise
ParseMonIDP_P[™Times[™Minus[c], m̄]] = { ParsePolF_P[-c] ·_IntDiffPolys[𝓕,𝕂] ParseMonIDP_P[™Times[m̄]]   ⇐ ∈_𝓕[c]
                                      { −_IntDiffPolys[𝓕,𝕂] ParseMonIDP_P[™Times[c, m̄]]   ⇐ otherwise
ParseMonIDP_P[™Power[u, k]] = ⟨⟨1_𝓕, k⟩⟩
ParseMonIDP_P[™Times[c, ™Power[u[0], k], ™Power[u, l], d̄]] = where[p = ParsePolF_P[c],
  ⟨⟨p, ⟨k, l, ParseFunc_P[™Times[d̄]]⟩⟩⟩]
ParseMonIDP_P[™Times[™Power[u[0], k], ™Power[u, l], d̄]] = ⟨⟨1_𝓕, ⟨k, l, ParseFunc_P[™Times[d̄]]⟩⟩⟩
ParseMonIDP_P[™Times[c, ™Power[u[0], k], "∫", d̄]] = where[p = ParsePolF_P[c],
  ⟨⟨p, ⟨k, ⟨⟩, ParseFunc_P[™Times["∫", d̄]]⟩⟩⟩]
ParseMonIDP_P[™Times[™Power[u[0], k], "∫", d̄]] = ⟨⟨1_𝓕, ⟨k, ⟨⟩, ParseFunc_P[™Times["∫", d̄]]⟩⟩⟩
ParseMonIDP_P[™Times[c, ™Power[u, l], "∫", d̄]] = where[p = ParsePolF_P[c],
  ⟨⟨p, ⟨⟨⟩, l, ParseFunc_P[™Times["∫", d̄]]⟩⟩⟩]
ParseMonIDP_P[™Times[™Power[u, l], "∫", d̄]] = ⟨⟨1_𝓕, ⟨⟨⟩, l, ParseFunc_P[™Times["∫", d̄]]⟩⟩⟩
ParseMonIDP_P[™Times[c, ™Power[u[0], k]]] = where[p = ParsePolF_P[c],
  ⟨⟨p, ⟨k, ⟨⟩, ⟨⟩⟩⟩⟩]
ParseMonIDP_P[™Times[™Power[u[0], k], c̄]] = ⟨⟨1_𝓕, ⟨k, ⟨⟩, ParseFunc_P[™Times[c̄]]⟩⟩⟩
ParseMonIDP_P[™Times[c, u[0]]] = where[p = ParsePolF_P[c],
  ⟨⟨p, ⟨⟨1⟩, ⟨⟩, ⟨⟩⟩⟩⟩]
ParseMonIDP_P[u[0]] = ⟨⟨1_𝓕, ⟨⟨1⟩, ⟨⟩, ⟨⟩⟩⟩⟩
ParseMonIDP_P[™Times[c, ™Power[u, k]]] = where[p = ParsePolF_P[c],
  ⟨⟨p, ⟨⟨⟩, k, ⟨⟩⟩⟩⟩]
```

```
Definition["ParsingIDO", any[𝓕],
 ParsingIDO[𝓕] = Functor[P, any[c, m, n, r, λ, m̄, n̄],
```

```
𝕤 = 〈〉
```

```
ParsePol_P[] = 〈〉

ParsePol_P[™Tuple[m̄]] = 〈ParsePol_P[〈m̄〉_i] | i=1,…,|〈m̄〉|〉

ParsePol_P[™Plus[]] = 〈〉

ParsePol_P[™Plus[m, n̄]] = ParsePol_P[m] ⧺ ParsePol_P[™Plus[n̄]]

ParsePol_P[™Times[™Minus[c], m̄]] = ParseTerm_P[™Times[™Minus[c], m̄]]

ParsePol_P[™Times[c, m̄]] = ParseTerm_P[™Times[c, m̄]]

ParsePol_P[™Minus[™Times[™Minus[c], m̄]]] = ParseTerm_P[™Times[c, m̄]]

ParsePol_P[™Minus[™Times[c, m̄]]] = ParseTerm_P[™Times[™Minus[c], m̄]]

ParsePol_P[™Minus[m, n]] = ParsePol_P[m] ⧺ ParsePol_P[™Minus[n]]

ParsePol_P[™Minus[m, n, r, m̄]] = ParsePol_P[m] ⧺ ParsePol_P[™Minus[™Minus[n], r, m̄]]

ParsePol_P[™Minus[c]] = { 〈〈-c, 〈〉〉〉              ⇐ ™IsNumber[c]
                        { ParseTerm_P[™Minus[c]]   ⇐ otherwise

ParsePol_P[c] = { 〈〈c, 〈〉〉〉       ⇐ ™IsNumber[c]
                { ParseTerm_P[c]   ⇐ otherwise

ParseTerm_P[™Minus[c]] = { 〈〈-c, 〈〉〉〉              ⇐ ™IsNumber[c]
                         { -1 ·_𝓕 ParseMon_P[c]      ⇐ otherwise

ParseTerm_P[™Times[™Times[m̄]]] = ParseTerm_P[™Times[m̄]]

ParseTerm_P[™Times[™Minus[c], m̄]] = { -c ·_𝓕 ParseTerm_P[™Times[m̄]]        ⇐ ™IsNumber[c]
                                    { -1 ·_𝓕 ParseTerm_P[™Times[c, m̄]]     ⇐ otherwise

ParseTerm_P[™Times[c, m̄]] = { c ·_𝓕 ParseTerm_P[™Times[m̄]]   ⇐ ™IsNumber[c]
                            { ParseMon_P[™Times[c, m̄]]       ⇐ otherwise

ParseTerm_P[c] = { 〈〈c, 〈〉〉〉      ⇐ ™IsNumber[c]
                 { ParseMon_P[c]   ⇐ otherwise

ParseTerm_P[™Times[]] = 〈〉

ParseMon_P[™Times[™Power[u, n], ™Power[v, λ]]] = 〈〈1, 〈〈"⌈⌉", 〈〈〈〈1_𝓕, 〈〈〉, n, 〈〉〉〉〉, 〈〈〉, 〈〉, 〈〉〉〉〉〉, 〈"⌈⌉",

ParseMon_P[™Times[™Times[™Power[u, n], ™Power[v, λ]]]] = 〈〈1, 〈〈"⌈⌉", 〈〈〈〈1_𝓕, 〈〈〉, n, 〈〉〉〉〉, 〈〈〉, 〈〉, 〈〉〉〉〉

ParseMon_P[™Power[u, n]] = 〈〈1, 〈〈"⌈⌉", 〈〈〈〈1_𝓕, 〈〈〉, n, 〈〉〉〉〉, 〈〈〉, 〈〉, 〈〉〉〉〉〉〉〉〉

ParseMon_P[™Power[u[0], λ]] = 〈〈1, 〈〈"⌈⌉", 〈〈〈〈1_𝓕, 〈λ, 〈〉, 〈〉〉〉〉, 〈〈〉, 〈〉, 〈〉〉〉〉〉〉〉〉

ParseMon_P[™Floor[c]] = 〈〈1, 〈〈"⌊⌋", c〉〉〉〉

ParseMon_P[™Times[™Floor[c]]] = 〈〈1, 〈〈"⌊⌋", c〉〉〉〉

ParseMon_P["E"] = 〈〈1, 〈〈"⌊⌋", 0〉〉〉〉

ParseMon_P[™Times["E"]] = 〈〈1, 〈〈"⌊⌋", 0〉〉〉〉

ParseMon_P["∂"] = 〈〈1, 〈"∂"〉〉〉

ParseMon_P[™Times["∂"]] = 〈〈1, 〈"∂"〉〉〉

ParseMon_P["∫"] = 〈〈1, 〈"∫"〉〉〉

ParseMon_P[™Times["∫"]] = 〈〈1, 〈"∫"〉〉〉

ParseMon_P[™Power[m, 0]] = 〈〉

ParseMon_P[™Power[m, 1]] = ParseMon_P[m]

ParseMon_P[™Power[m, n]] = ParseMon_P[m] ⋆_𝓕 ParseMon_P[™Power[m, n -_MmaAuxF[] 1]]

ParseMon_P[™Times[™Power[m, n]]] = ParseMon_P[™Power[m, n]]

ParseMon_P[™Times[™Times[m̄]]] = ParseMon_P[™Times[m̄]]

ParseMon_P[™Times[m, n, n̄]] = ParseMon_P[m] ⋆_FreeIntDiffOp[IntDiffPolys[IntDiffPolys[𝓕,𝕂],𝕂],𝕂] ParseMon_P[™Times[n, n̄]]

ParseMon_P[c] = 〈〈c, 〈〉〉〉
```

```
ParseMon_P[] = ⟨⟩
```

```
Definition["ParsingRules", any[ℱ],
 ParsingRules[ℱ] = Functor[P, any[c, m, n, r, λ, m̄, n̄],

   𝕤 = ⟨⟩
   ―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
   ParsePol_P[™Plus[™Times[™Minus[™Power[u, ™Tuple[1]]], ™Power[v, ™Tuple[1]]], ™CenterDot[™Power[u, ™Tupl
   ParsePol_P[™Plus[™Minus["E"], ™Times["E", "E"]]] = ⟨⟨1, ⟨⟨"⌊⌋", 0⟩, ⟨"⌊⌋", 0⟩⟩⟩, ⟨-1, ⟨⟨"⌊⌋", 0⟩⟩⟩⟩
   ParsePol_P[™Minus[™Times["E", ™Power[u, ™Tuple[1]]], ™Times[™Power[u[0], ™Tuple[1]], "E"]]] = ⟨⟨1, ⟨⟨"⌊⌋'
   ParsePol_P[™Minus[™Times["∂", ™Power[u, ™Tuple[1]]], ™Power[u, ™Tuple[0, 1]], ™Times[™Power[u, ™Tuple[1]
   ParsePol_P[™Times["∂", "E"]] = ⟨⟨1, ⟨"∂", ⟨"⌊⌋", 0⟩⟩⟩⟩
   ParsePol_P[™Plus[™Minus[1], ™Times["∂", "∫"]]] = ⟨⟨1, ⟨"∂", "∫"⟩⟩, ⟨-1, ⟨⟩⟩⟩
   ParsePol_P[™Minus[™Plus[™Times["∫", ™Power[u, ™Tuple[1]], "∫"], ™Times["∫", "∫", ™Power[u, ™Tuple[1]]]],
   ParsePol_P[™Plus[™Minus[™Plus[™Times["∫", ™Power[u, ™Tuple[0, 1]]], ™Times["∫", ™Power[u, ™Tuple[1]], "∂
   ParsePol_P[™Minus[™Times["∫", ™Power[u, ™Tuple[1]], "E"], ™Times["∫", ™Power[u, ™Tuple[1]], "E"]]] = ⟨⟨1,
   ParseRules_P[⟨m̄⟩] = ⟨ParsePol_P[⟨m̄⟩_i] | i=1,…,|⟨m̄⟩|⟩
   ParsePol_P[™Minus[™Minus[™Times[™CenterDot["∫", ™Power[u, ™Tuple[1]]], ™CenterDot["∫", ™Power[v, ™Tuple
   ParsePol_P[™Plus[™Minus[™Times["∫", ™Power[v, ™Tuple[1]], "∫"], ™Times[™CenterDot["∫", ™Power[v, ™Tuple[
   ParsePol_P[™Plus[™Minus[™Times["∫", ™Power[u, ™Tuple[1]], "∫"], ™Times[™CenterDot["∫", ™Power[u, ™Tuple[
   ParsePol_P[™Plus[™Minus[™Times["∫", ™Power[u, ™Tuple[1]], "∂"], ™Power[u, ™Tuple[1]]], ™Plus[™Times["∫",
   ParsePol_P[™Plus[™Minus[™Plus[™Times["∫", ™Power[u, ™Tuple[1]], "∂"], ™Times["∫", ™Power[u, ™Tuple[0, 1]
  ]]
```

ParseP

```
AddFreshNameKeywords[{"ParseP", "ParseF", "ParseDP", "ParseDPMon", "ParseFunc", "ParseIDPMon",
   "ParseIDP", "ParseDP", "Parse", "ComputeP", "FormatP", "GreensOp", "Mult", "Fact", "MultF",
   "GreensFct", "Act", "AddIDP", "MultIDP", "MinIDP", "DiffIDP", "IntIDP", "ExpBasis", "ParseIDO",
   "ParseRules", "ParseRule", "SpolRed", "Spol", "Spol1", "ReducePol", "RedPol", "GB"}];
```

```
Clear[ParseP, ParseF, ParseDP, ParseDPMon, Parse, ComputeP, FormatP,
  GreensOp, Mult, Fact, MultF, GreensFct, Act, ParseFunc, ParseMonIDP, ParseIDP];
```

```
Theory IDP
```

```
Definition["Integro-Differential Polynomials",
                         𝔼xp = ExpPolys[𝕂]
                          𝕂 = ReductionField[ℂ]
                          𝔽 = DiffPolys[𝔼xp, 𝕂]
                          𝕀 = IntDiffOp[𝔼xp, 𝕂]
                         𝔾𝕣 = GreenSystem[𝔼xp, 𝕂]
                        𝕀𝔻ℙ = IntDiffPolys[𝔼xp, 𝕂]
                          𝕒 = Parsing[𝕀]
 ℙ = GB[FreeIntDiffOp[IntDiffPolys[IntDiffPolys[𝔼xp, 𝕂], 𝕂], 𝕂]]
                          𝕡 = PrettyPrintingIDP[𝔽]
   𝕘𝕤 = GreenSystem[IntDiffPolys[IntDiffPolys[𝔼xp, 𝕂], 𝕂], 𝕂]
 𝔾 = GB[FreeIntDiffOp[IntDiffPolys[IntDiffPolys[𝔼xp, 𝕂], 𝕂], 𝕂]]
]
```

```
Theory["Integro-Differential Polynomials",
          Definition["Natural Numbers"]
                 Definition["ℕ"]
       Definition["Rational Number Field"]
                 Definition["ℚ"]
          Definition["Reduction Field"]
       Definition["Complex Number Field"]
                 Definition["ℂ"]
          Definition["Complex Numbers"]
                 Definition["ℂM"]
           Definition["Tuples Monoid"]
         Definition["Cartesian Product"]
          Definition["Cartesian Power"]
            Definition["Free Module"]
           Definition["Monoid Algebra"]
             Definition["Exp Polys"]
           Definition["Mma Functions"]
     Definition["Differential Polynomials"]
 Definition["Integro-Differential Polynomials"]
        Definition["Term Monoid for IDP"]
    Definition["Integro-Differential Polys"]
            Definition["Finite Chain"]
            Definition["Word Monoid"]
 Definition["Word Monoid with Degree Ordering"]
        Definition["Groebner Extension"]
    Definition["Theorema general functions"]
               Definition["Basis"]
            Definition["Double Basis"]
           Definition["PrettyPrinting"]
                 Definition["↦"]
            Definition["Green System"]
    Definition["Groebner Basis Normal Form"]
           Definition["FreeIntDiffOp"]
       Definition["Basis for Characters"]
         Definition["Quotient Algebra"]
             Definition["IntDiffOp"]
            Definition["Parsing Exp"]
             Definition["ParsingF"]
            Definition["ParsingIDO"]
        Definition["Parsing Diff Polys"]
            Definition["ParsingRules"]
 Definition["Integro-Differential Polynomials"]
]
```

```
Built-in["Integro-Differential Polynomials",
   Built-in["Numbers"]
 Built-in["Quantifiers"]
    Built-in["Tuples"]
 Built-in["Connectives"]
      Built-in["Sets"]
  Built-in["Operators"]
  Built-in["IsComplex"]
  Built-in["ComplexLt"]
      Built-in["Mult"]
     Built-in["Minus"]
   Built-in["IsNumber"]
 Built-in["MmaSimplify"]
  Built-in["FuncOrder"]
     Built-in["FuncQ"]
   Built-in["MmaDiff"]
    Built-in["MmaInt"]
     Built-in["ToMma"]
   Built-in["MmaPlus"]
  Built-in["MmaMinus"]
  Built-in["MmaTimes"]
   Built-in["MmaSubs"]
   Built-in["TmaTRFM"]
  Built-in["TmaSplit"]
]
```

```
Use[Theory["Integro-Differential Polynomials"]]
```

```
ParseP[p___] := Compute[ParsePol[p],
                            a
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseF[f___] := Compute[ ParsePolF          [f],
                          ParsingDP[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseDP[f___] := Compute[ ParseDP          [f],
                           ParsingDP[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseDPMon[f___] := Compute[ParseMonDP     [f],
                             ParsingDP[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseFunc[f___] := Compute[ParseFunc       [f],
                            ParsingDP[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseIDPMon[f___] := Compute[ParseMonIDP       [f],
                             ParsingDP[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseIDP[f___] := Compute[ ParseIDP      [f],
                          ParsingDP[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseIDO[f___] := Compute[ ParsePol        [f],
                          ParsingIDO[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseRules[f___] := Compute[ ParseRules        [f],
                            ParsingRules[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
ParseRule[f___] := Compute[ ParsePol          [f],
                           ParsingRules[Exp]
   built-in → ⟨Built-in["Tuples"], Built-in["Quantifiers"], Built-in["Connectives"],
     Built-in["Mult"], Built-in["Minus"], Built-in["IsNumber"], Built-in["MmaSimplify"]⟩];
```

```
AddIDP[p_, q_] := ComputeP[ParseIDP[p]           +            ParseIDP[q]] // PrintIDP;
                                       IntDiffPolys[Exp,K]
```

```
MinIDP[p_, q_] := ComputeP[ParseIDP[p]           -            ParseIDP[q]] // PrintIDP;
                                       IntDiffPolys[Exp,K]
```

```
MultIDP[p_, q_] := ComputeP[ParseIDP[p]           *            ParseIDP[q]] // PrintIDP;
                                        IntDiffPolys[Exp,K]
```

```
DiffIDP[p_] := ComputeP[∂IntDiffPolys[Exp,K] ParseIDP[p]] // PrintIDP;
```

```
IntIDP[p_] := ComputeP[∫                        ParseIDP[p]] // PrintIDP;
                        IntDiffPolys[Exp,K]
```

```
ExpBasis[p_] := ComputeP[exp-basis[ParseIDP[p]]] // PrintIDP;
                                   IDP
```

```
AddFreshNameKeywords[{"PrintDP", "PrintIDP", "PrintP", "Print", "PrintRules"}];
```

```
PrintDP[x_] := Compute[PrettyPrint       [x], built-in → Built-in["Integro-Differential Polynomials"]];
                       DiffPolys[Exp,K]
```

```
PrintIDP[p_] := Compute[ PrettyPrint       [p], built-in → Built-in["Integro-Differential Polynomials"]];
                        IntDiffPolys[Exp,K]
```

```
PrintP[p_] := Compute[PrettyPrint_PrettyPrinting[FreeIntDiffOp[IntDiffPolys[IntDiffPolys[Exp,K],Exp],K],K][p],
   built-in → Built-in["Integro-Differential Polynomials"]];
```

```
PrintRules[r_] := Compute[PrettyPrintRules_PrettyPrinting[FreeIntDiffOp[IntDiffPolys[IntDiffPolys[Exp,K],Exp],K],K][r],
   built-in → Built-in["Integro-Differential Polynomials"]];
```

```
ComputeP[p_] := Compute[p, built-in → Built-in["Integro-Differential Polynomials"]];
```

```
Parse[p_] := ParseP[p /. mytimes → ™Times];
```

```
Mult[p_, q_] := PrintIDP[ComputeP[Parse[p] ⋆_I Parse[q]]];
```

```
Act[g_, f_] := FormatP[ComputeP[Parse[g] ⊙_F ParseF[f]]] /.
   {™Times → Times, ™Plus → Plus, ™Minus → Minus, ™Power → Power};
```

```
SpolRed[f_, g_] := PrintP[spolred_G[ParseRules[f], g]];
```

```
SpolRed[f_] := PrintP[spolred_G[ParseRules[f], 𝒢_gs]];
```

```
Spol[f_, g_] := PrintP[spol_G[ParseRule[f], ParseRule[g]]];
```

```
ReducePol[f_] := PrintP[tred_G[ParseRule[f], 𝒢_gs]];
```

```
GB[s_] := PrintP[Gb_G[ParseRules[s], 𝒢_gs]];
```

Example Computations IDP

```
Use[Theory["Integro-Differential Polynomials"]]
```

```
MultIDP[u[0]^〈1〉 u^〈2〉 "∫" x u^〈1,1〉 "∫" (3 x^2) u^〈0,2〉, 3 u[0]^〈2,3〉 u^〈3,1〉 "∫" (4 x) u^〈1,0,1〉]
```

```
3 u[0]^〈3,3〉 u^〈5,1〉 ∫ x u^〈1,1〉 ∫ 4 x u^〈1,0,1〉 ∫ 3 x^2 u^〈0,2〉 +
 3 u[0]^〈3,3〉 u^〈5,1〉 ∫ x u^〈1,1〉 ∫ 3 x^2 u^〈0,2〉 ∫ 4 x u^〈1,0,1〉 + 3 u[0]^〈3,3〉 u^〈5,1〉 ∫ 4 x u^〈1,0,1〉 ∫ x u^〈1,1〉 ∫ 3 x^2 u^〈0,2〉
```

$$\texttt{DiffIDP}\Big[3\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}+x^3-x\right)\,u^{\langle 5,6\rangle}-\left(3\,E^x+x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,3\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,6\rangle}+\left(3\,E^x+x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,5\rangle}-\left(3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,2\rangle}\Big]$$

$$-12\,E^x\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 3,2\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,2\rangle}+\left(4\,x+12\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 3,3\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,5\rangle}+$$
$$\left(-4\,x-12\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 3,4\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}+12\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 3,6\rangle}\int -x+x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}-$$
$$3\,E^x\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,2\rangle}+\left(1+3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,5\rangle}+$$
$$\left(-1-3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,3\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}+\left(3\,x^3\,E^x+6\,x^3\,E^{5\,x}\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 9,3\rangle}+$$
$$\left(-x^4-3\,x^3\,E^x-2\,x^4\,E^{4\,x}-6\,x^3\,E^{5\,x}\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 9,7\rangle}+\left(x^4+3\,x^3\,E^x+2\,x^4\,E^{4\,x}+6\,x^3\,E^{5\,x}\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 9,9\rangle}+$$
$$\left(-3\,x+3\,x^3-6\,x^3\,E^{4\,x}\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 9,11\rangle}-3\,E^x\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,0,1\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,2\rangle}+$$
$$\left(2\,x+6\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1,1\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,5\rangle}+$$
$$\left(-3\,x-9\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2,1\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}+$$
$$15\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,4,1\rangle}\int -x+x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}$$

$$\texttt{IntIDP}\Big[3\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5,6,1\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}+x^3-x\right)\,u^{\langle 5,6\rangle}-\left(3\,E^x+x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,3\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,6\rangle}+\left(3\,E^x+x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,5\rangle}-\left(3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,2\rangle}\Big]$$

$$-\frac{1}{7}\,u[0]^{\langle 1,2,3\rangle}\int 12\,u^{\langle 3,6,7\rangle}\int -x+x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}-\frac{1}{7}\,u[0]^{\langle 1,2,3\rangle}\int 15\,u^{\langle 4,4,8\rangle}\int -x+x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}-$$
$$\frac{1}{5}\,u[0]^{\langle 1,2,3\rangle}\int -3\,E^x\,u^{\langle 5\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,2\rangle}+u[0]^{\langle 1,2,3\rangle}\int -x-3\,E^x\,u^{\langle 4,3\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}+$$
$$u[0]^{\langle 1,2,3\rangle}\int x+3\,E^x\,u^{\langle 4,2\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,5\rangle}-\frac{1}{7}\,u[0]^{\langle 1,2,3\rangle}\int -3\,x+3\,x^3-6\,x^3\,E^{4\,x}\,u^{\langle 9,11,7\rangle}-$$
$$\frac{1}{5}\,u[0]^{\langle 1,2,3\rangle}\int 3\,x^3\,E^x+6\,x^3\,E^{5\,x}\,u^{\langle 10,2\rangle}-\frac{3}{5}\,E^x\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 5\rangle}\int -x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,2\rangle}+$$
$$\frac{3}{7}\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5,7\rangle}\int -x+x^3-2\,x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}$$

$$\texttt{ExpBasis}\Big[3\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}+x^3-x\right)\,u^{\langle 5,6\rangle}-\left(3\,E^x+x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,3\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,6\rangle}+\left(3\,E^x+x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,5\rangle}-\left(3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1\rangle}\,\text{"}\!\int\!\text{"}\,\left(-2\,x^3\,E^{4\,x}-x^3\right)\,u^{\langle 5,2\rangle}\Big]$$

$$3\,E^x\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1\rangle}\int x^3\,u^{\langle 5,2\rangle}+6\,E^x\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,1\rangle}\int x^3\,E^{4\,x}\,u^{\langle 5,2\rangle}+$$
$$\left(-x-3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2\rangle}\int x^3\,u^{\langle 5,5\rangle}+\left(-2\,x-6\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,2\rangle}\int x^3\,E^{4\,x}\,u^{\langle 5,5\rangle}+$$
$$\left(x+3\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,3\rangle}\int x^3\,u^{\langle 5,6\rangle}+\left(2\,x+6\,E^x\right)\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,3\rangle}\int x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}-3\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5\rangle}\int x\,u^{\langle 5,6\rangle}+$$
$$3\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5\rangle}\int x^3\,u^{\langle 5,6\rangle}-6\,u[0]^{\langle 1,2,3\rangle}\,u^{\langle 4,5\rangle}\int x^3\,E^{4\,x}\,u^{\langle 5,6\rangle}$$