Contractor

Contractor[*form*, *vector*]
represents the contraction of a vector on a form.

Details

- On output Contractors are formatted as PineGreen AngleBrackets.
- A Contractor template is on the × tab of the Common Operations palette.
- A Contractor requires both 1-form and 1-vector elements but you can only be in one of the bases at a time. Nevertheless, the Grassmann Symbols palette will allow you to paste both types of elements and you can easily type the form symbols.
- Note that the first slot always contains an m-form and the second slot a k-vector. The result of evaluation will be zero unless `m ≥ k`. If the k-vector is within the form the result of evaluation will be an (m-k)-form.
- Essentially, a form "measures" the vector. If the vector doesn't ""fit" into the form the result is zero.
- A `LeftContractor` is equivalent to a `Contractor`. A `RightContractor` will sometimes give a different sign. `RightContractor` == $(-1)^{\text{Grade[vector] (Grade[form]-Grade[vector])}}$ `LeftContractor`.
- The form and vector symbols are available on a current `GrassmannSymbolsPalette`.
- A Contractor should be distinguished from an uncolored AngleBracket, also used in Grassmann Algebra to represent the coefficient of a general n-element in the unit n-element.

Examples (5)

Basic Examples (1)

See below for a general overview of the contraction routines

```
In[1]:= << GrassmannCalculus`

In[2]:= SetActiveAssociation[PublicGrassmannAtlas["Euclidean 3-Space"]]
```

A Contractor can be entered as a plain expression or from the palette. `ContractorRules` evaluates scalar contractors.

```
In[3]:= {Contractor[dx, e_x], ⟨dx, e_x⟩}
        % /. ContractorRules[]
Out[3]= {⟨dx, e_x⟩, ⟨dx, e_x⟩}

Out[3]= {1, 1}
```

General exterior elements can be entered.

```
In[4]:= ⟨1 + dx + dx ∧ dy + dx ∧ dy ∧ dz, a e_x + b e_x ∧ e_y ∧ e_z⟩
        % // EvaluateContractors
Out[4]= ⟨1 + dx + dx ∧ dy + dx ∧ dy ∧ dz, a e_x + b e_x ∧ e_y ∧ e_z⟩

Out[4]= a + b + a dy + a dy ∧ dz
```

A `LeftContractor` is equivalent to a `Contractor`. A `RightContractor` may give a change of sign.

```
In[5]:= {⟨dx ∧ dy, e_x⟩, e_x ↦ dx ∧ dy, dx ∧ dy ↤ e_x}
        % // EvaluateContractors
Out[5]= {⟨dx ∧ dy, e_x⟩, e_x ↦ dx ∧ dy, dx ∧ dy ↤ e_x}

Out[5]= {dy, dy, -dy}
```

General Overview of Contraction Routines (4)

Set 6-dimensional Euclidean space.

```
In[33]:= SetEuclideanNSpace[6, {u, v, w, x, y, z}, "Vector"]
         **S[a, b, c, d]
```

The following are contractions of a 1-vector on a 1-form, which results in a scalar.

```
In[35]:= {⟨du, e_u⟩, ⟨du, e_v⟩}
         % // EvaluateContractors
Out[35]= {⟨du, e_u⟩, ⟨du, e_v⟩}

Out[36]= {1, 0}
```

The following can be evaluated directly or expanded first and evaluated with rules.

```
In[37]:= ⟨a dx + b dy, c e_x + d e_y⟩
         % // EvaluateContractors
Out[37]= ⟨a dx + b dy, c e_x + d e_y⟩

Out[38]= a c + b d

In[51]:= ⟨a dx + b dy, c e_x + d e_y⟩
         % // GrassmannBreakout[Contractor, All]
         % // EvaluateContractors
Out[51]= ⟨a dx + b dy, c e_x + d e_y⟩

Out[52]= a c ⟨dx, e_x⟩ + a d ⟨dx, e_y⟩ + b c ⟨dy, e_x⟩ + b d ⟨dy, e_y⟩

Out[53]= a c + b d
```

Grassmann interior products can usually be evaluated faster by evaluating with `ContractInteriorProducts`, which converts them to `Contractors` using the metric.

```
In[72]:= (e_z ∧ (a e_x + b e_y)) ⊖ (c e_x + d e_y)
         % // ToMetricElements // Simplify // Timing
         %% // ContractInteriorProducts // Timing
Out[72]= e_z ∧ (a e_x + b e_y) ⊖ (c e_x + d e_y)

Out[73]= {0.312002, -(a c + b d) e_z}

Out[74]= {0.0468003, -(a c + b d) e_z}
```

Contractors don't require the use of a metric. (A metric is required to convert between forms and vectors and between interior products and contractors.) Without a metric, contractors can be evaluated by reducing them to scalar contractors and using `ContractorRules`. These are equivalent to Kronecker delta on the dual elements.

```
In[24]:= ContractorRules[]
Out[24]= {⟨du, e_u⟩ → 1, ⟨du, e_v⟩ → 0, ⟨du, e_w⟩ → 0, ⟨du, e_x⟩ → 0, ⟨du, e_y⟩ → 0, ⟨du, e_z⟩ → 0,
          ⟨dv, e_u⟩ → 0, ⟨dv, e_v⟩ → 1, ⟨dv, e_w⟩ → 0, ⟨dv, e_x⟩ → 0, ⟨dv, e_y⟩ → 0, ⟨dv, e_z⟩ → 0,
          ⟨dw, e_u⟩ → 0, ⟨dw, e_v⟩ → 0, ⟨dw, e_w⟩ → 1, ⟨dw, e_x⟩ → 0, ⟨dw, e_y⟩ → 0, ⟨dw, e_z⟩ → 0,
          ⟨dx, e_u⟩ → 0, ⟨dx, e_v⟩ → 0, ⟨dx, e_w⟩ → 0, ⟨dx, e_x⟩ → 1, ⟨dx, e_y⟩ → 0, ⟨dx, e_z⟩ → 0,
          ⟨dy, e_u⟩ → 0, ⟨dy, e_v⟩ → 0, ⟨dy, e_w⟩ → 0, ⟨dy, e_x⟩ → 0, ⟨dy, e_y⟩ → 1, ⟨dy, e_z⟩ → 0,
          ⟨dz, e_u⟩ → 0, ⟨dz, e_v⟩ → 0, ⟨dz, e_w⟩ → 0, ⟨dz, e_x⟩ → 0, ⟨dz, e_y⟩ → 0, ⟨dz, e_z⟩ → 1}

In[75]:= ⟨a dx + b dy, c e_x + d e_y⟩
         % // GrassmannBreakout[Contractor, All]
         % /. ContractorRules[]
Out[75]= ⟨a dx + b dy, c e_x + d e_y⟩

Out[76]= a c ⟨dx, e_x⟩ + a d ⟨dx, e_y⟩ + b c ⟨dy, e_x⟩ + b d ⟨dy, e_y⟩

Out[77]= a c + b d
```

Higher order inner product contraction.

```
In[1]:= ⟨a dx ∧ dy, b e_x ∧ e_y⟩
        % // EvaluateContractors
Out[1]= ⟨a dx ∧ dy, b e_x ∧ e_y⟩

Out[1]= a b
```

In the following the vector is not contained in the form because e_u is not in the space of the form. The result is zero.

```
In[2]:= b e_u ∧ e_v ↦ a dx ∧ dv
        % // EvaluateContractors
Out[2]= b e_u ∧ e_v ↦ a dx ∧ dv

Out[2]= 0
```

Contracting different grade elements

```
In[1]:= e_u ∧ e_w ↦ du ∧ dv ∧ dw
        % // EvaluateContractors
Out[1]= e_u ∧ e_w ↦ du ∧ dv ∧ dw

Out[1]= -dv
```

Using more general Grassmann elements.

```
In[2]:= ⟨a du ∧ dv ∧ dw + b du ∧ dx, c e_u + d e_x⟩
        % // EvaluateContractors
Out[2]= ⟨b du ∧ dx + a du ∧ dv ∧ dw, c e_u + d e_x⟩

Out[2]= -b d du + b c dx + a c dv ∧ dw

In[3]:= ⟨1 + dx + dx ∧ dy + dx ∧ dy ∧ dz, 5 + a e_x + b e_x ∧ e_y ∧ e_z⟩
        % // EvaluateContractors
Out[3]= ⟨1 + dx + dx ∧ dy + dx ∧ dy ∧ dz, 5 + a e_x + b e_x ∧ e_y ∧ e_z⟩

Out[3]= 5 + a + b + 5 dx + a dy + 5 dx ∧ dy + a dy ∧ dz + 5 dx ∧ dy ∧ dz
```

A larger vector space contracts to zero on a smaller form space.

```
In[4]:= e_u ∧ e_v ∧ e_w ↦ du
        % // EvaluateContractors
Out[4]= e_u ∧ e_v ∧ e_w ↦ du

Out[4]= 0
```

Spacetime contractions

```
In[1]:= SetActiveAssociation[PublicGrassmannAtlas[["Minkowski Space"]]]
        ★★S[m, vx, vy, vz]; ★V[{}];
```

Writing the momentum of a particle p in terms of it's spatial velocity v.

```
In[2]:= v = {vx, vy, vz}.Drop[VectorBasis, 1]
        p = m / √(1 - v⊖v) (e_t + v)
Out[2]= vx e_x + vy e_y + vz e_z

Out[2]= m (e_t + vx e_x + vy e_y + vz e_z) / √(1 - (vx e_x + vy e_y + vz e_z) ⊖ (vx e_x + vy e_y + vz e_z))
```

The square of the Measure, or spacetime interval is:

```
In[3]:= p⊖p // ToMetricElements // Timing
Out[3]= {1.13881, -m^2}
```

This can also be calculated somewhat faster in the `Contractor` formalism as:

```
In[4]:= (p = p // ContractInteriorProducts) // Timing
        p ↦ (p /. VectorToForm)
        % // EvaluateContractors // Simplify // Timing
Out[4]= {0.0156001, m (e_t + vx e_x + vy e_y + vz e_z) / √(1 - vx^2 - vy^2 - vz^2)}

Out[4]= m (e_t + vx e_x + vy e_y + vz e_z) / √(1 - vx^2 - vy^2 - vz^2) ↦ m (-dt + dx vx + dy vy + dz vz) / √(1 - vx^2 - vy^2 - vz^2)

Out[4]= {0.156001, -m^2}
```

Notice that VectorToForm uses the metric, as did ContractInteriorProducts.

```
In[5]:= VectorToForm
Out[5]= {e_t → -dt, e_x → dx, e_y → dy, e_z → dz}
```

See Also

LeftContractor ▪ **RightContractor** ▪ **ConvertToContractor** ▪ **ContractorRules** ▪ **ToScalarContractor** ▪ **EvaluateContractors** ▪ **ContractInteriorProducts** ▪ **FormBasis** ▪ **VectorBasis**

Related Guides

- Contractors
- Derivatives
- Grassmann Calculus