FieldsX — An extension package for the xAct tensor computer algebra suite to include fermions, gauge fields and BRST cohomology

Markus B. Fröb

Institut für Theoretische Physik, Universität Leipzig, Brüderstraße 16, 04103 Leipzig, Germany

E-mail: mfroeb@itp.uni-leipzig.de

ABSTRACT: I present the tensor computer algebra package FieldsX, which extends the xAct suite of tensor algebra packages to perform computations in field theory with fermions and gauge fields. This includes the standard tools of curved-space γ matrices, Fierz identities, invariant tensors on Lie algebras, arbitrary gradings and left and right variational derivatives, as well as the decomposition of spinor products into irreducible components following the approach of d'Auria, Fré, Maina and Regge [*Annals Phys.* **139** (1982) 93]. Lastly, it also includes functions to work with nilpotent differentials such as the BV–BRST differential for (supersymmetric) gauge theories and to compute their (relative) cohomologies, from which anomalies and gauge-invariant operators can be determined. I illustrate the use of the package with the example of $\mathcal{N} = 1$ Super-Yang–Mills theory.

PROGRAM SUMMARY:

Program title:	FieldsX
Version:	1.1.4
Programming language:	MATHEMATICA (version 8.0 or later)
Obtainable from:	Included with the arXiv submission, or GitHub
License:	GNU GPL 2.0 or later

Contents

1 Introduction

The free xAct suite of packages [1–8] for tensor algebra is a powerful tool to perform computations in general relativity and field theory which has found widespread use; the webpage on papers that use xAct (which is certainly incomplete) lists about 630 papers to date. However, while functionality to work with the Newman-Penrose two-component spinor calculus in 3+1 dimensions is available [5], working with γ matrices, non-commutative spinors or Fierz identities requires additional and quite non-trivial effort from part of the user. Such functionality is available in the competitor computer algebra system Cadabra [9, 10], which however lacks other functionality present in Mathematica that is invaluable to deal with expressions containing tens of thousands of terms. FieldsX tries to fill this gap and make the xAct suite also useful for computations with fermions and gauge theories. For an efficient generation of suitable ansätze for the computation of BRST cohomology, FieldsX relies on the Multisets package which is © 2011 David Bevan and distributed under the Wolfram Library Archive License. FieldsX is fully integrated with other xAct packages, for example the xPert package to compute perturbations (including perturbations of curved-space γ matrices and spin connections) or the TexAct package for TeX output.

This paper is organised as follows: section 2 describes how to install and load the package, section 3 illustrates the main parts of FieldsX by verifying the supersymmetry and conformal symmetry of (classical) $\mathcal{N} = 1$ Super-Yang–Mills theory and by computing its possible anomalies, and section 4 contains a complete list of all functions and variables (with short descriptions) contained in FieldsX. It is assumed that the reader is already familiar with the basic functionality of xAct; commands that the user has to execute in

MATHEMATICA are given in typewriter face (`Command[]`), while output from MATHEMATICA is given in smaller size.

2 Installation

FIELDSX requires a working installation of XACT, at least version 1.1.4 released on 16 February 2020. FIELDSX is then installed by downloading the package file `FieldsX.m`, which is included as ancillary file with the arXiv submission of this paper and published on GitHub at `github.com/mfroeb/FieldsX`, and placing it into the XACT installation directory, which for a per-user installation of XACT is obtained from `FileNameJoin[$UserBaseDirectory,` `"Applications/xAct"]`, or for a system-wide installation from `FileNameJoin[$BaseDirec-` `tory, "Applications/xAct"]`. To install the required MULTISETS package, the package file `Multisets.m` must be downloaded from `library.wolfram.com/infocenter/MathSource/` `8115/` and placed into the application directory `FileNameJoin[$UserBaseDirectory, "Appli-` `cations"]` (for a per-user install) or `FileNameJoin[$BaseDirectory, "Applications"]` (for a system-wide install).[1]

The package can then be loaded in the standard way with `Get["xAct‘FieldsX‘"]` or `Needs["xAct‘FieldsX‘"]`, and will itself load the required packages from the XACT suite. Short information for any function is displayed in the same manner as for all Mathematica functions by typing `?FunctionName`.

2.1 Compatibility with other packages

The XACT package SPINORS to work with two-component spinors in four dimensions also defines the commands `DefSpinStructure`, `UndefSpinStructure`, `DefSpinor` and `UndefSpinor`.[2] If both packages are to be loaded at the same time, one could use the fully qualified names `xAct‘Spinors‘DefSpinStructure`, `xAct‘FieldsX‘DefSpinStructure` and so on to refer to the respective commands. Alternatively, if FIELDSX is loaded *after* SPINORS, the commands `DefSpinStructure`, `UndefSpinStructure`, `DefSpinor` and `UndefSpinor` from the SPINORS package are redefined by FIELDSX to `Def2CompSpinStructure`, `Undef2CompSpin-` `Structure`, `Def2CompSpinor` and `Undef2CompSpinor` if the variable `$Keep2CompDefs` is not set or `False`, while the commands `DefSpinStructure`, `UndefSpinStructure`, `DefSpinor` and `UndefSpinor` from the FIELDSX package are redefined to `DefGenSpinStructure`, `UndefGen-` `SpinStructure`, `DefGenSpinor` and `UndefGenSpinor` if the variable `$Keep2CompDefs` is `True`. Note that `$Keep2CompDefs` must be set *before* loading FIELDSX.

3 $\mathcal{N} = 1$ Super-Yang–Mills theory

To show how this package can be used, let us consider $\mathcal{N} = 1$ Super-Yang–Mills theory [11–13], which in addition to the Yang–Mills vector boson contains a Majorana spinor χ^a and

---

[1]If the cohomology functionality of FIELDSX is not needed, installation of the MULTISETS package is not necessary since it is loaded dynamically with `DeclarePackage`.

[2]I thank Thomas Bäckdahl for pointing this out to me.

an auxiliary field D^a in the adjoint representation. The action reads

$$S = -\frac{1}{4} \int F^a_{\mu\nu} F^{\mu\nu a} \, \mathrm{d}x - \frac{1}{2} \int \bar{\chi}^a \gamma^\mu (D_\mu \chi)^a \, \mathrm{d}x + \frac{1}{2} \int D^a D_a \, \mathrm{d}x \,, \tag{3.1}$$

where $\mathrm{d}x \equiv \sqrt{-g} \, \mathrm{d}^4 x$, with the field strength

$$F^a_{\mu\nu} \equiv \nabla_\mu A^a_\nu - \nabla_\nu A^a_\mu + \mathrm{i}g f_{abc} A^b_\mu A^c_\nu \tag{3.2}$$

and the gauge-covariant derivative

$$(D_\mu \chi)^a \equiv \nabla_\mu \chi^a + \mathrm{i}g f_{abc} A^b_\mu \chi^c \,, \tag{3.3}$$

and is invariant under the gauge transformation

$$\delta^{\mathrm{gauge}}_\xi A^a_\mu = (D_\mu \xi)^a \,, \qquad\qquad \delta^{\mathrm{gauge}}_\xi D^a = -\mathrm{i}g f_{abc} \xi^b D^c \,, \tag{3.4a}$$

$$\delta^{\mathrm{gauge}}_\xi \chi^a = -\mathrm{i}g f_{abc} \xi^b \chi^c \,, \qquad\qquad \delta^{\mathrm{gauge}}_\xi \bar{\chi}^a = -\mathrm{i}g f_{abc} \xi^b \bar{\chi}^c \,, \tag{3.4b}$$

and the supersymmetry transformation

$$\delta^{\mathrm{susy}}_\epsilon A^a_\mu = -\bar{\epsilon} \gamma_\mu \chi^a \,, \qquad\qquad \delta^{\mathrm{susy}}_\epsilon D^a = \mathrm{i}\bar{\epsilon}\gamma_* \gamma^\mu (D_\mu \chi)^a \,, \tag{3.5a}$$

$$\delta^{\mathrm{susy}}_\epsilon \chi^a = \frac{1}{2} \gamma^{\mu\nu} F^a_{\mu\nu} \epsilon + \mathrm{i} D^a \gamma_* \epsilon \,, \qquad \delta^{\mathrm{susy}}_\epsilon \bar{\chi}^a = -\frac{1}{2} \bar{\epsilon} \gamma^{\mu\nu} F^a_{\mu\nu} + \mathrm{i} D^a \bar{\epsilon}\gamma_* \,, \tag{3.5b}$$

where ϵ is a constant Grassmann-odd Majorana spinor. To check invariance under the supersymmetry transformation, one needs to use Fierz rearrangement identities [14, 15][3] and the Bianchi identities for the field strength tensor. Two supersymmetry transformations close into a gauge transformation and a translation,

$$[\delta^{\mathrm{susy}}_2, \delta^{\mathrm{susy}}_1] A^a_\mu = -b^\nu F^a_{\mu\nu} = b^\nu \nabla_\nu A^a_\mu + \delta^{\mathrm{gauge}}_\xi A^a_\mu \,, \tag{3.6a}$$

$$[\delta^{\mathrm{susy}}_2, \delta^{\mathrm{susy}}_1] \chi^a = b^\nu \nabla_\nu \chi^a + \delta^{\mathrm{gauge}}_\xi \chi^a \,, \tag{3.6b}$$

$$[\delta^{\mathrm{susy}}_2, \delta^{\mathrm{susy}}_1] D^a = b^\nu \nabla_\nu D^a + \delta^{\mathrm{gauge}}_\xi D^a \,, \tag{3.6c}$$

(Fierz rearrangement identities are needed again), with the translation parameter b^ν and the field-dependent gauge transformation parameter ξ^a defined by

$$b^\nu \equiv 2\bar{\epsilon}_1 \gamma^\nu \epsilon_2 \,, \qquad \xi^a \equiv -b^\nu A^a_\nu \,. \tag{3.7}$$

3.1 Gauge and supersymmetry invariance

To check (or obtain) these results using FIELDSX, we first need to load the package (which automatically loads all required xACT packages). For demonstration purposes, we also set the TEX output symbol for the non-commutative product to a dot; all output shown later on was copied using `CopyToClipboard@TexPrint[expr]` and not modified further.

```
<< xAct`FieldsX`
$CenterDotTexSymbol = "·";
```

---

[3] Attributed to Pauli by Fierz himself: "The content of this first section originates from Prof. W. Pauli and I am indebted to him for ceding me his calculations." [14], footnote 2.

```
------------------------------------------------------------
Package xAct`xPerm`  version 1.2.3, {2015,8,23}
CopyRight (C) 2003-2018, Jose M. Martin-Garcia, under the General Public License.

... (many lines of output not displayed)
```

We then define the manifold M, the metric gg and covariant derivative ∇, the spin structure (γ matrices) and the inner bundle (abstract Lie algebra in the adjoint representation of dimension *liedim*, with metric *cartankilling*). The option `SymCovDQ` from the xTras package makes it possible to use symmetrised covariant derivatives $\nabla_{\mu\cdots\nu} \equiv \nabla_{(\mu}\cdots\nabla_{\nu)}$, which is extremely useful to obtain canonical forms of expressions, required for the computation of cohomologies.

```
DefManifold[M, 4, {α, β, γ, δ, μ, ν, ρ, σ}];
DefMetric[-1, gg[-μ, -ν], CD, {";", "∇"}, PrintAs → "g", SymCovDQ → True];
DefSpinStructure[gg, {A, B, F, G, H, J, K, L, P, Q}];
DefConstantSymbol[liedim];
DefVBundleWithMetric[lie, M, liedim, {a, b, c, d, e, f, i, j, k, l, m, n, p, q},
    cartankilling];
```

Next we define the fields of the theory and the coupling g. Note that the spinors χ and ϵ have a spin-bundle index $-A$, which must always be the last index.

```
DefEvenTensor[AA[-μ, a], M, PrintAs → "A"];
DefOddSpinor[chi[a, -A], M, SpinorType → Majorana, PrintAs → "χ"];
DefEvenTensor[auxD[a], M, PrintAs → "D"];
DefEvenTensor[xi[a], M, PrintAs → "ξ"];
DefOddSpinor[eps[-A], M, SpinorType → Majorana, PrintAs → "ε"];
DefConstantSymbol[g];
```

The vector boson A_μ^a, the auxiliary field D^a and the gauge parameter ξ^a are Grassmann even, while the spinor χ_A^a and supersymmetry parameter ϵ_A are Grassmann odd. Together with χ and ϵ, the charge conjugate spinors $\bar{\chi}$ and $\bar{\epsilon}$ are automatically defined, and read `barchi[a, A]` and `bareps[A]`. We also have to ensure that the supersymmetry parameter is covariantly constant; a space-time dependent supersymmetry parameter leads to super-gravity theories.

```
CD[__]@eps[_] ^:= 0;
CD[__]@bareps[_] ^:= 0;
```

The field strength and gauge-covariant derivative are given in terms of the vector boson, but we first need to obtain the structure constants of the Lie algebra as the antisymmetric invariant tensor of rank 3 (and define a pretty output):

```
lief = InvariantTraceTensor[lie, 3, Antisymmetric];
Tex[lief] = "f";
```

** DefTensor: Defining tensor Invlief[a,b,c].

We then define the field strength and functions to convert it into derivatives of the vector boson and back. For the transformation back, we replace derivatives $\nabla_\mu A_\nu^a$ by the sum

of its symmetric part and the antisymmetric part, and rewrite the antisymmetric part in terms of $F^a_{\mu\nu}$:

```
DefEvenTensor[FF[-μ, -ν, a], M, Antisymmetric[{-μ, -ν}], PrintAs → "F"];
FToGradA[expr_] := expr /. {FF[μ_, ν_, a_] :> CD[μ]@AA[ν, a] - CD[ν]@AA[μ, a]
    + I g With[{b = DummyIn[lie], c = DummyIn[lie]},
    lief[a, -b, -c] AA[μ, b] AA[ν, c]]}
GradAToF[expr_] := expr /. {CD[μ_]@AA[ν_, a_] :> 1/2 (CD[μ]@AA[ν, a]
    + CD[ν]@AA[μ, a]) + 1/2 (FF[μ, ν, a] - I g With[{b = DummyIn[lie],
    c = DummyIn[lie]}, lief[a, -b, -c] AA[μ, b] AA[ν, c]])}
```

** DefTensor: Defining tensor FF[-μ,-ν,a].

While xAct provides a function to define the gauge-covariant derivative, we need to specify that it acts on the lie bundle (and the spin bundle). We then need to give the correct values to the relevant generalised Christoffel symbols, where we have to check the bundle to which the indices belong using the condition operator /;, and we check the results by converting the gauge-covariant derivative into the covariant derivative ∇. Since the gauge-covariant derivative does not act on space-time or spin indices, the corresponding Christoffel symbols are the same as for ∇.

```
DefCovD[DD[-μ], {lie, SpinM}, {":", "D"}];
```

** DefCovD: Defining covariant derivative DD[-μ].

** DefTensor: Defining vanishing torsion tensor TorsionDD[α,-β,-γ].

** DefTensor: Defining symmetric Christoffel tensor ChristoffelDD[α,-β,-γ].

...

** DefTensor: Defining nonsymmetric AChristoffel tensor AChristoffelDD[a,-β,-c].

```
AChristoffelDD[a_, μ_, b_] /; (VBundleOfIndex[a] == lie) && (VBundleOfIndex[b]
    == lie) := - I g With[{c = DummyIn[lie]}, lief[a, b, c] AA[μ, -c]]
AChristoffelDD[a_, μ_, b_] /; (VBundleOfIndex[a] == SpinM) &&
    (VBundleOfIndex[b] == SpinM) := AChristoffelCD[a, μ, b]
ChristoffelDD[α_, μ_, ν_] := ChristoffelCD[α, μ, ν]
ChangeCovD[ChangeCovD[DD[-μ]@AA[-ν, a], DD, PD], PD, CD]
ChangeCovD[ChangeCovD[DD[-μ]@chi[a, -A], DD, PD], PD, CD]
```

$$- ig A_{\mu c} A_\nu{}^b f^a{}_b{}^c + \nabla_\mu A_\nu{}^a$$
$$- ig A_{\mu c} \chi^b{}_A f^a{}_b{}^c + \nabla_\mu \chi^a{}_A$$

We can now write down the action (or rather the Lagrangian), where the non-commutative product of spinors is written · and can be entered with Esc + . + Esc. The γ matrices can be entered with Gammagg, which automatically selects the right one (Gammagg1, ...) depending on the number of indices. We also define a pretty output for them:

```
Tex[Gammagg1] ^= "\\gamma"; Tex[Gammagg2] ^= "\\gamma";
Tex[Gammagg3] ^= "\\gamma"; Tex[Gammagg4] ^= "\\gamma";
action = -1/4 FF[μ, ν, a] FF[-μ, -ν, -a] + 1/2 auxD[a] auxD[-a]
    - 1/2 barchi[a, A] · Gammagg[μ, -A, B] · DD[-μ]@chi[-a, -B]
```

$$\frac{1}{2} D_a D^a - \frac{1}{4} F_{\mu\nu a} F^{\mu\nu a} - \frac{1}{2} \bar\chi^{aA} \cdot D_\mu \chi_{aB} \, \gamma^\mu{}_A{}^B$$

We now expand the action, writing everything in terms of the basic fields and covariant derivative:

```
action2 = CollectTensors@ChangeCovD[ChangeCovD[FToGradA[action], DD, PD], PD, CD]
```

$\frac{1}{2}D_aD^a - \frac{1}{2}\bar{\chi}^{aA} \cdot \nabla_\mu \chi_{aB}\gamma^\mu{}_A{}^B + \frac{1}{2}igA^{\mu a}\bar{\chi}_b{}^A \cdot \chi_{cB}\gamma_{\mu A}{}^B f_a{}^{bc} + \frac{1}{4}g^2 A_\mu{}^b A^{\mu a} A_\nu{}^d A^{\nu c} f_{ac}{}^e f_{bde}$

$+ igA^{\mu a}A^{\nu b}f_{abc}\nabla_\nu A_\mu{}^c + \frac{1}{2}\nabla_\mu A_{\nu a}\nabla^\nu A^{\mu a} - \frac{1}{2}\nabla_\nu A_{\mu a}\nabla^\nu A^{\mu a}$

Expanding the action, we obtain a term with the product of two structure constants $f_{ade}f^a{}_{bc}$. Since the structure constants satisfy the multiterm Jacobi identity $f_{ab[c}f^a{}_{de]} = 0$, it is more difficult to bring such terms into a canonical form. One way to deal with multiterm identities are Young projectors (as used in the xTras package to deal with the Bianchi identities of the Riemann tensor), where one has to project $f_{abc}f^a{}_{de}$ on the tableau $\begin{smallmatrix}b&d\\c&e\end{smallmatrix}$ — antisymmetric in each pair bc and de, and symmetric under the interchange of the two pairs. This results in

```
YoungProject[lief[-a, -b, -c] lief[a, -d, -e], {{-b, -d}, {-c, -e}}]
```

$-\frac{1}{3}f_{be}{}^a f_{cda} + \frac{1}{3}f_{bd}{}^a f_{cea} + \frac{2}{3}f_{bc}{}^a f_{dea}\,,$

and one checks that the Jacobi identity is automatically fulfilled. The disadvantage of this approach is that one has to find the required Young tableau by hand, and FieldsX thus takes a different approach, inspired by the approach to Fierz identities of d'Auria, Fré, Maina and Regge [16] based on the decomposition of the product of group representations into irreducible components. For the invariant tensors of the Lie algebra, the corresponding group is the permutation group, but it is not necessary to perform a full decomposition into irreducibles. Instead, it is enough to use the formula for the commutator of two generators $\mathfrak{t}_a$ of the Lie algebra, multiply by other generators and take the trace to obtain

$$\mathrm{i}f_{ab}{}^c \, \mathrm{tr}[\mathfrak{t}_c\mathfrak{t}_d \cdots \mathfrak{t}_e] = \mathrm{tr}[[\mathfrak{t}_a, \mathfrak{t}_b]\mathfrak{t}_d \cdots \mathfrak{t}_e] = \mathrm{tr}[\mathfrak{t}_a\mathfrak{t}_b\mathfrak{t}_d \cdots \mathfrak{t}_e] - \mathrm{tr}[\mathfrak{t}_b\mathfrak{t}_a\mathfrak{t}_d \cdots \mathfrak{t}_e]\,,$$

expressing the product of a structure constant with an invariant tensor of the Lie algebra by the sum of two invariant tensors of higher rank. This gives

```
ReduceInvariantTraceTensors[lief[-a, -b, -c] lief[a, -d, -e]]
```

** DefTensor: Defining tensor Invlie4[a,b,c,d].

$i(tr[lie]_{bcde} - tr[lie]_{bced} - tr[lie]_{bdec} + tr[lie]_{bedc})$

and the action

```
action3 = CollectTensors@ReduceInvariantTraceTensors@Expand[action2]
```

$\frac{1}{2}D_aD^a - \frac{1}{2}\bar{\chi}^{aA} \cdot \nabla_\mu \chi_{aB}\gamma^\mu{}_A{}^B - \frac{1}{2}ig^2 A_\mu{}^b A^{\mu a} A_\nu{}^d A^{\nu c} tr[lie]_{abcd} + \frac{1}{2}ig^2 A_\mu{}^b A^{\mu a} A_\nu{}^d A^{\nu c} tr[lie]_{acbd}$

$+ \frac{1}{2}igA^{\mu a}\bar{\chi}_b{}^A \cdot \chi_{cB}\gamma_{\mu A}{}^B f_a{}^{bc} + igA^{\mu a}A^{\nu b}f_{abc}\nabla_\nu A_\mu{}^c + \frac{1}{2}\nabla_\mu A_{\nu a}\nabla^\nu A^{\mu a} - \frac{1}{2}\nabla_\nu A_{\mu a}\nabla^\nu A^{\mu a}$

Now gauge and supersymmetry invariance can be checked. We define the gauge transformations

```
gauge[xi_][AA[μ_, a_]] := DD[μ]@xi[a]
gauge[xi_][FF[μ_, ν_, a_]] := - I g With[{b = DummyIn[lie], c = DummyIn[lie]},
```

```
            lief[a, -b, -c] xi[b] FF[μ, ν, c]]
    gauge[xi_][chi[a_, B_]] := - I g With[{b = DummyIn[lie], c = DummyIn[lie]},
        lief[a, -b, -c] xi[b] chi[c, B]]
    gauge[xi_][barchi[a_, B_]] := - I g With[{b = DummyIn[lie], c = DummyIn[lie]},
        lief[a, -b, -c] xi[b] barchi[c, B]]
    gauge[xi_][auxD[a_] := - I g With[{b = DummyIn[lie], c = DummyIn[lie]},
        lief[a, -b, -c] xi[b] auxD[c]]
    gauge[xi_][_] := 0
```

and the supersymmetry transformations

```
    susy[eps_][AA[μ_, a_]] := With[{A = DummyIn[SpinM], B = DummyIn[SpinM]},
        - ConjugateSpinor[eps][A] · Gammagg[μ, -A, B] · chi[a, -B]]
    susy[eps_][FF[μ_, ν_, a_]] := With[{A = DummyIn[SpinM], B = DummyIn[SpinM]},
        ConjugateSpinor[eps][A] · Gammagg[μ, -A, B] · DD[ν]@chi[a, -B]
        - ConjugateSpinor[eps][A] · Gammagg[ν, -A, B] · DD[μ]@chi[a, -B]]
    susy[eps_][chi[a_, B_]] := With[{μ = DummyIn[TangentM], ν = DummyIn[TangentM],
        A = DummyIn[SpinM]}, 1/2 Gammagg[μ, ν, B, A] FF[-μ, -μ, a] eps[-A]
        + I auxD[a] GammaggStar[B, A] eps[-A]]
    susy[eps_][barchi[a_, B_]] := With[{μ = DummyIn[TangentM], ν =
        DummyIn[TangentM], A = DummyIn[SpinM]}, - 1/2 ConjugateSpinor[eps][A]
        Gammagg[μ, ν, -A, B] FF[-μ, -ν, a]
        + I auxD[a] ConjugateSpinor[eps][A] GammaggStar[-A, B]]
    susy[eps_][auxD[a_]] := With[{μ = DummyIn[TangentM], A = DummyIn[SpinM],
        B = DummyIn[SpinM], C = DummyIn[SpinM]}, I ConjugateSpinor[eps][A] ·
        GammaggStar[-A, B] · Gammagg[μ, -B, C] · DD[-μ]@chi[a, -C]]
    susy[eps_][_] := 0
```

where the last definitions ensure that the gauge or supersymmetry transformation of any other tensor vanishes. Using the functionality of the xPert package, we then consider a generic first-order perturbation, and set the perturbation of the metric to zero and the perturbation of any other tensor or spinor to its gauge or supersymmetry transformation:

```
    gaugetrafo[expr_, xi_] := Expand[ExpandPerturbation[Perturbation[expr, 1]]
        /. {Perturbationgg[__] :> 0, Perturbation[term_] :> gauge[xi][term]}]
    susytrafo[expr_, eps_] := Expand[ExpandPerturbation[Perturbation[expr, 1]]
        /. {Perturbationgg[__] :> 0, Perturbation[term_] :> susy[eps][term]}]
```

Let us check that the transformations of the field strength are the correct ones:

```
    CollectTensors@ReduceInvariantTraceTensors@GradAToF@SymmetrizeCovDs[
        ChangeCovD[ChangeCovD[gaugetrafo[FToGradA[FF[-μ, -ν, a]], xi]
        - gaugetrafo[FF[-μ, -ν, a], xi], DD, PD], PD, CD]]
    CollectTensors@GradAToF@SymmetrizeCovDs[ChangeCovD[ChangeCovD[
        susytrafo[FToGradA[FF[-μ, -ν, a]], eps]
        - susytrafo[FF[-μ, -ν, a], eps], DD, PD], PD, CD]]
```

0

0 .

For the action, a gauge transformation gives

```
gtf = gaugetrafo[action3, xi]
```

$$-\tfrac{1}{2}igD_aD^cf^a{}_{bc}\xi^b + \tfrac{1}{2}ig\bar{\chi}^{cA}\cdot\nabla_\mu\chi_{aB}\gamma^\mu{}_A{}^Bf^a{}_{bc}\xi^b - \tfrac{1}{2}igD^aD^dcartankilling_{ba}f^b{}_{cd}\xi^c - \ldots$$

To show that this vanishes, we have to bring each term into canonical form: change the gauge-covariant derivatives into ∇, symmetrise covariant derivatives, contract all metrics, reduce the invariant tensors of the Lie algebra and collect all terms. We then obtain

```
mycanon[expr_] := CollectTensors@ReduceInvariantTraceTensors@ContractMetric@
    SymmetrizeCovDs@Expand[ChangeCovD[ChangeCovD[expr, DD, PD], PD, CD]]
mycanon[gtf]
```

** DefTensor: Defining tensor Invlie5[a,b,c,d,e].

0 ,

which shows that the action (in this case, even the Lagrangian) is gauge invariant. To obtain this result, `ReduceInvariantTraceTensors` expanded a product of a rank-4 invariant tensor with structure constants coming from the gauge transformation into rank-5 invariant tensors, which then cancelled.

For supersymmetry invariance, we have to work a bit harder. First perform again a perturbation of the action, insert the supersymmetry transformations, and use the same simplifications:

```
stf = susytrafo[action3, eps]
```

$$\tfrac{1}{4}\bar{\epsilon}^F\cdot\nabla_\mu\chi_{aB}F_{\alpha\beta}{}^a\gamma^\mu{}_A{}^B\gamma^{\alpha\beta}{}_F{}^A + \tfrac{1}{2}iD_a\bar{\epsilon}^A\cdot D_\alpha\chi^a{}_F\gamma^\alpha{}_B{}^F\gamma^*[g]_A{}^B$$
$$+ \tfrac{1}{2}iD^acartankilling_{ba}\bar{\epsilon}^A\cdot D_\alpha\chi^b{}_F\gamma^\alpha{}_B{}^F\gamma^*[g]_A{}^B - \tfrac{1}{2}iD^a\bar{\epsilon}^F\cdot\nabla_\mu\chi_{aB}\gamma^\mu{}_A{}^B\gamma^*[g]_F{}^A + \ldots$$

We see that there are products of (generalised) γ matrices (the $\gamma^\mu{}_A{}^B\gamma^{\alpha\beta}{}_B{}^F$ in the first term) that can be expressed in terms of generalised γ matrices of higher order, and that there are both $\bar{\epsilon}\chi$ and $\bar{\chi}\epsilon$ terms. The first issue can be solved using `JoinGammaMatrices`, and the second with `SortSpinor`. This gives

```
stf2 = mycanon@SortSpinor[JoinGammaMatrices[stf], eps → bareps]
```

$$\tfrac{1}{2}\bar{\epsilon}^A\cdot\nabla_\nu\chi_{aB}F_\mu{}^{\nu a}\gamma^\mu{}_A{}^B + \tfrac{1}{4}\bar{\epsilon}^A\cdot\nabla_\alpha\chi_{aB}F^{\mu\nu a}\gamma_{\mu\nu}{}^\alpha{}_A{}^B + \tfrac{1}{12}D^a\bar{\epsilon}^A\cdot\nabla_\beta\chi_{aB}\epsilon g_{\mu\nu\alpha}{}^\beta\gamma^{\mu\nu\alpha}{}_A{}^B$$
$$+ ig^2A_\mu{}^bA^{\mu a}A^{\nu c}\bar{\epsilon}^A\cdot\chi_{dB}\gamma_{\nu A}{}^Btr[lie]_{abc}{}^d + ig^2A_\mu{}^bA^{\mu a}A^{\nu c}\bar{\epsilon}^A\cdot\chi_{dB}\gamma_{\nu A}{}^Btr[lie]_{ab}{}^d{}_c - \ldots ,$$

and the Lagrangian is not invariant under supersymmetry. However, the result is a surface term (total derivative) such that the action is invariant:

```
surface = 1/4 mycanon@CD[-μ][ bareps[A] · ( 2 Gammagg[-ν, -A, B] gg[μ, -ρ]
    - Gammagg[-ν, -ρ, μ, -A, B] ) · chi[-a, -B] FF[ρ, ν, a] - 1/3 auxD[a]
    epsilongg[μ, ν, ρ, σ] bareps[A] · Gammagg[-ν, -ρ, -σ, -A, B] · chi[-a, -B] ];
stf3 = mycanon@FToGradA[stf2 - surface]
```

$$-\tfrac{1}{2}ig\bar{\chi}_a{}^A\cdot\chi_{bB}\bar{\epsilon}^F\cdot\chi_{cG}\gamma_{\mu F}{}^G\gamma^\mu{}_A{}^Bf^{abc} + \tfrac{1}{2}A^{\mu a}\bar{\epsilon}^A\cdot\chi_{aB}\gamma^{\nu\rho\sigma}{}_A{}^BR[\nabla]_{\mu\nu\rho\sigma} .$$

The second term vanishes because $\gamma^{\nu\rho\sigma}$ is totally antisymmetric, and by the Bianchi identity $R_{\mu[\nu\rho\sigma]} = 0$, which is taken care of by the function `RiemannYoungProject` from the

XTRAS package. On the other hand, the first term vanishes after using Fierz identities and symmetrising using the Majorana flip relations:

```
mycanon@SpinorFlipSymmetrize@JoinGammaMatrices@Expand[RiemannYoungProject@stf3
    /. {HoldPattern[x__ · chi[i1__] barchi[i2__] · y__] :> 1/3 x · chi[i1]
    barchi[i2] · y - 2/3 x · y FierzExpand[barchi[i2],chi[i1]]}]
0 .
```

Because of the automatic rules set up for the noncommutative product, in pattern matching `HoldPattern` or `Verbatim` must be used. The function `FierzExpand` gives the basic two-spinor Fierz identity (valid in this form for both commuting and anticommuting spinors)

$$\bar{\psi}^A \chi_B = 2^{-[d/2]} \sum_{k=0}^{d} \frac{1}{k!} (-1)^{k(k-1)/2} \gamma_{\mu_1 \cdots \mu_k B}{}^A \left(\bar{\psi}^C \gamma^{\mu_1 \cdots \mu_k}{}_C{}^D \chi_D \right), \tag{3.8}$$

which is nothing else than a completeness relation in the space of $[d/2] \times [d/2]$ matrices [15].

However, finding a suitable Fierz rearrangement in each case might not be easy, and a more systematic approach is to be preferred. This is the one of d'Auria, Fré, Maina and Regge [16] based on the decomposition of the product of group representations into irreducible components, applied to the Lorentz group. Consider two Dirac spinors in 4 dimensions, each transforming in the $(\frac{1}{2}, 0) \oplus (0, \frac{1}{2})$ representation of $\mathrm{SL}(2, \mathbb{C})$, the universal cover of the Lorentz group. The tensor product of two decomposes according to the well-known rules as

$$\left[(\tfrac{1}{2}, 0) \oplus (0, \tfrac{1}{2}) \right] \otimes \left[(\tfrac{1}{2}, 0) \oplus (0, \tfrac{1}{2}) \right] = (0, 0) \oplus (\tfrac{1}{2}, \tfrac{1}{2}) \oplus [(1, 0) \oplus (0, 1)] \oplus (\tfrac{1}{2}, \tfrac{1}{2}) \oplus (0, 0), \tag{3.9}$$

which are two scalars, two vectors and an antisymmetric tensor. This is exactly the result obtained using the Fierz identity (3.8) in four dimensions:

$$\begin{aligned}
\bar{\psi}^A \psi_B &= \frac{1}{4} \delta_B{}^A \left(\bar{\psi}\psi \right) + \frac{1}{4} \gamma_{\mu B}{}^A \left(\bar{\psi}\gamma^\mu \psi \right) - \frac{1}{8} \gamma_{\mu\nu B}{}^A \left(\bar{\psi}\gamma^{\mu\nu}\psi \right) - \frac{1}{24} \gamma_{\mu\nu\rho B}{}^A \left(\bar{\psi}\gamma^{\mu\nu\rho}\psi \right) \\
&\quad + \frac{1}{96} \gamma_{\mu\nu\rho\sigma B}{}^A \left(\bar{\psi}\gamma^{\mu\nu\rho\sigma}\psi \right) \\
&= \frac{1}{4} \delta_B{}^A \Psi^{(2,\mathbf{1})} + \frac{1}{4} \gamma^\mu{}_B{}^A \Psi^{(2,\mathbf{4'})}_\mu - \frac{1}{8} \gamma^{\mu\nu}{}_B{}^A \Psi^{(2,\mathbf{6})}_{\mu\nu} - \frac{i}{4} (\gamma_* \gamma^\mu)_B{}^A \Psi^{(2,\mathbf{4})}_\mu - \frac{i}{4} \gamma_{*B}{}^A \Psi^{(2,\mathbf{1'})},
\end{aligned} \tag{3.10}$$

where the scalars are $\Psi^{(2,\mathbf{1})} = \bar{\psi}\psi$ and $\Psi^{(2,\mathbf{1'})} = \frac{1}{24} \epsilon^{\mu\nu\rho\sigma} \bar{\psi} \gamma_{\mu\nu\rho\sigma} \psi = i\bar{\psi}\gamma_*\psi$, the vectors are $\Psi^{(2,\mathbf{4})}_\mu = \frac{1}{6} \epsilon_{\mu\nu\rho\sigma} \bar{\psi}\gamma^{\nu\rho\sigma}\psi = -i\bar{\psi}\gamma_*\gamma_\mu\psi$ and $\Psi^{(2,\mathbf{4'})}_\mu = \bar{\psi}\gamma_\mu\psi$, and the antisymmetric tensor is $\Psi^{(2,\mathbf{6})}_{\mu\nu} = \bar{\psi}^C \gamma_{\mu\nu} \psi$. Here, we used the alternative convention of labeling objects transforming in irreducible representations by their dimension $\boldsymbol{d}$ instead of the rank[4], and it is easily checked that applying the Fierz identity (3.8) again to each tensor $\Psi^{(2,\boldsymbol{d})}$ the same tensor is obtained, such that they indeed transform irreducibly. For a Majorana spinor, Majorana flip relations show that $\Psi^{(2,\mathbf{4'})}_\mu = 0 = \Psi^{(2,\mathbf{6})}_{\mu\nu}$, which is consistent with dimension counting:

---

[4]In the work of d'Auria, Fré, Maina and Regge [16], yet another labeling is used, where instead of (j, k) one writes $[j + k, |j - k|]$.

a Majorana spinor in four dimensions has four degrees of freedom, such that the $\frac{4\cdot 3}{2} = 6$ degrees of freedom in a tensor product of anticommuting spinors are decomposed into $1 + 1 + 4$ of two scalars and a vector. In the following, we only consider Majorana spinors.

For the product of three spinors, the tensors transforming in irreducible representations have a free spinor index, and any trace and their contraction on the spinor and any tensor index with a γ matrix (spinor trace) vanishes. For example, we have the decomposition

$$\Psi_\mu^{(2,\mathbf{4})}\psi_A = \Psi_{\mu A}^{(3,\mathbf{12})} + \frac{1}{4}\gamma_{\mu A}{}^B \Psi_B^{(3,\mathbf{4'})} \,, \tag{3.11}$$

where $\gamma^\mu{}_A{}^B \Psi_{\mu B}^{(3,\mathbf{12})} = 0$, analogous to the decomposition of a symmetric 2-tensor into a trace and a traceless part. Applying the Fierz identity (3.8), one actually obtains $\Psi_{\mu B}^{(3,\mathbf{12})} = 0$ and $\Psi_A^{(3,\mathbf{4'})} = -\frac{1}{6}\epsilon^{\mu\nu\rho\sigma}\gamma_{\mu\nu\rho\sigma A}{}^B \Psi_B^{(3,\mathbf{4})}$, where

$$\Psi_A^{(3,\mathbf{4})} = \Psi^{(2,\mathbf{1})}\psi_A \,, \tag{3.12}$$

and in addition the decomposition

$$\Psi^{(2,\mathbf{1'})}\psi_A = -\frac{1}{24}\epsilon^{\mu\nu\rho\sigma}\gamma_{\mu\nu\rho\sigma A}{}^B \Psi_B^{(3,\mathbf{4})} \,. \tag{3.13}$$

Taking all together, one obtains

$$\psi_A \bar\psi^B \psi_C = -\frac{1}{4}\left[\delta_{[A}^B \delta_{C]}^D + \frac{1}{6}\gamma_{\mu\nu\rho[A}{}^B \gamma^{\mu\nu\rho}{}_{C]}{}^D - \frac{1}{24}\gamma_{\mu\nu\rho\sigma[A}{}^B \gamma_{C]}^{\mu\nu\rho\sigma\,D}\right]\Psi_D^{(3,\mathbf{4})} \,, \tag{3.14}$$

which agrees with the counting of degrees of freedom for anticommuting Majorana spinors: $4(4-1)(4-2)/(2\cdot 3) = 4$. Lastly, for four spinors it follows that

$$\psi_A \bar\psi^B \psi_C \bar\psi^D = -\frac{1}{16}\left[\delta_{[A}^B \delta_{C]}^D + \frac{1}{6}\gamma_{\mu\nu\rho[A}{}^B \gamma^{\mu\nu\rho}{}_{C]}{}^D - \frac{1}{24}\gamma_{\mu\nu\rho\sigma[A}{}^B \gamma_{C]}^{\mu\nu\rho\sigma\,D}\right]\Psi^{(4,\mathbf{1})} \tag{3.15}$$

with

$$\Psi^{(4,\mathbf{1})} = \left(\bar\psi^A \psi_A\right)^2 \,, \tag{3.16}$$

and the product of five or more anticommuting spinors vanishes since at least two components will be equal.

For Lie-algebra valued spinors, one has to decompose representations of the product of Lorentz group and symmetric group (for permutations of the Lie algebra indices), where the representations of the symmetric group are labeled by Young tableaux; see appendix A for details. The decomposition of a product of spinors into tensors transforming in irreducible representations is done using `IrreducibleSpinDecompose`, and the γ matrix orthogonality conditions and projections on the Young tableau (for Lie-algebra valued spinors) are imposed with `IrreducibleSpinProject`. Invariance of the action under supersymmetry transformation then follows straightforwardly:

```
IrreducibleSpinProject[mycanon@JoinGammaMatrices@mycanon@
    IrreducibleSpinDecompose[RiemannYoungProject@stf3, chi], chi]
```

`0.`

Lastly, we also want to check the closure (3.6) of the supersymmetry transformations on the fields. To reduce clutter, we now suppress informational output.

```
$DefInfoQ = False;
$UndefInfoQ = False;
```

We declare two constant Grassmann-odd Majorana spinors ϵ_i and define the auxiliary vector b^μ (3.7):

```
DefOddSpinor[eps1[-A], M, PrintAs → "ε1"];
DefOddSpinor[eps2[-A], M, PrintAs → "ε2"];
eps1 /: CD[_]@eps1[_] := 0
bareps1 /: CD[_]@bareps1[_] := 0
eps2 /: CD[_]@eps2[_] := 0
bareps2 /: CD[_]@bareps2[_] := 0
bb[mu_] := With[A = DummyIn[SpinM], B = DummyIn[SpinM], 2 bareps1[A] ·
    Gammagg[mu, -A, B] · eps2[-B]]
```

We then compute the commutator of two supersymmetry transformations and obtain for the gauge boson the required result (3.6)

```
mycanon[SortSpinor[JoinGammaMatrices@mycanon[susytrafo[susytrafo[AA[-μ, a],
    eps1], eps2] - susytrafo[susytrafo[AA[-μ, a], eps2], eps1]],
    eps1 → bareps1] + bb[ν] FF[-μ, -ν, a]]
```

`0.`

To bring all terms into a form where only the conjugate spinor $\bar{\epsilon}_1$ appears, we have used Majorana flip relations [15]

$$\bar{\psi}^A \gamma_{\mu_1 \cdots \mu_k}{}_A{}^B \chi_B = s_k \bar{\chi}^A \gamma_{\mu_1 \cdots \mu_k}{}_A{}^B \psi_B \,, \tag{3.17}$$

where the sign $s_k = \pm 1$ depends on the number of space-time dimensions and is given in [15], table 3.1 (the choices in boldface). They are implemented in FIELDSX by the functions `FlipSpinor`, `SpinorFlipSymmetrize` and `SortSpinor`, with the last one used above.

For the gaugino χ^a, we compute

```
erg = mycanon@TimesToCenterDot@JoinGammaMatrices@mycanon[mycanon@
    JoinGammaMatrices@mycanon[susytrafo[susytrafo[chi[a, -A], eps1], eps2]
    - susytrafo[susytrafo[chi[a, -A], eps2], eps1]]]
```

$$\overline{\epsilon_1}^B \cdot \epsilon_{2G} \cdot \nabla_\beta \chi^a{}_F \gamma^\alpha{}_B{}^F \gamma_\alpha{}^\beta{}_A{}^G - \overline{\epsilon_2}^B \cdot \epsilon_{1G} \cdot \nabla_\beta \chi^a{}_F \gamma^\alpha{}_B{}^F \gamma_\alpha{}^\beta{}_A{}^G + \ldots \,.$$

Here we have used `TimesToCenterDot` to ensure that all non-commutative objects are properly multiplied with the non-commutative product. We then perform a Fierz rearrangement to ensure that the two parameters ϵ_i are contracted together, and use `GammaMatricesToDual` which replaces γ matrices with three and four indices by their duals to shorten the required γ matrix algebra.

```
erg2 = CollectTensors@JoinGammaMatrices@CollectTensors@JoinGammaMatrices@
    GammaMatricesToDual[erg /. {HoldPattern[bareps1[A_] · eps2[B_] · t_] :>
    FierzExpand[bareps1[A], eps2[B]] · t, HoldPattern[bareps2[A_] · eps1[B_] · t_]
    :> FierzExpand[bareps2[A], eps1[B]] · t, HoldPattern[bareps1[A_] · t_
    · eps2[B_]] :> -FierzExpand[bareps1[A], eps2[B]] · t, HoldPattern[bareps2[A_]
    · t_ · eps1[B_]] :> -FierzExpand[bareps2[A], eps1[B]] · t}];
```

Lastly, we bring all into a form where only the conjugate spinor $\bar{\epsilon}_1$ appears, using again Majorana flip relations.

```
erg3 = CollectTensors@JoinGammaMatrices@SortSpinor[erg2, eps1 → bareps1];
mycanon@JoinGammaMatrices[mycanon[erg3 - (bb[ν] CD[-ν]@chi[a, -A]
    + gauge[xi][chi[a, -A]] /. {xi[a_] :> With[{ν = DummyIn@TangentM},
    - bb[ν] AA[-ν, a]]})]]
```

0,

which is the required result (3.6). Lastly, we check the susy algebra on the auxiliary field D^a:

```
erg = mycanon@JoinGammaMatrices@SortSpinor[mycanon[susytrafo[susytrafo[
    auxD[a], eps1], eps2] - susytrafo[susytrafo[auxD[a], eps2], eps1]],
    eps1 → bareps1];
```

We again use the decomposition of a product of spinors into tensors transforming in irreducible representations, this time for the gaugino χ^a:

```
erg2 = mycanon@FToGradA@JoinGammaMatrices@mycanon@
    IrreducibleSpinDecompose[erg, chi];
```

and obtain the required result (3.6)

```
mycanon@RiemannYoungProject@mycanon[erg2 - (bb[ν] CD[-ν]@auxD[a]
    + gauge[xi][auxD[a]] /. {xi[a_] :> With[{ν = DummyIn@TangentM},
    - bb[ν] AA[-ν, a]]})]
```

0.

FieldsX is also fully integrated with the xPert package for perturbative expansions, as we have already seen previously for computing the gauge and supersymmetry variations. In particular, one can easily obtain the stress tensor by performing a general perturbation of the action (where the variation of the γ matrices and the spin connection follows [17]), setting all perturbations but the metric one to zero. Using `LeftVarD` then performs the needed integration by parts, and it follows that

```
actionvar = mycanon[2/Sqrt[-Detgg[]] (ExpandPerturbation@Perturbation[action
    Sqrt[-Detgg[]], 1] /. {Perturbation[_] :> 0})];
stress = mycanon[LeftVarD[Perturbationgg[LI[1], μ, ν], CD]@actionvar
    /. {delta[-LI[1], LI[1]] → 1}]
```

$F_\mu{}^{\alpha a} F_{\nu\alpha a} + \frac{1}{4}\bar{\chi}^{aA} \cdot \nabla_\nu \chi_{aB}\gamma_{\mu A}{}^B + \dots$,

which becomes the standard expression after expressing everything in terms of the gauge-covariant derivative:

```
stress2 = CollectTensors@JoinGammaMatrices@CollectTensors@
    ChangeCovD[ChangeCovD[stress, CD, PD], PD, DD]
```

$F_\mu{}^{\alpha a}F_{\nu\alpha a} + \frac{1}{8}\bar\chi^{aA}\cdot D_\nu\chi_{aB}\gamma_{\mu A}{}^B + \frac{1}{8}\chi^a{}_B\cdot D_\nu\bar\chi_a{}^A\gamma_{\mu A}{}^B + \frac{1}{8}\bar\chi^{aA}\cdot D_\mu\chi_{aB}\gamma_{\nu A}{}^B$
$+ \frac{1}{8}\chi^a{}_B\cdot D_\mu\bar\chi_a{}^A\gamma_{\nu A}{}^B + \frac{1}{2}D_a D^a g_{\mu\nu} - \frac{1}{4}F_{\alpha\beta a}F^{\alpha\beta a}g_{\mu\nu} - \frac{1}{4}\bar\chi^{aA}\cdot D_\alpha\chi_{aB}\gamma^\alpha{}_A{}^B g_{\mu\nu}$
$- \frac{1}{4}\chi^a{}_B\cdot D_\alpha\bar\chi_a{}^A\gamma^\alpha{}_A{}^B g_{\mu\nu}\,.$

3.2 BRST formalism

A very general approach to the quantisation of gauge theories is the BRST (Becchi–Rouet–Stora–Tyutin) formalism [18, 19], which is sufficient for theories with closed gauge algebras (such as supersymmetric theories with auxiliary fields). For theories with open gauge algebras (that only close modulo the equations of motion), the BV or field–antifield formalism [20–22] is necessary, which of course can also be treated with FIELDSX. Because extensive reviews of the BRST(-BV) formalism exist [23–25], I only list the essential steps: For each symmetry transformation $\delta_\xi^{(s)}$ with parameter ξ, one introduces a ghost field $c^{(s)}$ (of ghost number 1), an antighost field $\bar c^{(s)}$ (of ghost number -1) and an auxiliary field $B^{(s)}$ (of ghost number 0), where the ghost and antighost fields have opposite Grassmann parity to that of the parameter ξ, while $B^{(s)}$ has the same Grassmann parity; for global symmetries, the non-minimal fields (antighost and auxiliary field) are not needed [26, 27].[5] One then defines the (Grassmann-odd) BRST differential s, which acts on the original fields of the theory as the sum of all symmetry transformations, with the parameters replaced by the corresponding ghosts. The action of s on the ghosts is determined by requiring that $\mathsf{s}^2 = 0$ when acting on the original fields; the closure of the gauge algebra ensures that this also entails $\mathsf{s}^2 = 0$ on the ghosts. For the non-minimal fields, one sets $\mathsf{s}\bar c^{(s)} = B^{(s)}$ and $\mathsf{s}B^{(s)} = 0$, which ensures that the BRST differential is nilpotent, $\mathsf{s}^2 = 0$, when acting on anything.

The relevance of the BRST(-BV) formalism lies in the fact that all quantities of interest can be obtained as (representatives of) the cohomology of the BRST differential s in different gradings, and that Slavnov-Taylor-Ward-Takahashi identities can be straightforwardly derived [25, 34, 35]. In particular, the Wess-Zumino consistency conditions [36] are nothing else but the statement of s invariance for the integrated anomaly, and the problem of determining all consistent interactions for a given free theory is reduced to finding all elements of the cohomology at ghost number 0 [37, 38]; see [39, 40] for the case of $\mathcal{N} = 1$ super-Yang–Mills theory. Gauge-fixing terms are incorporated by choosing a gauge-fixing fermion Ψ of ghost number -1 and adding $\mathsf{s}\Psi$ to the action. Since the BRST differential incorporates all the symmetries under which the action S is invariant, one has $\mathsf{s}S = 0$, and since $\mathsf{s}^2 = 0$ adding the gauge-fixing term does not change this. For $\mathcal{N} = 1$ super-Yang–Mills theory, one needs the ghost for gauge transformations c^a (a Grassmann-odd Lie-algebra valued scalar), the ghost for supersymmetry transformations θ (a constant Grassmann-even

---

[5]For reducible symmetries (which become symmetries of the ghost fields) one has to repeat the procedure, leading to "ghosts for ghosts" or even higher "n-th order ghosts" [28–33]; the ghost number of the n-th order (anti-)ghost is $\pm n$.

Majorana spinor), and the ghost for translations α^ρ (a constant Grassmann-odd vector), which is necessary to close the supersymmetry transformations (3.6).

We thus define all the required gradings and fields:[6]

```
DefGrading[{Dimension, GhostNumber}];
DefOddTensor[cc[a], M, PrintAs → "c"];
DefEvenSpinor[theta[-A], M, SpinorType → Majorana, PrintAs → "θ"];
DefOddTensor[alpha[μ], M, PrintAs → "α"];
CD[__]@theta[__] ^:= 0;
CD[__]@bartheta[__] ^:= 0;
CD[__]@alpha[__] ^:= 0;
SetGrading[AA, {Dimension → 1, GhostNumber → 0}];
SetGrading[FF, {Dimension → 2, GhostNumber → 0}];
SetGrading[auxD, {Dimension → 2, GhostNumber → 0}];
SetGrading[cc, {Dimension → 1, GhostNumber → 1}];
SetGrading[chi, {Dimension → 3/2, GhostNumber → 0}];
SetGrading[theta, {Dimension → 1/2, GhostNumber → 1}];
SetGrading[alpha, {Dimension → 0, GhostNumber → 1}];
```

We then define the BRST transformations using the predefined `BRST` operator:

```
AA /: BRST[AA[μ_, a_]] := gauge[cc]@AA[μ, a] + susy[theta]@AA[μ, a]
    + With[{ρ = DummyIn@TangentM}, alpha[ρ] CD[-ρ]@AA[μ, a]]
FF /: BRST[FF[μ_, ν_, a_]] := gauge[cc]@FF[μ, ν, a] + susy[theta]@FF[μ, ν, a]
    + With[{ρ = DummyIn@TangentM}, alpha[ρ] CD[-ρ]@FF[μ, ν, a]]
chi /: BRST[chi[a_, -A_]] := gauge[cc]@chi[a, -A] + susy[theta]@chi[a, -A]
    + With[{ρ = DummyIn@TangentM}, alpha[ρ] · CD[-ρ]@chi[a, -A]]
barchi /: BRST[barchi[a_, A_]] := gauge[cc]@barchi[a, A] + susy[theta]@
    barchi[a, A] + With[{ρ = DummyIn@TangentM}, alpha[ρ] · CD[-ρ]@barchi[a, A]]
auxD /: BRST[auxD[a_]] := gauge[cc]@auxD[a] + susy[theta]@auxD[a]
    + With[{ρ = DummyIn@TangentM}, alpha[ρ] CD[-ρ]@auxD[a]]
cc /: BRST[cc[a_]] := With[{A = DummyIn@SpinM, B = DummyIn@SpinM, mu =
    DummyIn@TangentM}, AA[-mu, a] bartheta[A] Gammagg[mu, -A, B] theta[-B]]
    - I/2 g With[{b = DummyIn@lie, c = DummyIn@lie}, lief[a, b, c] cc[-b] · cc[-c]]
    + With[{mu = DummyIn@TangentM}, alpha[mu] · CD[-mu]@cc[a]]
theta /: BRST[theta[A_]] := 0
bartheta /: BRST[bartheta[A_]] := 0
alpha /: BRST[alpha[mu_]] := With[{A = DummyIn@SpinM, B = DummyIn@SpinM},
    - bartheta[A] Gammagg[mu, -A, B] theta[-B]]
```

To check BRST invariance of the action, we also have to define the BRST transformations of γ matrices and the ϵ tensor, which are not predefined (since they change under diffeomorphisms, which are needed for gravity theories), and to impose that it commutes with covariant derivatives:

---

[6]The given choice of engineering dimensions of the (anti-)ghosts and auxiliary fields is such that `s` augments the engineering dimension by 1.

```
Map[Function[BRST[#[i___]] := 0], $GammaMatrices];
BRST[epsilongg[__]] := 0;
BRST[CD[i__][expr_]] := CD[i]@BRST[expr];
BRST[DD[i__][expr_]] := BRST@CollectTensors@ChangeCovD[
    ChangeCovD[DD[i]@expr, DD, PD], PD, CD];
```

Let us check that with these definitions the BRST operator s is nilpotent:

```
IrreducibleSpinProject[mycanon@TimesToCenterDot@JoinGammaMatrices@mycanon@
    IrreducibleSpinDecompose[mycanon@JoinGammaMatrices@mycanon@
    BRST[BRST[cc[a]]],theta],theta]
BRST[BRST[alpha[μ]]]
mycanon@FToGradA@JoinGammaMatrices@mycanon@IrreducibleSpinDecompose[mycanon@
    JoinGammaMatrices@mycanon@BRST[BRST[AA[μ, a]]], theta]
mycanon@TimesToCenterDot@JoinGammaMatrices@mycanon@IrreducibleSpinDecompose[
    mycanon@JoinGammaMatrices@mycanon@BRST[BRST[chi[a, -A]]], theta]
CollectTensors@RiemannYoungProject@JoinGammaMatrices@mycanon@FToGradA@
    IrreducibleSpinDecompose[mycanon@JoinGammaMatrices@mycanon@
    IrreducibleSpinDecompose[mycanon@JoinGammaMatrices@mycanon@
    BRST[BRST[auxD[a]]], theta], chi]
```

0

0

$$-\frac{1}{2}A^{\alpha a}\alpha_\beta \cdot \alpha_\gamma R[\nabla]^\mu{}_\alpha{}^{\beta\gamma}$$
$$-\frac{1}{8}\alpha_\gamma \cdot \alpha_\delta \cdot \chi^a{}_B\gamma^{\alpha\beta}{}_A{}^B R[\nabla]_{\alpha\beta}{}^{\gamma\delta}$$
$$-\frac{1}{12}\bar\theta^A\alpha_\beta \cdot \chi^a{}_B\epsilon g_{\alpha\gamma\delta\mu}\gamma^{\gamma\delta\mu}{}_A{}^B R[\nabla]^{\alpha\beta}\,,$$

which quite obviously only vanishes in a flat background, as is well known, such that for the following we set all curvature tensors to vanish:

```
RiemannCD[__] := 0
RicciCD[__] := 0
RicciScalarCD[] := 0
```

For the action, we obtain as before that it is invariant up to a surface term:

```
surface = - I/2 bartheta[A] GammaggStar[-A, B] Gammagg[μ, -B, F] chi[-a, -F]
    auxD[a] - 1/2 bartheta[A] Gammagg[-ν, -A, B] chi[a, -B] FF[μ, ν, -a]
    - 1/4 bartheta[A] Gammagg[μ, ν, ρ, -A, B] chi[-a, -B] FF[-ν, -ρ, a]
    - alpha[μ] · ReplaceDummies[action];
sn = CollectTensors@GammaMatricesToDual@IrreducibleSpinDecompose[
    CollectTensors@SpinorFlipSymmetrize@RiemannYoungProject@mycanon@
    FToGradA@CollectTensors@JoinGammaMatrices@mycanon@
    TimesToCenterDot[BRST[action] + CD[-μ]@surface], chi];
IrreducibleSpinProject[CollectTensors@JoinGammaMatrices@sn, chi]
```

0.

We now turn to cohomologies (and thus in particular anomalies). To compute them, besides a brute-force approach (which for any but the most trivial cases is too lengthy) one can choose between various methods such as descent equations [25, 34] or spectral sequences [41]. A very useful concept are filtrations [34], defined by a filtration operator $\mathcal{F}$ with non-negative integer eigenvalues. $\mathcal{F}$ can be defined by assigning each field of the theory a certain weight, and defining the action of $\mathcal{F}$ on a monomial by the sum of the weights of the fields in the monomial. Assume further that the BRST differential s can be written as a sum of differentials s_k with integer $k \geq 0$, such that $[\mathcal{F}, \mathsf{s}_k] = k\,\mathsf{s}_k$. Then it follows that the lowest-order term s_0 is itself nilpotent, and that the cohomology of s is isomorphic to a subset of the cohomology of s_0 [34]. That is, one can determine the cohomology of s by first computing the cohomology of s_0 and then checking whether each element can be extended. A simple filtration operator is given by assigning each field wright 1, such that $\mathcal{F}$ simply counts the number of fields, and in this case s_0 is the BRST differential of the free theory. In general, the choice of weights is obviously restricted by the fact that $\mathsf{s} = \sum_{k \geq 0} \mathsf{s}_k$, and that s_0 is non-trivial. If $\mathcal{F}$ commutes with derivatives, the same result applies to the relative cohomologies. Functions to work with filtrations (and check whether a choice for the weights leads to a viable filtration) are defined by FIELDSX, see section 4.13.

We would like to verify that the well-known super-chiral anomalies [42–45] satisfy the consistency criterion. The gauge part of the anomaly reads

$$\mathcal{A}^{\text{gauge}} = \int c^a \epsilon^{\mu\nu\rho\sigma} d_{abc} \left[\nabla_\mu A_\nu^b \nabla_\rho A_\sigma^c + \frac{\mathrm{i}}{4} g f_{cde} \nabla_\mu \left(A_\nu^b A_\rho^d A_\sigma^e \right) \right] \mathrm{d}x \,, \qquad (3.18)$$

while the supersymmetric part of the anomaly is[7]

$$\mathcal{A}^{\text{susy}} = \int \epsilon^{\mu\nu\rho\sigma} d_{abc} \, \bar{\theta} \left[\frac{1}{8} \gamma_{\mu\nu\rho\sigma} \chi^a \left(\bar{\chi}^b \chi^c \right) - \gamma_\mu \chi^a \left(2 A_\nu^b \nabla_\rho A_\sigma^c + \frac{3}{4} \mathrm{i} g f_{bde} A_\nu^c A_\rho^d A_\sigma^e \right) \right] \mathrm{d}x \,.$$
$$(3.19)$$

Since gauge transformations close among themselves, the gauge transformation of $\mathcal{A}^{\text{gauge}}$ must vanish which is its consistency condition. On the other hand, because supersymmetry transformations close into a gauge transformation and a translation, the consistency condition for $\mathcal{A}^{\text{susy}}$ is [45]

$$\delta_\theta^{\text{susy}} \mathcal{A}^{\text{susy}} = \mathcal{A}^{\text{gauge}} \big|_{c^a \to -\bar{\theta}\gamma^\mu\theta A_\mu^a} \,. \qquad (3.20)$$

We thus first define the anomalies together with the needed surface terms:

```
lied = InvariantTraceTensor[lie, 3, Symmetric];
lied4 = InvariantTraceTensor[lie, 4];
gaugeanomaly = mycanon@mycanon[epsilongg[-μ, -ν, -ρ, -σ] cc[a] lied[-a, -b, -c]
    (CD[μ]@AA[ν, b] CD[ρ]]@AA[σ, c] + I/4 g lief[c, -d, -e]
    CD[μ]@(AA[ν, b] AA[ρ, d] AA[σ, e]))];
gaugeanomalysurface = mycanon[I g epsilongg[μ, ν, ρ, σ] (CD[-σ]@AA[-ν, a]
    AA[-ρ, b] cc[c] · cc[d] lied4[-a, -b, -c, -d] - AA[-ν, a] CD[-σ]@AA[-ρ, b]
    cc[c] · cc[d] lied4[-a, -b, -c, -d] + AA[-ν, a] AA[-ρ, b] cc[c] ·
```

<hr>

[7]This differs from the expression given in [45], since they use Weyl instead of Majorana fermions.

```
        CD[-σ]@cc[d] lied4[-a, -c, -b, -d] + g AA[-ν, a] AA[-ρ, b] AA[-σ, c]
        cc[d] · cc[e] Invlie5[-a, -b, -c, -d, -e])];
    susyanomaly = mycanon[lied[-a, -b, -c] epsilongg[-μ, -ν, -ρ, -σ] (1/8
        bartheta[A] Gammagg[μ, ν, ρ, σ, -A, B] chi[a,-B] · barchi[b,F] · chi[c,-F]
        - bartheta[A] · Gammagg[ν, -A, B] · chi[a,-B] (2 AA[ν, b] CD[ρ]@AA[σ, c]
        + 3/4 I g lief[b, d, e] AA[ν, c] AA[ρ, -d] AA[σ, -e]))];
    susyanomalysurface = mycanon[lied[-a, -b, -c] epsilongg[-ν, -ρ, -σ, -α] (1/48
        AA[-β, a] barchi[b,A] · Gammagg[ν, ρ, σ, α, -A, B] · chi[c, -B] bartheta[F]
        Gammagg[μ, β, -F, G] theta[-G] - 1/4 AA[ν, a] barchi[b, A] · chi[c, -A]
        gg[μ, α] bartheta[F] Gammagg[ρ, σ, -F, G] theta[-G] - 1/12 AA[ν, a]
        barchi[c, A] · Gammagg[ρ, σ, α, -A, B] · chi[b, -B] bartheta[F]
        Gammagg[μ, -F, G] theta[-G] - 1/12 AA[-β, a] barchi[b, A] · Gammagg[ν, ρ, σ,
        -A, B] · chi[c, -B] gg[μ, α] bartheta[F] Gammagg[β, -F, G] theta[-G])];
```

Since gauge transformations close among themselves, we can define a consistent filtration of the BRST differential:

```
    CheckFiltration[{AA → 0, FF →  0, chi → 0, barchi → 0, cc → 0, auxD → 0,
        theta → 2, bartheta → 1, alpha → 2}, BRST, Display → Full]
```

$$BRST0[A_\alpha{}^a] \to igA_\alpha{}^b c^c f^a{}_{bc} + \nabla_\alpha c^a$$
$$BRST0[F_{\alpha\beta}{}^a] \to -igc^b F_{\alpha\beta}{}^c f^a{}_{bc}$$
$$BRST0[\chi^a{}_A] \to -igc^b \cdot \chi^c{}_A f^a{}_{bc}$$
$$BRST0[\bar\chi^{aA}] \to -igc^b \cdot \bar\chi^{cA} f^a{}_{bc}$$
$$BRST0[c^a] \to -\tfrac{1}{2}igc_b \cdot c_c f^{abc}$$
$$BRST0[D^a] \to -igD^c c^b f^a{}_{bc}$$
$$BRST0[\theta_A] \to 0$$
$$BRST0[\bar\theta^A] \to 0$$
$$BRST0[\alpha^\alpha] \to 0\,,$$

and we see that indeed only the gauge transformations survive with this choice. We can thus define a filtrated differential and check that the gauge transformation of $\mathcal{A}^{\text{gauge}}$ vanishes:

```
    DefOddDifferential[BRSTgauge]
    Filtrate[{AA → 0, FF → 0, chi → 0, barchi → 0, cc → 0, auxD → 0,
        theta → 2, bartheta → 1, alpha → 2}, BRST → BRSTgauge]
    BRSTgauge[epsilongg[__]] := 0
    BRSTgauge[CD[i_]@AA[j__]] := CD[i]@BRSTgauge[AA[j]]
    mycanon[BRSTgauge[gaugeanomaly] + CD[-μ]@gaugeanomalysurface]
```
0.

On the other hand, supersymmetry transformations close into a gauge transformation and a translation, such that we can not use filtrations to check the corresponding consistency condition. Instead, we obtain an equivariant differential, for which we define the corresponding transformations directly:

```
DefOddDifferential[BRSTsusy]
BRSTsusy[expr_] := BRST[expr] /. {cc[_] :> 0, alpha[_] :> 0}
erg = mycanon@JoinGammaMatrices@Expand@IrreducibleSpinDecompose[mycanon@
    JoinGammaMatrices@mycanon[BRSTsusy[susyanomaly]
    + CD[-μ]@susyanomalysurface], theta];
erg2 = CollectTensors@FToGradA@JoinGammaMatrices@GammaMatricesToDual@
    IrreducibleSpinDecompose[erg, chi];
erg3 = mycanon@GradAToF@JoinGammaMatrices[erg2]
```

$$- A^{\alpha a}\epsilon g_{\alpha\mu\nu\rho}F_{\beta}{}^{\mu b}F^{\nu\rho c}d[lie]_{abc}\mathrm{Irr}\mathbb{S}M^{(2,\mathbf{4})}[\theta]^{\beta} - igA^{\alpha a}A^{\beta b}A^{\mu c}\epsilon g_{\alpha\beta\mu\rho}F_{\nu}{}^{\rho d}tr[lie]_{abcd}\mathrm{Irr}\mathbb{S}M^{(2,\mathbf{4})}[\theta]^{\nu}.$$

The result is not yet equal to the gauge anomaly $\mathcal{A}^{\mathrm{gauge}}$ with the ghost c^a replaced by $-\bar{\theta}\gamma^{\mu}\theta A_{\mu}^a$ [45], which is due to dimension-dependent identities that arise from antisymmetrising an arbitrary expression over five or more indices (which identically vanishes in four dimensions). A convenient way to use these identities is the projection of the corresponding expression onto the Young tableau describing the symmetry, where the dimension-dependent identities correspond to the fact that the projectors for Young tableaux with five or more rows vanish, and which for the ϵ tensor is implemented using `EpsilonYoungProject`:

```
sn = erg3 - mycanon[mycanon@GradAToF@gaugeanomaly /. {cc[a_] :> With[{μ =
    DummyIn@TangentM}, -IrreducibleSpinTensor[theta, 2, "4"][μ] AA[-μ, a]]}];
CollectTensors[sn /. {epsilongg[i__] IrreducibleSpinTensor[theta, 2, "4"][j_]
    :> EpsilonYoungProject[epsilongg[i] IrreducibleSpinTensor[theta, 2, "4"][j],
    gg]}]
```

0.

Since the consistency conditions are nothing else but the statement that the anomaly is BRST-exact, this shows that $\mathsf{s}(\mathcal{A}^{\mathrm{gauge}} + \mathcal{A}^{\mathrm{susy}}) = 0$, which of course can also be checked directly with FIELDSX.

In general, to bring expressions into a fully canonical form, one would in principle have to consider the interplay between permutations of the Lie algebra indices and the Lorentz indices (i.e., the composition of the multiterm symmetries described by the associated Young tableaux), a problem related to plethysms in the representation theory of the symmetric group. This is a hard problem in general, but the corresponding results can be emulated by repeatedly applying `EpsilonYoungProject` and bringing the result into canonical form using only monoterm symmetries (what xACT's `ToCanonical` command achieves).[8] In the present case, a single use of `EpsilonYoungProject` was fortunately sufficient, since the expression only contained five different indices, one of which belonged to a tensor that only appeared once in the expression, namely the irreducible spin tensor $\mathrm{Irr}\mathbb{S}M^{(2,\mathbf{4})}[\theta]_{\mu}$.

Lastly, I want to show how one can also derive the gauge anomaly directly in FIELDSX, at least in the $A - c$ sector where no canonicalisation of spinors is needed. The reason is that the algorithm implemented in FIELDSX is a brute-force one, which simply applies the BRST differential and checks whether the result vanishes. Therefore, the result will only be complete (and thus correct) if a fully canonical form can be found for each term,

---

[8]I thank Igor Khavkine for discussions on this subject.

which is quite involved for spinors. We thus have to make an ansatz for all possible terms that can possibly appear, which is restricted by ghost number and engineering dimension. Furthermore, we do not need to include traces of invariant tensors of the Lie algebra, since they can be expressed using invariant tensors of lower rank [46, 47], and need to include at most one totally antisymmetric ϵ tensor because the product of two can be reduced to products of metrics. We also need ansätze for surface terms and BRST-exact terms (and the corresponding surface terms), and define ghost number and dimension of the invariant tensors and ϵ to ensure this is correctly taken into account:

```
invtensors = {lief, lied, Invlie4, Invlie5};
Map[SetGrading[#, {Dimension → 0, GhostNumber → 0}] &, invtensors];
SetGrading[epsilongg, {Dimension → 0, GhostNumber → 0}];
SetGrading[CD, {Dimension → 1, GhostNumber → 0}];
tracelessreps = {t_?xTensorQ[i___, a_, j___, -a_, k___] :> 0};
ansatz = GenerateMonomials[{AA, cc}, {invtensors, {epsilongg}}, Constraint →
    (((Dimension[#]==5) && (GhostNumber[#]==1))&), Replacements →
    tracelessreps, MaxNumberOfFields → 5, MaxNumberOfInvTensors → 1,
    MaxNumberOfDerivatives → 3];
dansatz = GenerateMonomials[{AA, cc}, {invtensors, {epsilongg}}, Constraint →
    (((Dimension[#]==5) && (GhostNumber[#]==2))&), Replacements →
    tracelessreps, MaxNumberOfFields → 5, MaxNumberOfInvTensors → 1,
    MaxNumberOfDerivatives → 3, FreeIndices → {μ}];
csansatz = GenerateMonomials[{AA, cc}, {invtensors, {epsilongg}}, Constraint →
    (((Dimension[#]==4) && (GhostNumber[#]==0))&), Replacements →
    tracelessreps, MaxNumberOfFields → 4, MaxNumberOfInvTensors → 1,
    MaxNumberOfDerivatives → 2];
cdansatz = GenerateMonomials[{AA, cc}, {invtensors, {epsilongg}}, Constraint →
    (((Dimension[#]==4) && (GhostNumber[#]==1))&), Replacements →
    tracelessreps, MaxNumberOfFields → 4, MaxNumberOfInvTensors → 1,
    MaxNumberOfDerivatives → 2, FreeIndices → {μ}];
```

Since both A and c have engineering dimension 1, at most five (respectively four) fields are needed, and at most three (respectively two) derivatives, since any term with more derivatives would be a total derivative and not contribute anyway to the cohomology. For a more effective generation of monomials, the function `GenerateMonomialsByGrading` could be used as well. The computation of the relative cohomology then results in

```
RelativeCohomologyFromAnsatz[BRSTgauge, ansatz, CD[-μ], dansatz, BRSTgauge,
    csansatz, CD[-μ], cdansatz, CanonicalizeMethod → mycanon]
```

$$\{\{c^a \epsilon g_{\alpha\beta\gamma\delta} d[lie]_{abc} \nabla^\beta A^{\alpha b} \nabla^\delta A^{\gamma c} + ig A^{\alpha a} A^{\beta b} c^c \epsilon g_{\alpha\beta\gamma\delta} tr[lie]_{abcd} \nabla^\delta A^{\gamma d}$$
$$+ ig A^{\alpha a} A^{\beta b} c^c \epsilon g_{\alpha\beta\gamma\delta} tr[lie]_{abdc} \nabla^\delta A^{\gamma d} + ig A^{\alpha a} A^{\beta b} c^c \epsilon g_{\alpha\beta\gamma\delta} tr[lie]_{acbd} \nabla^\delta A^{\gamma d},$$
$$ig^2 A^{\alpha a} A^{\beta b} A^{\gamma c} c_d \cdot c_e \epsilon g^\mu{}_{\alpha\beta\gamma} tr[lie]_{abc}{}^{de} - ig A^{\alpha a} c_c \cdot c_d \epsilon g^\mu{}_{\alpha\beta\gamma} tr[lie]_{ab}{}^{cd} \nabla^\gamma A^{\beta b}$$
$$- ig A^{\alpha a} c_c \cdot c_d \epsilon g^\mu{}_{\alpha\beta\gamma} tr[lie]_a{}^c{}_b{}^d \nabla^\gamma A^{\beta b} - ig A^{\alpha a} c_c \cdot c_d \epsilon g^\mu{}_{\alpha\beta\gamma} tr[lie]_a{}^{cd}{}_b \nabla^\gamma A^{\beta b}\}\},$$

which agrees with $\mathcal{A}^{\text{gauge}}$. Denoting the first term by A and the second term in the result

by S^μ, S^μ is the needed surface term such that $\mathsf{s}A + \nabla_\mu S^\mu = 0$:

```
mycanon[erg[[1, 1]] - gaugeanomaly]
mycanon[BRSTgauge[erg[[1, 1]]] + CD[-μ]@erg[[1, 2]]]
```

0

0 .

4   Functions in FieldsX

In this section, I give a complete list of all functions and variables contained in FieldsX. In most cases the functions are self-explanatory, or their usage has already been demonstrated in the previous section. Other functions are only needed to implement advanced functionality, and will not be used by a normal user of the package. In the remaining cases, I give short descriptions and/or the rationale for the given implementation.

4.1   Helper functions

These are small helper functions that are needed for other parts of the package, but are also useful on their own.

4.1.1   DummiesIn

DummiesIn[*bundle*,*k*]
    returns a list of *k* unique abstract dollar-indices on the vector bundle *bundle*, using the last of the user-defined indices.

This is just a straightforward extension of the xAct command DummyIn.

4.1.2   TensorCount

TensorCount[*expr*,*T*]
    returns the number of tensors T occuring in *expr* including covariant derivatives of T. *expr* can be given in pseudo index-free notation.

TensorCount[*expr*,*T*,False,False]
    does not include covariant derivatives of T not its perturbations.
TensorCount[*expr*,*T*,False,True]
    does not include covariant derivatives of T but includes perturbations.
TensorCount[*expr*,*T*,True,True]
    includes both covariant derivatives of T and perturbations.

The pseudo index-free notation was introduced in the xTras package, where only heads of tensors are given and not their indices. Expressions in pseudo index-free notation must be wrapped in the head IndexFree. For example, IndexFree[CD@RicciCD] stands for CD[-μ][RicciCD[-α,-β].

4.1.3 `AllTensors`

`AllTensors[`*expr*`]`
    returns a list containing all tensors occuring in *expr* together with their multi-
    plicity. *expr* must be given in pseudo index-free notation.

To convert an expression with indices to pseudo index-free notation, use the command
`ToIndexFree` of the xTras package.

4.2 Inner bundles (Lie algebra functions)

These are functions to work with inner bundles in general, and Lie-algebra valued fields in
particular, including invariant tensors.

4.2.1 `DefVBundleWithMetric`

`DefVBundleWithMetric[`*bundle*`,`*M*`,`*dim*`,{`*a*`,`*b*`,`*c*`,...},`*metric*`]`
    defines *bundle* to be a vector bundle with base manifold M and fiber vector
    space with dimension given by *dim* and represented by the abstract indices
    $\{a, b, c, \dots\}$. It defines *metric* to be a metric on the vector bundle *bundle* with-
    out associated covariant derivative, which is constant with respect to the co-
    variant derivative on the tangent bundle of M.

By default, it is not possible to define metrics on inner vector bundles with xAct;
this command is adapted from code of the xAct mailing list and the Spinors package
where such a definition was already needed. The dimension *dim* can be an integer or
any constant symbol (defined with `DefConstantSymbol`). Since the metric is defined to be
constant with respect to the covariant derivative of the tangent bundle of the manifold, a
covariant derivative must be defined on M first. The main application of this command is for
principal G-bundles with G a semisimple Lie group or products thereof, since gauge fields
transform in some representation of the associated Lie algebra $\mathfrak{g}$ (in a local trivialisation
of the bundle). The metric is then just given by the Cartan–Killing form of the Lie algebra
$\mathfrak{g}$, and *dim* is the dimension of the chosen representation.

4.2.2 `$InvariantTraceTensors`

`$InvariantTraceTensors`
    is a global variable storing the list of all currently defined invariant tensors on
    inner bundles.

This is the analogue of the lists of metrics, etc.

4.2.3 `InvariantTraceTensor`

`InvariantTraceTensor[`*bundle*`,`*n*`,`*sym*`]`
    returns the invariant tensor on the inner bundle *bundle* obtained as the trace
    over n basis elements. For $n = 3$ *sym* determines whether the antisymmetric f
    tensor or the symmetric d tensor is returned.

By the Chevalley restriction theorem, the *G*-invariant polynomials for a simple Lie algebra $\mathfrak{g}$ are linear combinations of products of traces of the generators $\mathfrak{t}^a$ in some representation of $\mathfrak{g}$. This function returns $\mathrm{tr}(\mathfrak{t}^{a_1} \cdots \mathfrak{t}^{a_n})$ (only the head of the tensor, without the abstract indices). For $n = 2$, the invariant tensor is the Cartan–Killing form (the metric on *bundle*). For $n = 3$, specifying $sym \to$ `Antisymmetric` returns the completely antisymmetric f^{abc} (equal to the structure constants of $\mathfrak{g}$), while using $sym \to$ `Symmetric` returns the completely symmetric part d^{abc}. In the general case $n \geq 4$ it would in principle be possible to further decompose the traces according to their transformations under the permutation group, but this seems not be useful in applications such that only the cyclic symmetry of the trace is imposed. Note that for any given Lie algebra, invariant tensors of high enough rank are not independent and expressable in terms of tensors of lower rank or vanish. For example, $d^{abc} = 0$ for $\mathfrak{su}(2)$; see [46, 47] for derivations of such relations. These relations are not taken into account by FIELDSX and must be imposed by the user, giving rules for the corresponding tensors.

4.2.4 `InvariantTraceTensorQ`

`InvariantTraceTensorQ[`*expr*`]`
gives `True` if *expr* is an invariant tensor on some inner bundle, and `False` otherwise.

This is the analogue of the corresponding functions for metrics, etc.

4.2.5 `$StructureConstantSign`

`$StructureConstantSign`
defines the global sign of the structure constants of inner bundles: $[\mathfrak{t}_a, \mathfrak{t}_b] =$ `$StructureConstantSign` $f_{ab}{}^c \mathfrak{t}_c$. The default value is i.

By default, FIELDSX assumes Hermitian generators $\mathfrak{t}$ and structure constants.

4.2.6 `ReduceInvariantTraceTensors`

`ReduceInvariantTraceTensors[`*expr*`]`
expands products of the structure constants $f_{ab}{}^c$ with invariant tensors on inner bundles into sums of invariant tensors.

`ReduceInvariantTraceTensors[`*expr*`,`*tens*`]`
expands products of $f_{ab}{}^c$ with the invariant tensor *tens* only.

`ReduceInvariantTraceTensors[`*expr*`,{`*tens1*`,`*tens2*`,...}]`
expands products of $f_{ab}{}^c$ with the invariant tensors *tens1*, *tens2*, ...

Since gauge transformations involve the structure constants, this function can be used to put the result back into canonical form.

4.3 Functions extended to work with more than one bundle

These functions are needed for other parts of the package, and extend existing functionality of XACT.

4.3.1 `BundleSymmetryOf`

> `BundleSymmetryOf[`*expr*`]`
>
> gives a description (a result with head `Symmetry`) of the symmetry of *expr*. This includes a generating set for that symmetry using `Cycles` notation on the indices of *expr*. Extending `SymmetryOf`, the number of indices and the symmetry group are ordered by bundle.
>
> `Sorted`
>
> is an option for `BundleSymmetryOf` that specifies if the replacements should be sorted by slot number. By default, it is `True`.
>
> `Offset`
>
> is also an option for `BundleSymmetryOf` that specifies if the generating sets should use offsets for the slot numbers. By default, it is `False`.

This is needed for computing cohomologies.

4.4 Noncommuting product, Grassmann-even and -odd tensors

Here functions to work with non-commuting objects are listed.

4.4.1 `CenterDot`

> `CenterDot`
>
> stands for the non-commutative product of Grassmann-odd indexed objects.

`CenterDot` ($\cdot$) is an operator in MATHEMATICA with no built-in rules, which makes it ideal to use as the non-commutative product, in contrast to `NonCommutativeMultiply` ($**$). Commuting objects are automatically taken out of the non-commutative product, so it is safe to use `CenterDot` for all objects. `CenterDot` can be entered as $\boxed{\text{Esc}} + \boxed{\text{ . }} + \boxed{\text{Esc}}$. Since $**$ is nevertheless faster to input, to automatically convert `NonCommutativeMultiply` to `CenterDot` one can use the definition

```
Unprotect[NonCommutativeMultiply];
NonCommutativeMultiply /: NonCommutativeMultiply[expr___] := CenterDot[expr];
Protect[NonCommutativeMultiply];
```

Because of the automatic rules set up for `CenterDot`, in replacement rules it is necessary to use the `Verbatim` or `HoldPattern` wrappers: `Verbatim[CenterDot][expr___] :> ...` or `HoldPattern[ expr1_ · expr2_ ] :> ...`

4.4.2 `Parity`

> `Parity[`*expr*`]`
>
> returns the Grassmann parity of *expr*.

This gives either 0 or 1.

4.4.3 `TimesToCenterDot`

`TimesToCenterDot[`*expr*`]`
    returns *expr* with all products replaced by non-commutative ones.

This is useful when converting from pseudo index-free notation when non-commutative objects are included, since `IndexFree` does not work well with the non-commutative product.

4.4.4 `DefEvenTensor`

`DefEvenTensor[`*T*`[-`*a*`,`*b*`,...],`*M*`]`
    defines *T* to be a Grassmann-even tensor field on the manifold *M* and the base manifolds associated to the vector bundles of its indices $-a, b, \dots$

`DefEvenTensor[`*T*`[-`*a*`,`*b*`,...],`*M*`,`*sym*`]`
    defines *T* to be a Grassmann-even tensor field with symmetry *sym*.

This extends the xAct command `DefTensor` for commuting objects, and takes the same additional arguments (for example, `PrintAs`).

4.4.5 `DefOddTensor`

`DefOddTensor[`*T*`[-`*a*`,`*b*`,...],`*M*`]`
    defines *T* to be a Grassmann-odd tensor field on the manifold *M* and the base manifolds associated to the vector bundles of its indices $-a, b, \dots$

`DefOddTensor[`*T*`[-`*a*`,`*b*`,...],`*M*`,`*sym*`]`
    defines *T* to be a Grassmann-odd tensor field with symmetry *sym*.

This extends the xAct command `DefTensor` for non-commuting objects, and takes the same additional arguments (for example, `PrintAs`).

4.4.6 `$CenterDotTexSymbol`

`$CenterDotTexSymbol`
    gives the symbol to use for TEX output of the non-commutative product. The default value is a space.

FieldsX includes proper TEX output that works together with the TexAct package and can be customised using its commands.

4.5 Frame bundles and spin connections

These are functions to work with frame bundles on the manifold.

4.5.1 `FrameBundleQ`

`FrameBundleQ[`*bundle*`]`
    gives `True` if *bundle* is a frame bundle, and `False` otherwise.

This is the analogue of the corresponding functions for metrics, etc.

4.5.2 `DefFrameBundle`

> `DefFrameBundle[`*frame*`[-`μ`,`*a*`],`*eta*`[-`*a*`,`*-b*`],{`*a*`,`*b*`,...}]`
> defines a frame bundle *FrameM* with abstract indices $a, b, \ldots$ on the base manifold M of the tangent bunde represented by the abstract index μ. It also defines the flat frame bundle metric $eta[-a, -b]$ and the frame field $frame[-\mu, a]$ with inverse $frame[\mu, -a]$.

Before using this command, a metric (with its associated covariant derivative) must be defined on M. The frame field, its determinant *Detframe* and the frame bundle metric are defined to be constant with respect to this covariant derivative.

4.5.3 `UndefFrameBundle`

> `UndefFrameBundle[`*frame*`]`
> undefines the frame bundle whose frame field is *frame*.

If there are spin connections defined on the bundle, they must be undefined first.

4.5.4 `FrameFieldOfBundle`

> `FrameFieldOfBundle[`*tb*`,`*fb*`]`
> returns the frame field connecting the tangent bundle *tb* and the frame bundle *fb*.

This is needed for other parts of the package.

4.5.5 `$SpinConnections`

> `$SpinConnections`
> is a global variable storing the list of all currently defined spin connections.

This is the analogue of the lists of metrics, etc.

4.5.6 `DefSpinConnection`

> `DefSpinConnection[`ω`[-`μ`, -`*a*`, -`*b*`],`*CD*`]`
> defines the spin connection $\omega[\text{-}\mu,\text{-}a,\text{-}b]$ of the covariant derivative *CD* and the associated curvature and torsion tensors.

If the covariant derivative is the standard Levi-Civita one, the corresponding torsion and contortion vanishes. The torsion and contortion defined by FIELDSX for the spin connection agree with the conventions of [15], which differ in the ordering of the indices from the convention used by xAct.

4.5.7 `UndefSpinConnection`

> `UndefSpinConnection[`ω`]`
> undefines the spin connection ω.

This also removes the associated curvature and torsion tensors.

4.5.8 `CovDOfSpinConnection`

> `CovDOfSpinConnection[`ω`]`
>
>   returns the covariant derivative corresponding to the spin connection ω.

This is needed for other parts of the package.

4.5.9 `SpinConnectionToFrame`

> `SpinConnectionToFrame[`*expr*`,`ω`]`
>
>   expands all occurrences of the spin connection ω inside *expr* in terms of derivatives of the frame field and the torsion. It also expands perturbations of the spin connection in terms of covariant derivatives of the perturbation of the frame field and the torsion. The second argument can be a list of spin connections.

This is useful in a second-order formalism, where one only needs to specify the perturbation of the frame field.

4.5.10 `ContortionToTorsion`

> `ContortionToTorsion[`*expr*`,`*CD*`]`
>
>   expresses the contortion tensor *ContortionCD* of the covariant derivative *CD* occurring inside *expr* by the torsion tensor *TorsionCD*.

> `ContortionToTorsion[`*expr*`,`ω`]`
>
>   expresses the contortion tensor *Contortionω* of the spin connection ω occurring inside *expr* by the torsion tensor *Torsionω*.

>   In both cases, the second argument can be a list of covariant derivatives or spin connections.

The contortion defined by FIELDSX for a spin connection agrees with the conventions of [15], while the one defined for the covariant derivative differs in the ordering of the indices to match the torsion convention used by xACT.

4.6 Spin structure, γ matrices

These are functions to define a spin structure on the manifold and work with the corresponding curved-space γ matrices.

4.6.1 `GammaMatrixQ, GammaStarQ, GammaZeroQ`

> `GammaMatrixQ[`*expr*`]`
>
>   gives `True` if *expr* is a γ matrix, and `False` otherwise.

> `GammaStarQ[`*expr*`]`
>
>   gives `True` if *expr* is the γ_* (chiral) matrix, and `False` otherwise.

> `GammaZeroQ[`*expr*`]`
>
>   gives `True` if *expr* is the γ^0 matrix, and `False` otherwise.

This is the analogue of the corresponding functions for metrics, etc. The chiral γ_* matrix is only defined for even dimensions of the manifold, and γ^0 exists only in Lorentzian signature; both of them are fixed (numerical) matrices.

4.6.2 `$GammaStarSign`

`$GammaStarSign`
 defines the global sign of the γ_* (chiral) matrix. By default it is 1.

By default, FIELDSX assumes the conventions of [15], such that

$$\gamma_* = \frac{(-\mathrm{i})^{d/2+3}}{d!}\epsilon_{\mu_1\cdots\mu_d}\gamma^{\mu_1}\cdots\gamma^{\mu_d}$$

in even dimensions d and $\gamma_*^2 = \mathbb{1}$. `$GammaStarSign` can be used to insert an additional overall factor. In odd dimensions d, the $(d-1)$-dimensional chiral matrix is used as the last basis element of the Clifford algebra, and `$GammaStarSign` is the sign in eq. (3.40) of [15].

4.6.3 `GammaMatrix`

`GammaMatrix[`*metric*`,`*n*`]`
 returns the generalized (totally antisymmetric) γ matrix of order n of the Clifford algebra associated to the metric *metric*.

`GammaMatrix[`*metric*`,`Star`]`
 returns the γ_* (chiral) matrix of the Clifford algebra associated to the metric *metric*.

`GammaMatrix[`*metric*`,`Zero`]`
 returns the γ^0 matrix of the Clifford algebra associated to the metric *metric*.

The generalised γ matrices are normalised with unit strength: $\gamma^{\mu_1\cdots\mu_n} = \gamma^{[\mu_1}\cdots\gamma^{\mu_n]}$. The chiral γ_* matrix is only defined for even dimensions of the manifold, and γ^0 exists only in Lorentzian signature. These functions only return the head of the tensor, without the abstract (spin bundle) indices.

4.6.4 `MetricOfGammaMatrix`

`MetricOfGammaMatrix[`γ`]`
 returns the metric associated to the Clifford algebra of γ.

This is needed for other parts of the package.

4.6.5 `$GammaMatrices`

`$GammaMatrices`
 is a global variable storing the list of all currently defined γ matrices.

This is the analogue of the lists of metrics, etc.

4.6.6   `$PrecomputeGammaMatrixProducts`

`$PrecomputeGammaMatrixProducts`
    controls whether products of generalised γ matrices are precomputed when a spin structure is defined, to speed up later computations. By default, it is `True`.

In low dimensions, the precomputation only takes a few seconds, but in higher dimensions it might be preferable to switch it off.

4.6.7   `DefSpinStructure`

`DefSpinStructure[`*metric*`,{A,B,C,...}]`
    defines a spin structure on the base manifold M of the metric *metric*. This includes the γ matrices of the Clifford algebra associated to *metric* and a spin bundle *SpinM* with abstract indices $\{A, B, C, \dots\}$, whose covariant derivative is induced from the one of *metric*, with the same name.

Before using this command, a metric and (Levi-Civita) covariant derivative must be defined on M, whose dimension must be a positive integer. The curvature $(\mathcal{R}_{\mu\nu})_A{}^B$ of the induced spin connection is given in terms of the Riemann tensor $R_{\mu\nu\rho\sigma}$ of the metric by the well-known formula

$$(\mathcal{R}_{\mu\nu})_A{}^B = \frac{1}{4} R_{\mu\nu\rho\sigma} (\gamma^{\rho\sigma})_A{}^B. \tag{4.1}$$

If `$PrecomputeGammaMatrixProducts` is `True`, `DefSpinStructure` also precomputes products of generalised γ matrices to speed up later computations. If a frame bundle has been defined, FIELDSX also defines the corresponding frame γ matrices.

4.6.8   `UndefSpinStructure`

`UndefSpinStructure[`*metric*`]`
    undefines the spin structure on the base manifold of the metric *metric*.

This also removes all related definitions of γ matrices etc.

4.6.9   `SpinBundleQ`

`SpinBundleQ[`*bundle*`]`
    gives `True` if *bundle* is a spin bundle, and `False` otherwise.

This is the analogue of the corresponding functions for metrics, etc.

4.6.10   `SplitGammaMatrix, SplitGammaMatrices`

`SplitGammaMatrix[`γ`,`*keep*`]`
    decomposes the generalized γ matrix γ into an antisymmetrized product of individual γ matrices. If *keep* $=$ `True`, the γ_* (chiral) matrix is kept.

`SplitGammaMatrices[`*expr*`,`*keep*`]`
    decomposes all the generalized γ matrix appearing within *expr* into an antisymmetrized product of individual γ matrices. If *keep* $=$ `True`, the γ_* (chiral) matrix is kept.

For example, this is needed for computing perturbations of curved-space γ matrices from the perturbation of the individual matrices.

4.6.11 `JoinGammaMatrices`

`JoinGammaMatrices[`*expr*`,`*keep*`]`
    replaces products of γ matrices within *expr* by generalized γ matrices. If *keep* = `True`, products of a single γ_* (chiral) matrix and other generalized γ matrices are kept.

Since supersymmetry transformations involve generalised γ matrices, this function can be used to put the result back into canonical form. `JoinGammaMatrices` always simplifies expressions with products of generalised γ matrices and at least two chiral γ_* matrices, but keeps products of generalised γ matrices and a single γ_* if the second parameter is `True`.

The formula for the product of two generalised γ matrices is [**?** , eq. (3.41)]

$$\gamma_{a_1\cdots a_i}\gamma^{b_1\cdots b_j} = \sum_{s=0}^{\min(i,j)} \frac{i!j!}{s!(i-s)!(j-s)!}\delta^{[b_1}_{[a_i}\cdots\delta^{b_s}_{a_{i-s+1}}\gamma_{a_1\cdots a_{i-s}]}{}^{b_{s+1}\cdots b_j]}\,.$$

4.6.12 `DualGammaMatrix`

`DualGammaMatrix[`γ`[`μ`, ..., -A, B]]`
    returns the dual matrix of the generalized γ matrix $\gamma^{\mu\cdots}{}_A{}^B$.

The duality relations are the ones given in [15], eqs. (3.41) and (3.42). In odd dimensions, `$GammaStarSign` gives the sign used in eq. (3.41).

4.6.13 `GammaMatricesToDual`

`GammaMatricesToDual[`*expr*`]`
    replaces generalized γ matrices with more than $\frac{d}{2}$ indices within *expr* by their dual, where d is the dimension of the manifold.

`GammaMatricesToDual[`*expr*`,`All`]`
    replaces all generalized γ matrices within *expr* by their dual.

`GammaMatricesToDual[`*expr*`,`γ`]`
    replaces only the matrix γ within *expr* by its dual.

The dual γ matrices are given by `DualGammaMatrix`. It is often useful to perform computations with γ matrices with less indices, since canonicalisation is faster.

4.6.14 `ChangeGammaMatrices`

`ChangeGammaMatrices[`*expr*`,`*met1*`,`*met2*`]`
    converts all generalized γ matrices within *expr* depending on the metric *met1* to the corresponding ones depending on the metric *met2*. One of the metrics must be the metric of a tangent bundle, and the other one of the associated frame bundle.

To convert between curved-space γ matrices and the ones in a frame, a frame bundle must be defined using `DefFrameBundle`.

4.6.15   `EpsilonGammaReduce, EpsilonYoungProject`

> `EpsilonGammaReduce[`*expr*`,`*metric*`]`
>   replaces products of the totally antisymmetric ϵ tensor and generalized γ matrices associated to the metric *metric* within *expr* by suitably contracted ones.

> `EpsilonYoungProject[`*expr*`,`*metric*`]`
>   replaces products of the totally antisymmetric ϵ tensor associated to the metric *metric* and other tensors within *expr* onto the corresponding Young tableaux.

This can also be used to put expressions into canonical form. The list of replacements is given in appendix B.

4.7   Spinors

These are functions to define spinors of various types.

4.7.1   `MajoranaQ, DiracQ, SpinorQ, SpinorUnbarQ, SpinorBarQ`

> `MajoranaQ[`*expr*`]`
>   gives `True` if *expr* is a Majorana spinor, and `False` otherwise.

> `DiracQ[`*expr*`]`
>   gives `True` if *expr* is a Dirac spinor, and `False` otherwise.

> `SpinorQ[`*expr*`]`
>   gives `True` if *expr* is a spinor, and `False` otherwise.

> `SpinorUnbarQ[`*expr*`]`
>   gives `True` if *expr* is a spinor but not a conjugate one, and `False` otherwise.

> `SpinorBarQ[`*expr*`]`
>   gives `True` if *expr* is a conjugate spinor, and `False` otherwise.

This is the analogue of the corresponding functions for metrics, etc. FIELDSX does not (yet) support neither 2- nor 4-component Weyl spinors. Since 4-component Weyl spinors can be viewed as Dirac spinors ψ_A that satisfy

$$\psi_a = (\mathcal{P}_\pm \psi)_A = \frac{1}{2}(\mathbb{1} \pm \gamma_*)_A{}^B \psi_B = \frac{1}{2}\psi_A \pm \frac{1}{2}(\gamma_*)_A{}^B \psi_B \,,$$

they can be emulated by defining rules that replace contractions of γ_* with ψ.

4.7.2 `DefSpinor`

> `DefSpinor[`ψ`[-`A`],`*M*`]`
>> defines ψ to be a spinor field on the manifold M and the spin bundle associated to the index $-A$. The conjugate spinor $\bar{\psi}$ is automatically defined, with name $bar\psi$.
>
> `DefSpinor[`ψ`[-`A`],`*M*`,`*sym*`]`
>> defines ψ to be a spinor field with symmetry *sym*.
>
> `DefSpinor[`ψ`[`b`,-`A`],`*M*`]`
>> defines ψ to be a spinor field valued in the inner bundle associated to the index b.
>
> `SpinorType`
>> is an option for `DefSpinor` that specifies the type of spinor. By default, it is `Majorana`.
>
> `Majorana`
>> is a value for the option `SpinorType` of `DefSpinor`.
>
> `Dirac`
>> is a value for the option `SpinorType` of `DefSpinor`.
>
> `Conjugate`
>> is also an option for `DefSpinor` that specifies if the roles of spinor and conjugate spinor should be switched.

`DefSpinor` is the general function (extending `DefTensor` from xAct, and taking thus the same additional arguments such as `PrintAs`), but for applications it is probably more useful to use `DefEvenSpinor` and `DefOddSpinor` to define spinors with given Grassmann parity. The `Conjugate` option is useful in the BV formalism, where the antifield $\psi^{\dagger}$ for a spinor ψ is really a conjugate spinor, while the antifield $\bar{\psi}^{\dagger}$ of the conjugate $\bar{\psi}$ is not a conjugate one, such that their respective products are scalars.

4.7.3 `UndefSpinor`

> `UndefSpinor[`ψ`]`
>> undefines ψ and the conjugate spinor $\bar{\psi}$.

This ensures that both ψ and its conjugate are removed.

4.7.4 `DefEvenSpinor`

`DefEvenSpinor[`ψ`[-`A`],`*M*`]`
  defines ψ to be a Grassmann-even spinor field on the manifold M and the spin
  bundle associated to the index $-A$. The conjugate spinor $\bar{\psi}$ is automatically
  defined, with name *bar*ψ.

`DefEvenSpinor[`ψ`[-`A`],`*M*`,`*sym*`]`
  defines ψ to be a Grassmann-even spinor field with symmetry *sym*.

`DefEvenSpinor[`ψ`[`b`,-`A`],`*M*`]`
  defines ψ to be a Grassmann-even spinor field valued in the inner bundle asso-
  ciated to the index b.

This extends `DefSpinor`, such that the options given there (and the ones of the xAct
command `DefTensor`) are also valid for `DefEvenSpinor`. Lie-algebra valued spinors are de-
fined with the last variant; the inner bundle must be defined first with `DefVBundleWithMetric`.

4.7.5 `DefOddSpinor`

`DefOddSpinor[`ψ`[-`A`],`*M*`]`
  defines ψ to be a Grassmann-odd spinor field on the manifold M and the spin
  bundle associated to the index $-A$. The conjugate spinor $\bar{\psi}$ is automatically
  defined, with name *bar*ψ.

`DefOddSpinor[`ψ`[-`A`],`*M*`,`*sym*`]`
  defines ψ to be a Grassmann-odd spinor field with symmetry *sym*.

`DefOddSpinor[`ψ`[`b`,-`A`],`*M*`]`
  defines ψ to be a Grassmann-odd spinor field valued in the inner bundle asso-
  ciated to the index b.

This extends `DefSpinor`, such that the options given there (and the ones of the xAct
command `DefTensor`) are also valid for `DefOddSpinor`. Lie-algebra valued spinors are defined
with the last variant; the inner bundle must be defined first with `DefVBundleWithMetric`.

4.7.6 `ConjugateSpinor`

`ConjugateSpinor[`ψ`]`
  returns $\bar{\psi}$.

`ConjugateSpinor[`$\bar{\psi}$`]`
  returns ψ.

Note that the conjugate is the charge conjugate for Majorana spinors and the Dirac
conjugate $\bar{\psi} = i\psi^{\dagger}\gamma^0$ for Dirac spinors.

4.7.7 `SpinScalar`

> `SpinScalar[`*`expr`*`]`
>
> gives `True` if *expr* (given in pseudo index-free notation) is a spin bundle scalar (i.e., all indices with values in a spin bundle can be contracted), and `False` otherwise.

This is needed for computing cohomologies. The expression *expr* must be wrapped in the head `IndexFree` of the XTRAS package.

4.8 Flip and Fierz relations

These are functions for flip relations (valid for Majorana spinors) and Fierz identities and rearrangements (valid for all spinors).

4.8.1 `$SpinorFlipSigns`

> `$SpinorFlipSigns`
>
> is a table of signs appearing in the Majorana flip relations.

The signs are taken from [15], table 3.1 (the choices in boldface).

4.8.2 `SignOfGammaMatrix`

> `SignOfGammaMatrix[`γ`]`
>
> returns the sign needed for the generalized γ matrix γ appearing in the Majorana flip relations.

This is the sign appearing in the Majorana flip relation (3.51) of [15].

4.8.3 `FindSpinChain`

> `FindSpinChain[`*`expr`*`,`*`start`*`[`*`inds`*`]]`
>
> returns a list of spinors and γ matrices appearing within *expr* whose indices are contracted with each other (spin chain). The chain starts with *start*[*inds*], which must be a (conjugate) spinor appearing in *expr*.

This functions extracts a spinor bilinear which can be flipped using the Majorana flip relations. The functions `FlipSpinor`, `FlipSpinorsToConjugateAmount`, and `SpinorFlipSymmetrize` might be more useful for applications.

4.8.4 `FlipSpinChain`

> `FlipSpinChain[`*`expr`*`,`*`chain`*`]`
>
> returns *expr* with the spin chain *chain* flipped using the Majorana flip relations.

The spin chain *chain* can be found using `FindSpinChain`. The functions `FlipSpinor`, `FlipSpinorsToConjugateAmount`, and `SpinorFlipSymmetrize` might be more useful for applications.

4.8.5 `FlipSpinor`

`FlipSpinor[`*expr*`]`
   returns *expr* with a spinor bilinear flipped using the Majorana flip relations.
   expr must contain a single spinor bilinear.

`FlipSpinor[`*expr*`,`ψ`]`
   returns *expr* with a spinor bilinear flipped using the Majorana flip relations.
   expr must contain a single bilinear formed with the spinor ψ.

`FlipSpinor[`*expr*`,`ψ_1`,`ψ_2`]`
   returns *expr* with a spinor bilinear flipped using the Majorana flip relations.
   expr must contain a single bilinear formed with the spinors ψ_1 and ψ_2.

This function uses the Majorana flip relations on a single bilinear, with the variants controlling which bilinear is flipped.

4.8.6 `FlipSpinorsToConjugateAmount`

`FlipSpinorsToConjugateAmount[`*expr*`,`ψ`,`*count*`]`
   returns *expr* with spinor bilinears formed with the spinor ψ flipped using the Majorana flip relations until *count* conjugate spinors $\bar{\psi}$ remain.

This function is useful to ensure that all terms of a long expression have the same form, in order to exhibit cancellations.

4.8.7 `SortSpinor`

`SortSpinor[`*expr*`,`$\psi \to \bar{\psi}$`]`
   returns *expr* with all bilinears formed with the spinor ψ flipped using the Majorana flip relations. The spinor ψ and its conjugate $\bar{\psi}$ can be exchanged.

`SortSpinor[`*expr*`,{`$\psi_1 \to \bar{\psi}_1$`, ...}]`
   returns *expr* with all bilinears formed with the spinors $\psi_1, \dots$ flipped using the Majorana flip relations. The spinors ψ_i and their conjugates $\bar{\psi}_i$ can be exchanged.

Also this function is useful to bring all terms of a long expression into the same form.

4.8.8 `SpinorFlipSymmetrize`

`SpinorFlipSymmetrize[`*expr*`]`
   returns *expr* with all spinor bilinears symmetrized using the Majorana flip relations.

This function replaces all bilinears in *expr* with half of the sum of the bilinear and its flipped version.

4.8.9 `FierzExpand`

> `FierzExpand[`$\bar{\psi}_1[$*inds*$_1]$`,`$\psi_2[$*inds*$_2]$`]`
> expands the tensor product $\bar{\psi}_1[$*inds*$_1]\psi_2[$*inds*$_2]$ in the basis of generalized γ matrices (Fierz rearrangement). Both spinors may have covariant derivatives acting on them.

This is the basic Fierz rearrangement formula

$$\bar{\psi}^A \chi_B = 2^{-[d/2]} \sum_{k=0}^{d} \frac{1}{k!} (-1)^{k(k-1)/2} \gamma_{\mu_1 \cdots \mu_k B}{}^A \left(\bar{\psi}^C \gamma^{\mu_1 \cdots \mu_k}{}_C{}^D \chi_D \right),$$

which is valid in the given form for both commuting and anticommuting spinors. From this, various rearrangement formulas for three, four and more spinors can be obtained by contracting this equation with more γ matrices and spinors.

4.9 Irreducible spin tensors

These functions implement the approach to Fierz identities of d'Auria, Fré, Maina and Regge [16] based on the decomposition of the product of group representations into irreducible components. Currently, the corresponding decompositions are worked out for the tensor product of up to four Majorana spinors in four dimensions, which can have an additional inner bundle index.

4.9.1 `IrreducibleSpinTensor`

> `IrreducibleSpinTensor[`ψ`,`*n*`,`*rep*`]`
> returns the irreducible tensor of the Lorentz group corresponding to the representation *rep* appearing in the decomposition of the tensor product of *n* copies of the spinor ψ.

This function returns the invariant tensor $\Psi^{(n,\boldsymbol{rep})}$ (only the head of the tensor, without the abstract indices) that transforms in the irreducible representation *rep* (labeled by its dimension) of the Lorentz group (and the symmetric group for spinors with inner bundle indices) contained in the product of *n* spinors ψ. The list of valid representations is given in appendix A.

4.9.2 `IrreducibleSpinTensorQ`

> `IrreducibleSpinTensorQ[`*expr*`]`
> gives `True` if *expr* is an irreducible tensor of the Lorentz group for some representation, and `False` otherwise.

This is the analogue of the corresponding functions for metrics, etc.

4.9.3   `ExpandIrreducibleSpinTensor, ExpandIrreducibleSpinTensors`

`ExpandIrreducibleSpinTensor[`*tens*`[`ψ`,...][`*inds*`]]`
expands the irreducible tensor *tens* into a sum of products of spinors ψ with the right symmetries.

`ExpandIrreducibleSpinTensors[`*expr*`]`
expands all irreducible tensors of the Lorentz group within *expr* into a sum of products of spinors with the right symmetries.

These functions give the explicit form of the invariant tensors in terms of the spinor ψ and γ matrices. It is the inverse function to `IrreducibleSpinDecompose`.

4.9.4   `IrreducibleSpinDecompose`

`IrreducibleSpinDecompose[`*expr*`,`ψ`]`
decomposes all products of spinors ψ within *expr* into sums of irreducible representations of the Lorentz group.

This function performs the decomposition of products of the spinor ψ into invariant tensors transforming in irreducible representations of the Lorentz group (and the symmetric group for spinors with inner bundle indices). It is the inverse function to `ExpandIrreducibleSpinTensors`.

4.9.5   `IrreducibleSpinProject`

`IrreducibleSpinProject[`*expr*`,`ψ`]`
projects all irreducible tensors of the Lorentz group within *expr* depending on the spinor ψ onto their Young tableaux. This includes their products with γ matrices, and induced symmetries if the spinor ψ depends on inner bundles.

This function ensures that multiterm symmetries are correctly taken into account when bringing terms into canonical form, using Young projectors. The list of applied projections is given in appendix B.

4.10   Gradings

These are functions to work with arbitrary gradings, for example ghost number, antifield number or engineering dimension.

4.10.1   `$Gradings`

`$Gradings`
is a global variable storing the list of all currently defined gradings.

This is the analogue of the lists of metrics, etc.

4.10.2 `DefGrading`

`DefGrading[`*grad*`]`
    defines the grading *grad*.

`DefGrading[{`*grad*$_1$`,...}]`
    defines the gradings *grad*$_1$,...

`SumGrading`
    is an option for `DefGrading` that specifies a function that determines the grading
    of a sum. By default it is `Undefined&`.

`ZeroGrading`
    is an option for `DefGrading` that specifies the grading of 0. By default it is
    `Undefined`.

Numeric expressions are not taken into account when computing the grading of an expression, i.e. $grad[2\,expr] = grad[expr]$. The grading of a product is the sum of the gradings of the individual terms, and grading of a covariant derivative of an expression is the sum of the gradings of the expression and the derivative. Gradings work properly with the symmetrised covariant derivatives of the xTras package, where the grading of the derivative gets multiplied by the number of indices it contains. In some cases, it is meaningful to assign a definite grading to a sum or to 0, for example if all the terms in the sum have the same ghost number.

4.10.3 `UndefGrading`

`UndefGrading[`*grad*`]`
    undefines the grading *grad*.

This function does not remove gradings set for individual tensors.

4.10.4 `SetGrading`

`SetGrading[`*T*`,`*grad* $\to$ *val*`]`
    sets the grading *grad* of the tensor or spinor T to the value *val*.

`SetGrading[{`*T*$_1$`,`*T*$_2$`,...},`*grad* $\to$ *val*`]`
    sets the grading of all the tensors T_i.

`SetGrading[`*T*`,{`*grad*$_1$ $\to$ *val*$_1$`,`*grad*$_2$ $\to$ *val*$_2$`,...}]`
    sets all gradings *grad*$_i$ to their respective values *val*$_i$.

This function sets the grading of individual tensors or spinors, which must be defined first with `DefGrading`. Setting the grading for a spinor automatically sets the same grading for the conjugate spinor; if the spinor and its conjugate have different gradings the one of the spinor must be set first. Gradings can also be set for covariant derivatives, for example an engineering dimension, and for the symmetrised covariant derivatives of the xTras package get multiplied by the number of indices.

4.11 Left and right variational derivatives

For non-commuting objects, left and right variational derivatives can differ by signs, and the functions here handle non-commutative products correctly.

4.11.1 LeftVarD

LeftVarD[*T*[*inds*]][*expr*]
> returns the left variational derivative of *expr* with respect to the tensor or spinor field $T[inds]$.

LeftVarD[*T*[*inds*],*covd*][*expr*]
> returns the left variational derivative of *expr* with respect to the tensor or spinor field $T[inds]$. Integration by parts uses the covariant derivative *covd* instead of the partial derivative PD.

The left variational or functional derivative δ_{L} is defined such that the Leibniz (product) rule holds from the left:

$$\frac{\delta_{\mathrm{L}}}{\delta T(x)}(AB) = \frac{\delta_{\mathrm{L}}A}{\delta T(x)}B + (-1)^{\epsilon_T \epsilon_A}A\frac{\delta_{\mathrm{L}}B}{\delta T(x)}\,,$$

where ϵ_T is the Grassmann parity of *T*. *expr* must be a scalar (or scalar density) and is implicitly assumed to be integrated over, such that integration by parts can be performed, as for the VarD command of xAct. If a tensor T implicitly depends on another tensor U and the functional derivative of T^{it} with respect to U^{iu} is given by the tensor $R^{it,iu}$, this can be defined using

```
T /: ImplicitTensorDepQ[T, U] = True;
Unprotect[LeftVarD];
LeftVarD[U[iu__], cd_][T[it__], l_, r_] := CenterDot[l, R[it,iu], r];
Protect[LeftVarD];
```

For Majorana spinors where the spinor and its (charge) conjugate are not independent, the appropriate definitions are automatically set up by DefSpinor, DefEvenSpinor, and DefOddSpinor.

4.11.2 RightVarD

RightVarD[*T*[*inds*]][*expr*]
> returns the right variational derivative of *expr* with respect to the tensor or spinor field $T[inds]$.

RightVarD[*T*[*inds*],*covd*][*expr*]
> returns the right variational derivative of *expr* with respect to the tensor or spinor field $T[inds]$. Integration by parts uses the covariant derivative *covd* instead of the partial derivative PD.

The same comments as for the left variational derivative `LeftVarD` apply, except that for the right variational derivative δ_{R} the Leibniz rule holds from the right:

$$\frac{\delta_{\mathrm{R}}}{\delta T(x)}(AB) = A\frac{\delta_{\mathrm{R}}B}{\delta T(x)} + (-1)^{\epsilon_T \epsilon_B}\frac{\delta_{\mathrm{R}}A}{\delta T(x)}B \, .$$

4.12 Contractions and Monomials

These are functions to generate all possible terms with given tensor content or grading, extending functionality from the xTras package, and can be used to find actions and ansätze for the computation of cohomologies.

4.12.1 `FindAllContractions`

FindAllContractions[*expr*]

> returns a list of all possible full contractions of *expr* over its free indices. Extending `AllContractions`, this function also works if the tensors within *expr* depend on more than one bundle. *expr* can be given in pseudo index-free notation.

FindAllContractions[*expr*, {$a, b, \ldots$}]

> returns a list of all possible full contractions of *expr* that have $a, b, \ldots$ as free indices.

FindAllContractions[*expr*, {$a, b, \ldots$}, *sym*]

> returns a list of all possible full contractions of *expr* with the symmetry *sym* imposed on the free indices $a, b, \ldots$

SymmetrizeMethod

> is also an option for `FindAllContractions` that specifies a function to symmetrize the free indices. By default, it is `ImposeSymmetry`.

AuxiliaryTensor

> is also an option for `FindAllContractions` that specifies the name of the auxiliary tensor used for the free indices.

Parallelization

> is also an option for `FindAllContractions` that specifies whether contractions should be calculated in parallel.

Given a product of tensors (a monomial), this function finds all possible contractions of them with a specified set of free indices. `FindAllContractions` extends the corresponding functionality of `AllContractions` from the xTras package (using some of the internal xTras routines) to also allow spinors and tensor with indices in inner bundles (for example, Lie-algebra valued ones). The resulting list can be used with `MakeAnsatz` from the xTras package to determine a complete ansatz with undetermined constants. If one is not interested in a single monomial, but needs all monomials containing a given set of tensors, the functions `GenerateMonomialsByGrading` or `GenerateMonomialsByGrading` are more useful.

4.12.2 `GenerateMonomials`

`GenerateMonomials[`*fields,invtens*`]`

returns a list of all monomials that can be formed from the fields *fields*, their covariant derivatives, and the invariant tensors *invtens*.

`FreeIndices`

is an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies a list of free (uncontracted) indices that the returned monomials should have. By default, it is an empty list.

`Constraint`

is an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies the constraint function, a function returning `True` if its argument should be added to the list of monomials and `False` otherwise. By default, it is given by `True&` (i.e., no constraint).

`Replacements`

is an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies a list of replacements to be made after contracting free indices in each monomial. By default, it is an empty list.

`MaxNumberOfFields`

is an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies the maximum number of fields of each type that can appear in an monomial. It can be an integer, or a list of integers specifying the maximum number for each field. By default, it is `Infinity` for `GenerateMonomialsByGrading` and 5 for `GenerateMonomials`.

`MaxNumberOfInvTensors`

is an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies the maximum number of invariant tensors of each type that can appear in an monomial. It can be an integer, or a list of integers specifying the maximum number for each tensor. By default, it is `Infinity` for `GenerateMonomialsByGrading` and 1 for `GenerateMonomials`.

`MaxNumberOfDerivatives`

is an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies the maximum number of derivatives that can be applied to a field. It can be an integer, or a list of integers specifying the maximum number for each field. By default, it is `Infinity` for `GenerateMonomialsByGrading` and 3 for `GenerateMonomials`.

`IndexFree`

is also an option for `GenerateMonomials` and `GenerateMonomialsByGrading` that specifies if the list of monomials should returned in pseudo index-free notation. By default, it is `False`.

This function generates a list of all products of tensors (monomials) that can be obtained with a maximum number of a given tensor or spinor (`MaxNumberOfFields`), with a maximum number of derivatives acting on them (`MaxNumberOfDerivatives`) and a maximum number of invariant tensors (`MaxNumberOfInvTensors`), for example metrics, γ matrices, or the tensors returned from `InvariantTraceTensor`. For an efficient generation, the MULTI-SETS package is used. The resulting list is then passed to `FindAllContractions` to suitably contract free indices, except if the option `IndexFree` is given. The only difference between the *fields* and *invtens* arguments is that covariant derivatives are only generated on *fields*, but otherwise arbitrary tensors or spinors can be used for both. A constraint function is applied on each monomial before it is added to the list, and can be used to filter terms (for example, to impose a certain grading: engineering dimension, ghost number, . . .). In many cases, one needs a list of monomials with a fixed grading, and `GenerateMonomialsByGrading` can be more efficient.

4.12.3   `GenerateMonomialsByGrading`

`GenerateMonomialsByGrading[`*fields*`,`*invtens*`,`*grad* $\to$ *n*`]`
> returns a list of all monomials that can be formed from the fields *fields*, their covariant derivatives, and the invariant tensors *invtens*, restricted to the value *n* for the grading *grad*.

`FilterGammaMatrices`
> is an option for `GenerateMonomialsByGrading` that specifies whether spinor bilinears containing more than one generalized γ matrix should be dropped. By default, it is `True`.

All options of the `GenerateMonomials` command can also be used for `GenerateMonomials-ByGrading`. This function uses a more efficient algorithm to generate monomials than `GenerateMonomials`, based on integer partitions, and does not rely on the MULTISETS package. However, this leads to limitations (which are fulfilled in many appplications): the grading *grad* must be non-negative and rational for all tensors in *fields* and *invtens*. For fields with zero grading, a maximum number must be specified, as well as for derivatives whch have zero grading. Moreover, because of the way that fields with zero grading are incorporated, they cannot have derivatives acting on them. The `FilterGammaMatrices` option is used to filter terms of the form $\bar{\psi}\gamma^{\mu}\gamma^{\nu}\chi$, which can be expressed using $\gamma^{\mu\nu}$ and $g^{\mu\nu}$, and similar ones. (To obtain a complete ansatz, one of course needs to include all γ matrices.)

4.13   BRST operator and filtrations

These are functions to work with the BV/BRST formalism for gauge theories.

4.13.1   `DefOddDifferential`

`DefOddDifferential[`*brst*`]`
> defines a Grassmann-odd differential *brst* that commutes with covariant derivatives. The BRST differential BRST is predefined.

This function defines linear fermionic differentials, such as the BRST differential. It can be used for filtrations of the predefined `BRST` differential using the `Filtrate` command.

4.13.2 BRST

BRST[*expr*]

returns the BRST differential applied to *expr*. BRST transformations can be defined using F /: BRST[F[inds]] ^:= G[inds].

The BRST differential is set up as a linear fermionic operator commuting with covariant derivatives. However, nilpotency is not (and cannot be) enforced, but depends on the transformations defined by the user.

4.13.3 BRSTWeightInequalities

BRSTWeightInequalities[{*field*$_1$,*field*$_2$,...},*brst*,*weight*]

returns a list with inequalities that the weight function *weight* has to fulfill to be an admissible filtration for the BRST differential *brst* applied to the fields *field*$_1$, *field*$_2$, ...

BRSTWeightInequalities[{*field*$_1$,*field*$_2$,...},*brst*,*weight*,False]

does not apply the Reduce function to the obtained system of inequalities before returning it.

Filtrations, where each field is assigned a (non-negative) weight, can be used to simplify the computation of BRST cohomologies by first computing the cohomology at lowest weight [34]. To obtain a consistent filtration, the assignment of weights is restricted, and `BRSTWeightInequalities` gives a list of inequalities that the weight function *weight* has to fulfill. If some assignments of weights are already determined, this function can be used to obtain conditions on the remaining assignments.

4.13.4 FindBRSTWeights

FindBRSTWeights[{*field*$_1$,*field*$_2$,...},*brst*,*maxweight*]

returns a list of all weights admissible for filtrations of the BRST differential *brst* applied to the fields *field*$_1$, *field*$_2$, ..., with the maximum weight of each field restricted to be $\leq$ *maxweight*.

This function can be used to find all permissible weights, with *maxweight* a positive integer. `FindBRSTWeights` uses a brute-force algorithm, such that for large *maxweight*'s the computation may take quite long.

4.13.5 `CheckFiltration`

`CheckFiltration[{field_1, field_2, ...}, {w_1, w_2, ...}, brst]`
  displays a table of the lowest-order terms of the BRST differential *brst* applied
  to the fields $field_1, field_2, \ldots$ and filtrated according to the weights $w_1, w_2, \ldots$

`CheckFiltration[{field_1 → w_1, field_2 → w_2, ...}, brst]`
  displays a table of the lowest-order terms of the BRST differential *brst* applied
  to the fields $field_1, field_2, \ldots$ and filtrated according to the weights $w_1, w_2, \ldots$

`Display`
  is also an option for `CheckFiltration` with values `IndexFree` and `Full` that
  specifies how the table entries should be displayed.

This function allows to quickly check how the filtrated BRST differential (at lowest weight) looks like for a given assignment of weights.

4.13.6 `Filtrate`

`Filtrate[{field_1, field_2, ...}, {w_1, w_2, ...}, brst → brst_0]`
  defines rules for the operator $brst_0$ such that it acts as the lowest-order terms
  of the BRST differential *brst* applied to the fields $field_1, field_2, \ldots$ and filtrated
  according to the weights $w_1, w_2, \ldots$

`Filtrate[{field_1 → w_1, field_2 → w_2, ...}, brst → brst_0]`
  defines rules for the operator $brst_0$ such that it acts as the lowest-order terms
  of the BRST differential *brst* applied to the fields $field_1, field_2, \ldots$ and filtrated
  according to the weights $w_1, w_2, \ldots$

This function performs the actual filtration. To remove transformations that are already defined for the lowest-weight BRST differential $brst_0$, `RemoveFiltration` can be used.

4.13.7 `RemoveFiltration`

`RemoveFiltration[brst_0]`
  removes the rules defined for the differential $brst_0$ defined by Filtrate.

This function clears all user-defined rules for the differential $brst_0$.

4.14 Cohomology

4.14.1 `CohomologyFromAnsatz`

CohomologyFromAnsatz[*brst*,*ansatzcc*,*brstb*,*ansatzcb*]
returns a list of representatives of elements of the cohomology $\mathrm{Ker}(brst)/\mathrm{Im}(brstb)$. For the BRST cohomology H(s), one needs to take $brst = brstb = \mathtt{BRST}$. A list of possible elements (cocycles) must be given as *ansatzcc*, and a list of possible exact elements (coboundaries) as *ansatzcb*. Both lists could be calculated using GenerateMonomials or GenerateMonomialsByGrading.

CanonicalizeMethod
is an option for CohomologyFromAnsatz and RelativeCohomologyFromAnsatz that specifies the function applied to an expression after the differential has acted, to obtain a canonical form. By default, it is given by CollectTensors[ReduceInvariantTraceTensors[ContractMetric[SymmetrizeCov-Ds[Expand[#]]]]]&.

SimplifyMethod
is an option for CohomologyFromAnsatz and RelativeCohomologyFromAnsatz that specifies the function applied to representatives of elements of the cohomology before they are returned. By default, it is given by Identity (i.e., no transformation).

This function computes (representatives of) the (co-)homology of a pair of differentials, which can be elements of a (co-)chain complex. It uses a simple straightforward algorithm, evaluating the differential *brst* on each of the elements of the ansatz *ansatzcc* and checking whether the result can be written as a linear combination of the boundary differential *brstb* evaluated on the elements of the ansatz *ansatzcb*. Since this function is completely agnostic about the nature of the differentials *brst* and *brstb*, it can also compute homologies. The choice of CanonicalizeMethod is extremely important, since to check whether an element of the ansatz *ansatzcc* lies in the (co-)homology, it must be compared with zero (after application of the differential *brst*), and checking whether some expression is zero is in general a hard problem. CanonicalizeMethod must thus be chosen such that a canonical form is obtained, which for example can include the use of Young projectors to take into account multiterm symmetries; see the xTras package documentation for examples involving the Riemann tensor, which satisfies the multiterm Bianchi identities. The default value for CanonicalizeMethod is suitable for theories without fermions, while for theories with fermions one must also use JoinGammaMatrices, EpsilonGammaReduce, IrreducibleSpinDecompose, IrreducibleSpinProject, ... CohomologyFromAnsatz can be used to determine local gauge-invariant operators.

4.14.2 `RelativeCohomologyFromAnsatz`

> `RelativeCohomologyFromAnsatz[brst,ansatzcc,d,ansatzd,brstb,ansatzcb,db,ansatzdb]`
> returns a list of representatives of elements of the relative cohomology
> $\text{Ker}(brst|d)/\text{Im}(brstb|db)$. For the relative BRST cohomology $\text{H}(\mathsf{s}|\mathsf{d})$, one needs
> to take $brst = brstb = \texttt{BRST}$, and $d = db = \texttt{CD[-}\mu\texttt{]}$. A list of possible elements
> (cocycles) must be given as $ansatzcc$ and $ansatzd$, and a list of possible exact
> elements (coboundaries) as $ansatzcb$ and $ansatzdb$. All lists could be calculated
> using `GenerateMonomials` or `GenerateMonomialsByGrading`.

This is the analogue of `CohomologyFromAnsatz` for relative (co-)homologies, and can be used to determine possible anomalies and invariant actions, which both are only defined up to surface terms (total derivatives).

Acknowledgments

A Decomposition of tensor products

Here I give the full list of decompositions used by FIELDSX, valid for commuting and anticommuting Majorana spinors in four dimensions. These decompositions were obtained following the algorithm of [16], to which the reader is referred for details.

A.1 Commuting spinors without an inner bundle index

For two commuting spinors, Majorana flip relations show that all bilinears except the ones with one or two γ matrices vanish, such that

$$\bar{\psi}^A \psi_B = \frac{1}{4}\gamma^\mu{}_B{}^A \Psi_\mu^{(2,\mathbf{4})} - \frac{1}{8}\gamma^{\mu\nu}{}_B{}^A \Psi_{\mu\nu}^{(2,\mathbf{6})} \tag{A.1}$$

with the invariants

$$\Psi_\mu^{(2,\mathbf{4})} \equiv \bar{\psi}^C \gamma_{\mu C}{}^D \psi_D \,, \tag{A.2a}$$

$$\Psi_{\mu\nu}^{(2,\mathbf{6})} \equiv \bar{\psi}^C \gamma_{\mu\nu C}{}^D \psi_D \,. \tag{A.2b}$$

For three spinors, it follows that

$$\psi_A \bar{\psi}^B \psi_C = -\frac{1}{8}\gamma^{\mu\nu}{}_{(A}{}^B \Psi_{\mu\nu C)}^{(3,\mathbf{8})} + \frac{1}{8}\left(2\gamma^\nu{}_{(A}{}^B \delta^D_{C)} + \gamma^{\mu\nu}{}_{(A}{}^B \gamma_{\mu C)}{}^D\right)\Psi_{\nu D}^{(3,\mathbf{12})} \tag{A.3}$$

with the invariants

$$\Psi_{\mu A}^{(3,\mathbf{12})} \equiv \frac{3}{4}\Psi_\mu^{(2,\mathbf{4})}\psi_A - \frac{1}{4}\gamma_\mu{}^\nu{}_A{}^B \Psi_\nu^{(2,\mathbf{4})}\psi_B \,, \tag{A.4a}$$

$$\Psi_{\mu\nu A}^{(3,\mathbf{8})} \equiv \frac{1}{3}\Psi_{\mu\nu}^{(2,\mathbf{6})}\psi_A - \frac{1}{6}\gamma_{\mu\nu}{}^{\rho\sigma}{}_A{}^B\Psi_{\rho\sigma}^{(2,\mathbf{6})}\psi_B + \frac{1}{3}\gamma_{[\mu}{}^{\rho}{}_A{}^B\Psi_{\nu]\rho}^{(2,\mathbf{6})}\psi_B\,. \tag{A.4b}$$

For four spinors one obtains

$$\psi_A\bar{\psi}^B\psi_C\bar{\psi}^D = \frac{1}{16}\Big(2\gamma^\mu{}_{(A}{}^B\gamma^{\nu)}{}_C{}^D + \gamma^{\mu\rho}{}_{(A}{}^B\gamma^\nu{}_{\rho C)}{}^D\Big)\Psi_{\mu\nu}^{(4,\mathbf{9})}$$
$$- \frac{1}{6}\gamma^\mu{}_{(A}{}^B\gamma^{\rho\sigma}{}_{C)}{}^D\Psi_{\mu\rho\sigma}^{(4,\mathbf{16})} + \frac{1}{32}\gamma^{\mu\nu}{}_{(A}{}^B\gamma^{\rho\sigma}{}_{C)}{}^D\Psi_{\mu\nu\rho\sigma}^{(4,\mathbf{10})} \tag{A.5}$$

with the invariants

$$\Psi_{\mu\nu}^{(4,\mathbf{9})} \equiv \Psi_\mu^{(2,\mathbf{4})}\Psi_\nu^{(2,\mathbf{4})}\,, \tag{A.6a}$$

$$\Psi_{\mu\rho\sigma}^{(4,\mathbf{16})} \equiv \Psi_\mu^{(2,\mathbf{4})}\Psi_{\rho\sigma}^{(2,\mathbf{6})}\,, \tag{A.6b}$$

$$\Psi_{\mu\nu\rho\sigma}^{(4,\mathbf{10})} \equiv \Psi_{\mu\nu}^{(2,\mathbf{6})}\Psi_{\rho\sigma}^{(2,\mathbf{6})} - 2\Psi_{\alpha[\mu}^{(2,\mathbf{6})}g_{\nu][\rho}\Psi_{\sigma]}^{(2,\mathbf{6})\alpha} + \frac{1}{3}g_{\mu[\rho}g_{\sigma]\nu}\Psi_{\alpha\beta}^{(2,\mathbf{6})}\Psi^{(2,\mathbf{6})\alpha\beta}\,, \tag{A.6c}$$

where the relations (obtained from Fierz rearrangements)

$$\Psi_{[\mu}^{(2,\mathbf{4})}\Psi_{\nu]\rho}^{(2,\mathbf{6})} = -\frac{1}{2}\Psi_\rho^{(2,\mathbf{4})}\Psi_{\mu\nu}^{(2,\mathbf{6})}\,, \tag{A.7a}$$

$$\Psi_{\mu[\alpha}^{(2,\mathbf{6})}\Psi_{\beta]\nu}^{(2,\mathbf{6})} = -\frac{1}{2}\Psi_{\mu\nu}^{(2,\mathbf{6})}\Psi_{\alpha\beta}^{(2,\mathbf{6})} \tag{A.7b}$$

are needed to derive the above expressions.

A.2 Commuting spinors with an inner bundle index

For spinors with an additional inner bundle index (Lie-algebra valued Majorana spinors), the Majorana flip relations instead determine whether the symmetric or antisymmetric representation of the symmetric group (for permutations of the Lie algebra index) contributes. For two spinors, it follows that

$$\bar{\psi}^{aA}\psi_B^b = \frac{1}{4}\delta_B^A\Psi^{(2,\mathbf{d(d-1)/2})ab} + \frac{1}{4}\gamma_{\mu B}{}^A\Psi^{(2,\mathbf{2d(d+1)})ab\mu} - \frac{1}{8}\gamma_{\mu\nu B}{}^A\Psi^{(2,\mathbf{3d(d+1)})ab\mu\nu}$$
$$- \frac{1}{24}\epsilon^{\mu\nu\rho\sigma}\gamma_{\mu\nu\rho B}{}^A\Psi^{(2,\mathbf{2d(d-1)})ab}_\sigma - \frac{1}{96}\epsilon^{\mu\nu\rho\sigma}\gamma_{\mu\nu\rho\sigma B}{}^A\Psi^{(2,\mathbf{d(d-1)/2'})ab} \tag{A.8}$$

with the invariants

$$\Psi^{(2,\mathbf{d(d-1)/2})ab} \equiv \bar{\psi}^{[aC}\psi_C^{b]}\,, \tag{A.9a}$$

$$\Psi^{(2,\mathbf{2d(d+1)})ab}_\mu \equiv \bar{\psi}^{(aC}\gamma_{\mu C}{}^D\psi_D^{b)}\,, \tag{A.9b}$$

$$\Psi^{(2,\mathbf{3d(d+1)})ab}_{\mu\nu} \equiv \bar{\psi}^{(aC}\gamma_{\mu\nu C}{}^D\psi_D^{b)}\,, \tag{A.9c}$$

$$\Psi^{(2,\mathbf{2d(d-1)})ab}_\mu \equiv \frac{1}{6}\epsilon_{\mu\nu\rho\sigma}\bar{\psi}^{[aC}\gamma^{\nu\rho\sigma}{}_C{}^D\psi_D^{b]}\,, \tag{A.9d}$$

$$\Psi^{(2,\mathbf{d(d-1)/2'})ab} \equiv \frac{1}{24}\epsilon^{\mu\nu\rho\sigma}\bar{\psi}^{[aC}\gamma_{\mu\nu\rho\sigma C}{}^D\psi_D^{b]}\,, \tag{A.9e}$$

whose dimension now depends on the dimension d of the Lie algebra representation. In principle, the fully irreducible tensors are traceless also on the Lie algebra indices (and one thus should subtract the traces for the symmetric tensors $\Psi^{(2,\mathbf{d(d+1)/2})ab}$, ...), but this

does not seem to be useful for practical applications. Since a Lie-algebra valued Majorana spinor has $4d$ degrees of freedom, the counting of degrees of freedom also works out:

$$\frac{4d(4d+1)}{2} = \frac{d(d-1)}{2} + 2d(d+1) + 3d(d+1) + 2d(d-1) + \frac{d(d-1)}{2} . \qquad \text{(A.10)}$$

In the case $d = 1$, one naturally recovers the previous result.

For three spinors, the decomposition is already quite complicated:

$$
\begin{aligned}
\psi^a_A \bar\psi^{bB} \psi^c_C = & -\frac{1}{96}\left(24\delta^B_{[A}\delta^D_{C]} + 4\gamma_{\mu\nu\rho[A}{}^B \gamma^{\mu\nu\rho}{}_{C]}{}^D - \gamma_{\mu\nu\rho\sigma[A}{}^B \gamma^{\mu\nu\rho\sigma}{}_{C]}{}^D\right) \\
& \times \Psi^{(3,\mathbf{2d(d-1)(d-2)/3})\,\tiny\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array}\,abc}_{\quad D} \\
& -\frac{1}{192}\left(24\delta^B_A\delta^D_C + 6\gamma^\mu{}_C{}^B\gamma_{\mu A}{}^D - 2\gamma^{\mu\nu}{}_C{}^B\gamma_{\mu\nu A}{}^D - \gamma^{\mu\nu\rho}{}_A{}^B\gamma_{\mu\nu\rho C}{}^D\right) \\
& \times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3})\,\tiny\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\,abc}_{\quad D} \\
& -\frac{1}{192}\left(24\delta^B_C\delta^D_A + 6\gamma^\mu{}_A{}^B\gamma_{\mu C}{}^D - 2\gamma^{\mu\nu}{}_A{}^B\gamma_{\mu\nu C}{}^D - \gamma^{\mu\nu\rho}{}_C{}^B\gamma_{\mu\nu\rho A}{}^D\right) \\
& \times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3})\,\tiny\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\,cba}_{\quad D} \\
& -\frac{1}{48}\left(3\gamma^\mu{}_{[A}{}^B\gamma_{\mu C]}{}^D - \gamma^{\mu\nu}{}_{[A}{}^B\gamma_{\mu\nu C]}{}^D\right)\Psi^{(3,\mathbf{4d(d+1)(d-1)/3})\,\tiny\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\,cab}_{\quad D} \\
& -\frac{1}{192}\left(6\gamma^\mu{}_C{}^B\gamma_{\mu A}{}^D + 2\gamma^{\mu\nu}{}_C{}^B\gamma_{\mu\nu A}{}^D - \gamma^{\mu\nu\rho}{}_A{}^B\gamma_{\mu\nu\rho C}{}^D - \gamma^{\mu\nu\rho\sigma}{}_A{}^B\gamma_{\mu\nu\rho\sigma C}{}^D\right) \\
& \times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})\,\tiny\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\,abc}_{\quad D} \\
& -\frac{1}{192}\left(6\gamma^\mu{}_A{}^B\gamma_{\mu C}{}^D + 2\gamma^{\mu\nu}{}_A{}^B\gamma_{\mu\nu C}{}^D - \gamma^{\mu\nu\rho}{}_C{}^B\gamma_{\mu\nu\rho A}{}^D - \gamma^{\mu\nu\rho\sigma}{}_C{}^B\gamma_{\mu\nu\rho\sigma A}{}^D\right) \\
& \times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})\,\tiny\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\,cba}_{\quad D} \\
& -\frac{1}{48}\left(3\gamma^\mu{}_{[A}{}^B\gamma_{\mu C]}{}^D + \gamma^{\mu\nu}{}_{[A}{}^B\gamma_{\mu\nu C]}{}^D\right)\Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})\,\tiny\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\,cab}_{\quad D} \\
& +\frac{1}{8}\left(2\gamma^\nu{}_{(A}{}^B\delta^D_{C)} + \gamma^{\mu\nu}{}_{(A}{}^B\gamma_{\mu C)}{}^D\right)\Psi^{(3,\mathbf{2d(d+1)(d+2)})\,\tiny\begin{array}{|c|c|c|}\hline 1&2&3\\\hline\end{array}\,abc}_{\nu D} \\
& +\frac{1}{16}\left(2\gamma^\nu{}_A{}^B\delta^D_C - 2\gamma^{\mu\nu}{}_A{}^B\gamma_{\mu C}{}^D + \gamma^{\mu\nu\rho}{}_C{}^B\gamma_{\mu\rho A}{}^D\right)\Psi^{(3,\mathbf{4d(d+1)(d-1)})\,\tiny\begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array}\,abc}_{\nu D} \\
& +\frac{1}{16}\left(2\gamma^\nu{}_C{}^B\delta^D_A - 2\gamma^{\mu\nu}{}_C{}^B\gamma_{\mu A}{}^D + \gamma^{\mu\nu\rho}{}_A{}^B\gamma_{\mu\rho C}{}^D\right)\Psi^{(3,\mathbf{4d(d+1)(d-1)})\,\tiny\begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array}\,cba}_{\nu D} \\
& +\frac{1}{8}\gamma^{\mu\nu\rho}{}_{(A}{}^B\gamma_{\mu\nu C)}{}^D\Psi^{(3,\mathbf{4d(d+1)(d-1)})\,\tiny\begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array}\,cab}_{\rho D} \\
& -\frac{1}{8}\gamma_{\mu\nu(A}{}^B\Psi^{(3,\mathbf{4d(d+1)(d+2)/3})\,\tiny\begin{array}{|c|c|c|}\hline 1&2&3\\\hline\end{array}\,abc}_{\mu\nu C)} .
\end{aligned}
$$
$$\text{(A.11)}$$

The symmetry of the invariants with respect to permutations of the Lie algebra indices is now more complicated than total (anti-)symmetry, and described in the standard way by Young tableaux.[9] The invariants are given by

$$\Psi^{(3,\mathbf{2d(d-1)(d-2)/3})\,\tiny\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array}\,abc}_{\quad A} \equiv \Psi^{(2,\mathbf{d(d-1)/2})\,[ab}\psi^{c]}_A , \qquad \text{(A.12a)}$$

_____

[9]See for example [7] or the very readable lecture notes [48] and references therein.

$$\Psi^{(3,\mathbf{4d(d+1)(d-1)/3})}{}^{\boxed{\begin{smallmatrix}1&3\\2\end{smallmatrix}}abc}_A \equiv \frac{2}{3}\Psi^{(2,\mathbf{d(d-1)/2})ab}\psi^c_A - \frac{2}{3}\Psi^{(2,\mathbf{d(d-1)/2})c[a}\psi^{b]}_A\,, \tag{A.12b}$$

$$\Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})}{}^{\boxed{\begin{smallmatrix}1&3\\2\end{smallmatrix}}abc}_A \equiv \frac{2\mathrm{i}}{3}\gamma_{*A}{}^B\left[\Psi^{(2,\mathbf{d(d-1)/2'})ab}\psi^c_B - \Psi^{(2,\mathbf{d(d-1)/2'})c[a}\psi^{b]}_B\right]\,, \tag{A.12c}$$

$$\Psi^{(3,\mathbf{2d(d+1)(d+2)})}{}^{\boxed{\begin{smallmatrix}1&2&3\end{smallmatrix}}abc}_{\mu A} \equiv \frac{3}{4}\Psi^{(2,\mathbf{2d(d+1)})(ab}_\mu \psi^{c)}_A - \frac{1}{4}\gamma_\mu{}^\nu{}_A{}^B\Psi^{(2,\mathbf{2d(d+1)})(ab}_\nu \psi^{c)}_B\,, \tag{A.12d}$$

$$\Psi^{(3,\mathbf{4d(d+1)(d-1)})}{}^{\boxed{\begin{smallmatrix}1&2\\3\end{smallmatrix}}abc}_{\mu A} \equiv \frac{1}{2}\Psi^{(2,\mathbf{2d(d+1)})ab}_\mu \psi^c_A - \frac{1}{6}\gamma_\mu{}^\nu{}_A{}^B\Psi^{(2,\mathbf{2d(d+1)})ab}_\nu \psi^c_B$$
$$- \frac{1}{2}\Psi^{(2,\mathbf{2d(d+1)})c(a}_\mu \psi^{b)}_A + \frac{1}{6}\gamma_\mu{}^\nu{}_A{}^B\Psi^{(2,\mathbf{2d(d+1)})c(a}_\nu \psi^{b)}_B\,, \tag{A.12e}$$

$$\Psi^{(3,\mathbf{4d(d+1)(d+2)/3})}{}^{\boxed{\begin{smallmatrix}1&2&3\end{smallmatrix}}abc}_{\mu\nu A} \equiv \frac{1}{3}\Psi^{(2,\mathbf{3d(d+1)})(ab}_{\mu\nu}\psi^{c)}_A - \frac{1}{3}\gamma^\rho{}_{[\mu A}{}^B\Psi^{(2,\mathbf{3d(d+1)})(ab}_{\nu]\rho}\psi^{c)}_B$$
$$- \frac{1}{6}\gamma_{\mu\nu}{}^{\rho\sigma}{}_A{}^B\Psi^{(2,\mathbf{3d(d+1)})(ab}_{\rho\sigma}\psi^{c)}_B\,, \tag{A.12f}$$

and the counting of degrees of freedom also works out:

$$\frac{4d(4d+1)(4d+2)}{2\cdot 3} = \frac{2d(d-1)(d-2)}{3} + \frac{4d(d+1)(d-1)}{3} + \frac{4d(d+1)(d-1)}{3}$$
$$+ 2d(d+1)(d+2) + 4d(d+1)(d-1) + \frac{4d(d+1)(d+2)}{3}\,. \tag{A.13}$$

These decompositions assume that the Lie algebra and the chosen representation are sufficiently generic (with high enough dimension d), since otherwise it is possible that some of the invariant tensors vanish. For example, if $d = 2$ one obviously has

$$\Psi^{(2,\mathbf{2d(d-1)/2})ab} = \frac{1}{2}\epsilon^{ab}\epsilon_{cd}\Psi^{(2,\mathbf{2d(d-1)/2})cd} \tag{A.14}$$

with the antisymmetric tensor ϵ_{ab} normalised as $\epsilon_{12} = 1$, and total antisymmetrisation over more than two indices leads to a vanishing result: $\Psi^{(3,\mathbf{2d(d-1)(d-2)/3})}{}^{\boxed{\begin{smallmatrix}1\\2\\3\end{smallmatrix}}abc}_A = 0$. If applicable, such relations must be specified by the user.

A.3 Anticommuting spinors without an inner bundle index

For two anticommuting spinors, Majorana flip relations show that the bilinears with one or two γ matrices vanish, and one obtains

$$\bar{\psi}^A\psi_B = \frac{1}{4}\delta^A_B\Psi^{(2,\mathbf{1})} - \frac{1}{24}\gamma^{\mu\nu\rho}{}_B{}^A\epsilon_{\mu\nu\rho\sigma}\Psi^{(2,\mathbf{4})\sigma} - \frac{1}{96}\gamma^{\mu\nu\rho\sigma}{}_B{}^A\epsilon_{\mu\nu\rho\sigma}\Psi^{(2,\mathbf{1'})} \tag{A.15}$$

with the invariants

$$\Psi^{(2,\mathbf{1})} \equiv \bar{\psi}^C\psi_C\,, \tag{A.16a}$$

$$\Psi^{(2,\mathbf{4})}_\mu \equiv \frac{1}{6}\epsilon_{\mu\nu\rho\sigma}\bar{\psi}^C\gamma^{\nu\rho\sigma}{}_C{}^D\psi_D\,, \tag{A.16b}$$

$$\Psi^{(2,\mathbf{1'})} \equiv \frac{1}{24}\epsilon^{\mu\nu\rho\sigma}\bar{\psi}^C\gamma_{\mu\nu\rho\sigma C}{}^D\psi_D\,. \tag{A.16c}$$

For three anticommuting spinors, one has

$$\psi_A \bar{\psi}^B \psi_C = -\frac{1}{4}\left[\delta_{[A}^B \delta_{C]}^D + \frac{1}{6}\gamma_{\mu\nu\rho[A}{}^B \gamma^{\mu\nu\rho}{}_{C]}{}^D - \frac{1}{24}\gamma_{\mu\nu\rho\sigma[A}{}^B \gamma_{C]}^{\mu\nu\rho\sigma\,D}\right]\Psi_D^{(3,4)}, \qquad (A.17)$$

with the single independent invariant

$$\Psi_A^{(3,4)} \equiv \psi_A \Psi^{(2,1)}, \qquad (A.18)$$

which agrees with the counting of degrees of freedom $4(4-1)(4-2)/(2\cdot 3) = 4$. Lastly, for four spinors one has the decomposition

$$\psi_A \bar{\psi}^B \psi_C \bar{\psi}^D = \frac{1}{16}\left[\delta_{[A}^B \delta_{C]}^D + \frac{1}{6}\gamma_{\mu\nu\rho[A}{}^B \gamma^{\mu\nu\rho}{}_{C]}{}^D - \frac{1}{24}\gamma_{\mu\nu\rho\sigma[A}{}^B \gamma_{C]}^{\mu\nu\rho\sigma\,D}\right]\Psi^{(4,1)} \qquad (A.19)$$

with the invariant

$$\Psi^{(4,1)} = \left(\bar{\psi}^A \psi_A\right)^2, \qquad (A.20)$$

and the product of five or more anticommuting spinors vanishes since at least two components will be equal.

A.4 Anticommuting spinors with an inner bundle index

For Lie-algebra valued Majorana spinors, the Majorana flip relations again determine whether the symmetric or antisymmetric representation of the symmetric group (for permutations of the Lie algebra index) contributes. For two spinors, it follows that

$$\begin{aligned}
\bar{\psi}^{aA}\psi_B^b = {} & \frac{1}{4}\delta_B^A \Psi^{(2,\boldsymbol{d(d+1)/2})ab} + \frac{1}{4}\gamma_{\mu B}{}^A \Psi^{(2,\boldsymbol{2d(d-1)})ab\mu} - \frac{1}{8}\gamma_{\mu\nu B}{}^A \Psi^{(2,\boldsymbol{3d(d-1)})ab\mu\nu} \\
& - \frac{1}{24}\epsilon^{\mu\nu\rho\sigma}\gamma_{\mu\nu\rho B}{}^A \Psi^{(2,\boldsymbol{2d(d+1)})ab}{}_\sigma - \frac{1}{96}\epsilon^{\mu\nu\rho\sigma}\gamma_{\mu\nu\rho\sigma B}{}^A \Psi^{(2,\boldsymbol{d(d+1)/2'})ab}
\end{aligned} \qquad (A.21)$$

with the invariants

$$\Psi^{(2,\boldsymbol{d(d+1)/2})ab} \equiv \bar{\psi}^{(aC}\psi_C^{b)}, \qquad (A.22a)$$

$$\Psi^{(2,\boldsymbol{2d(d-1)})ab}{}_\mu \equiv \bar{\psi}^{[aC}\gamma_{\mu C}{}^D \psi_D^{b]}, \qquad (A.22b)$$

$$\Psi^{(2,\boldsymbol{3d(d-1)})ab}{}_{\mu\nu} \equiv \bar{\psi}^{[aC}\gamma_{\mu\nu C}{}^D \psi_D^{b]}, \qquad (A.22c)$$

$$\Psi^{(2,\boldsymbol{2d(d+1)})ab}{}_\mu \equiv \frac{1}{6}\epsilon_{\mu\alpha\beta\gamma}\bar{\psi}^{(aC}\gamma^{\alpha\beta\gamma}{}_C{}^D \psi_D^{b)}, \qquad (A.22d)$$

$$\Psi^{(2,\boldsymbol{d(d+1)/2'})ab} \equiv \frac{1}{24}\epsilon^{\mu\nu\rho\sigma}\bar{\psi}^{(aC}\gamma_{\mu\nu\rho\sigma C}{}^D \psi_D^{b)}, \qquad (A.22e)$$

whose dimension again depends on the dimension d of the Lie algebra representation, and the counting of degrees of freedom works out to be

$$\frac{4d(4d-1)}{2} = \frac{d(d+1)}{2} + 2d(d-1) + 3d(d-1) + 2d(d+1) + \frac{d(d+1)}{2}. \qquad (A.23)$$

For three anticommuting spinors, one obtains again a quite complicated result

$$
\psi_A^a \bar\psi^{bB} \psi_C^c = -\frac{1}{96}\Big(24\delta_{[A}^B \delta_{C]}^D + 4\gamma_{\mu\nu\rho[A}{}^B \gamma^{\mu\nu\rho}{}_{C]}{}^D - \gamma_{\mu\nu\rho\sigma[A}{}^B \gamma^{\mu\nu\rho\sigma}{}_{C]}{}^D\Big)
$$
$$
\times \Psi^{(3,\mathbf{2d(d+1)(d+2)/3})}{}_{\boxed{1\,2\,3}}{}^{abc}{}_D
$$
$$
-\frac{1}{192}\Big(24\delta_A^B \delta_C^D + 6\gamma^\mu{}_C{}^B \gamma_{\mu A}{}^D - 2\gamma^{\mu\nu}{}_C{}^B \gamma_{\mu\nu A}{}^D - \gamma^{\mu\nu\rho}{}_A{}^B \gamma_{\mu\nu\rho C}{}^D\Big)
$$
$$
\times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{abc}{}_D
$$
$$
+\frac{1}{192}\Big(24\delta_C^B \delta_A^D + 6\gamma^\mu{}_A{}^B \gamma_{\mu C}{}^D - 2\gamma^{\mu\nu}{}_A{}^B \gamma_{\mu\nu C}{}^D - \gamma^{\mu\nu\rho}{}_C{}^B \gamma_{\mu\nu\rho A}{}^D\Big)
$$
$$
\times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{cba}{}_D
$$
$$
-\frac{1}{48}\Big(3\gamma^\mu{}_{[A}{}^B \gamma_{\mu C]}{}^D - \gamma^{\mu\nu}{}_{[A}{}^B \gamma_{\mu\nu C]}{}^D\Big)\Psi^{(3,\mathbf{4d(d+1)(d-1)/3})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{cab}{}_D
$$
$$
-\frac{1}{192}\Big(6\gamma_{\mu C}{}^B \gamma^\mu{}_A{}^D + 2\gamma_{\mu\nu C}{}^B \gamma^{\mu\nu}{}_A{}^D - \gamma_{\mu\nu\rho A}{}^B \gamma^{\mu\nu\rho}{}_C{}^D - \gamma_{\mu\nu\rho\sigma A}{}^B \gamma^{\mu\nu\rho\sigma}{}_C{}^D\Big)
$$
$$
\times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{abc}{}_D
$$
$$
+\frac{1}{192}\Big(6\gamma_{\mu A}{}^B \gamma^\mu{}_C{}^D + 2\gamma_{\mu\nu A}{}^B \gamma^{\mu\nu}{}_C{}^D - \gamma_{\mu\nu\rho C}{}^B \gamma^{\mu\nu\rho}{}_A{}^D - \gamma_{\mu\nu\rho\sigma C}{}^B \gamma^{\mu\nu\rho\sigma}{}_A{}^D\Big)
$$
$$
\times \Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{cba}{}_D
$$
$$
-\frac{1}{48}\Big(3\gamma_{\mu[A}{}^B \gamma^\mu{}_{C]}{}^D + \gamma_{\mu\nu[A}{}^B \gamma^{\mu\nu}{}_{C]}{}^D\Big)\Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{cab}{}_D
$$
$$
+\frac{1}{8}\Big(2\gamma^\nu{}_{(A}{}^B \delta_{C)}^D + \gamma^{\mu\nu}{}_{(A}{}^B \gamma_{\mu C)}{}^D\Big)\Psi^{(3,\mathbf{2d(d-1)(d-2)})}{}_{\substack{\boxed{1}\\ \boxed{2}\\ \boxed{3}}}{}^{abc}{}_{\nu D}
$$
$$
+\frac{1}{16}\Big(2\gamma^\nu{}_A{}^B \delta_C^D - 2\gamma^{\mu\nu}{}_A{}^B \gamma_{\mu C}{}^D + \gamma^{\mu\nu\rho}{}_C{}^B \gamma_{\mu\rho A}{}^D\Big)\Psi^{(3,\mathbf{4d(d+1)(d-1)})}{}_{\substack{\boxed{1\,3}\\ \boxed{2}}}{}^{abc}{}_{\nu D}
$$
$$
-\frac{1}{16}\Big(2\gamma^\nu{}_C{}^B \delta_A^D - 2\gamma^{\mu\nu}{}_C{}^B \gamma_{\mu A}{}^D + \gamma^{\mu\nu\rho}{}_A{}^B \gamma_{\mu\rho C}{}^D\Big)\Psi^{(3,\mathbf{4d(d+1)(d-1)})}{}_{\substack{\boxed{1\,3}\\ \boxed{2}}}{}^{cba}{}_{\nu D}
$$
$$
+\frac{1}{8}\gamma^{\mu\nu\rho}{}_{(A}{}^B \gamma_{\mu\nu C)}{}^D \Psi^{(3,\mathbf{4d(d+1)(d-1)})}{}_{\substack{\boxed{1\,3}\\ \boxed{2}}}{}^{cab}{}_{\rho D}
$$
$$
-\frac{1}{8}\gamma^{\mu\nu}{}_{(A}{}^B \Psi^{(3,\mathbf{4d(d-1)(d-2)/3})}{}_{\substack{\boxed{1}\\ \boxed{2}\\ \boxed{3}}}{}^{abc}{}_{\mu\nu C)}
$$
$$
\tag{A.24}
$$

with the invariants

$$
\Psi^{(3,\mathbf{2d(d+1)(d+2)/3})}{}_{\boxed{1\,2\,3}}{}^{abc}{}_A \equiv \Psi^{(2,\mathbf{d(d+1)/2})(ab}\psi_A^{c)}\,,
\tag{A.25a}
$$

$$
\Psi^{(3,\mathbf{4d(d+1)(d-1)/3})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{abc}{}_A \equiv \frac{2}{3}\Psi^{(2,\mathbf{d(d+1)/2})ab}\psi_A^c - \frac{2}{3}\Psi^{(2,\mathbf{d(d+1)/2})c(a}\psi_A^{b)}\,,
\tag{A.25b}
$$

$$
\Psi^{(3,\mathbf{4d(d+1)(d-1)/3*})}{}_{\substack{\boxed{1\,2}\\ \boxed{3}}}{}^{abc}{}_A \equiv \frac{2i}{3}\gamma_{*A}{}^B\Big[\Psi^{(2,\mathbf{d(d+1)/2'})ab}\psi_B^c - \Psi^{(2,\mathbf{d(d+1)/2'})c(a}\psi_B^{b)}\Big]\,,
\tag{A.25c}
$$

$$
\Psi^{(3,\mathbf{2d(d-1)(d-2)})}{}_{\substack{\boxed{1}\\ \boxed{2}\\ \boxed{3}}}{}^{abc}{}_{\mu A} \equiv \frac{3}{4}\Psi^{(2,\mathbf{2d(d-1)})[ab}_\mu \psi_A^{c]} - \frac{1}{4}\gamma_\mu{}^\nu{}_A{}^B \Psi^{(2,\mathbf{2d(d-1)})[ab}_\nu \psi_B^{c]}\,,
\tag{A.25d}
$$

$$
\Psi^{(3,\mathbf{4d(d+1)(d-1)})}{}_{\substack{\boxed{1\,3}\\ \boxed{2}}}{}^{abc}{}_{\mu A} \equiv \frac{1}{2}\Psi^{(2,\mathbf{2d(d-1)})ab}_\mu \psi_A^c - \frac{1}{2}\Psi^{(2,\mathbf{2d(d-1)})c[a}_\mu \psi_A^{b]}
$$
$$
-\frac{1}{6}\gamma_\mu{}^\nu{}_A{}^B \Psi^{(2,\mathbf{2d(d-1)})ab}_\nu \psi_B^c + \frac{1}{6}\gamma_\mu{}^\nu{}_A{}^B \Psi^{(2,\mathbf{2d(d-1)})c[a}_\nu \psi_B^{b]}\,,
$$
$$
\tag{A.25e}
$$

$$
\Psi^{(3,4d(d-1)(d-2)/3)\,\tiny\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array}}{}^{abc}_{\mu\nu A} \equiv \frac{1}{3}\Psi^{(2,3d(d-1))\,[ab}_{\mu\nu}\psi^{c]}_A - \frac{1}{3}\gamma^\rho{}_{[\mu A}{}^B\Psi^{(2,3d(d-1))\,[ab}_{\nu]\rho}\psi^{c]}_B
$$
$$
- \frac{1}{6}\gamma_{\mu\nu}{}^{\rho\sigma}{}_A{}^B\Psi^{(2,3d(d-1))\,[ab}_{\rho\sigma}\psi^{c]}_B \,,
\tag{A.25f}
$$

and the counting of degrees of freedom

$$
\frac{4d(4d-1)(4d-2)}{6} = \frac{2d(d+1)(d+2)}{3} + \frac{4d(d+1)(d-1)}{3} + \frac{4d(d+1)(d-1)}{3}
$$
$$
+ 2d(d-1)(d-2) + 4d(d+1)(d-1) + \frac{4d(d-1)(d-2)}{3} \,.
\tag{A.26}
$$

Again we note that for $d=1$ we recover the previous result, and that for Lie algebra representations of low dimensionality some of these invariants can vanish.

B Young projectors

To deal with multiterm symmetries (such as the Bianchi identities for the Riemann tensor), we use Young projectors P associated to the corresponding Young tableaux, applied to the symmetric group describing permutations of the Lorentz indices.[10]

Let there be given tensor-spinors $t_{\mu A}$ and $t_{\mu\nu A} = t_{[\mu\nu]A}$ such that

$$
\gamma^\mu{}_A{}^B t_{\mu B} = 0 = \gamma^\mu{}_A{}^B t_{\mu\nu B} \,.
\tag{B.1}
$$

By multiplying with $\delta^A_C = \gamma_{*C}{}^D\gamma_{*D}{}^A$ and expanding $\gamma_{*D}{}^A\gamma_{\mu\cdots A}{}^B = -\frac{\mathrm{i}}{24}\epsilon_{\alpha\beta\gamma\delta}\gamma^{\alpha\beta\gamma\delta}{}_D{}^A\gamma_{\mu\cdots A}{}^B$ in generalised γ matrices in the following expressions, it follows that

$$
P_{\tiny\begin{array}{|c|}\hline\mu\\\hline\nu\\\hline\rho\\\hline\sigma\\\hline\end{array}}\gamma_{\mu\nu\rho A}{}^B t_{\sigma B} = \gamma_{[\mu\nu\rho A}{}^B t_{\sigma]B} = 0\,, \qquad P_{\tiny\begin{array}{|c|}\hline\mu\\\hline\nu\\\hline\rho\\\hline\end{array}}\gamma_{\mu A}{}^B t_{\nu\rho B} = \gamma_{[\mu A}{}^B t_{\nu\rho]B} = 0\,,
\tag{B.2a}
$$

$$
P_{\tiny\begin{array}{|c|}\hline\mu\\\hline\nu\\\hline\rho\\\hline\sigma\\\hline\end{array}}\gamma_{\mu\nu A}{}^B t_{\rho\sigma B} = \gamma_{[\mu\nu A}{}^B t_{\rho\sigma]B} = 0\,, \qquad P_{\tiny\begin{array}{|c|c|}\hline\alpha&\rho\\\hline\beta&\\\hline\mu&\\\hline\nu&\\\hline\end{array}}\gamma_{\mu\nu\rho A}{}^B t_{\alpha\beta B} = 0\,,
\tag{B.2b}
$$

and obviously all projectors which antisymmetrise five or more indices vanish in four dimensions. It follows that

$$
\gamma^{\mu\nu\rho}{}_A{}^B t^\alpha_B = P_{\tiny\begin{array}{|c|}\hline\mu\\\hline\nu\\\hline\rho\\\hline\end{array}}\otimes\tiny\begin{array}{|c|}\hline\alpha\\\hline\end{array}\,\gamma^{\mu\nu\rho}{}_A{}^B t^\alpha_B = \left(P_{\tiny\begin{array}{|c|c|}\hline\mu&\alpha\\\hline\nu&\\\hline\rho&\\\hline\end{array}} + P_{\tiny\begin{array}{|c|}\hline\mu\\\hline\nu\\\hline\rho\\\hline\alpha\\\hline\end{array}}\right)\gamma^{\mu\nu\rho}{}_A{}^B t^\alpha_B
\tag{B.3a}
$$
$$
= P_{\tiny\begin{array}{|c|c|}\hline\mu&\alpha\\\hline\nu&\\\hline\rho&\\\hline\end{array}}\gamma^{\mu\nu\rho}{}_A{}^B t^\alpha_B = \frac{3}{4}\gamma^{\mu\nu\rho}{}_A{}^B t^\alpha_B + \frac{3}{4}\gamma^{\alpha[\mu\nu}{}_A{}^B t^{\rho]}_B\,,
$$

$$
\gamma^\mu{}_A{}^B t^{\alpha\beta}_B = P_{\tiny\begin{array}{|c|c|}\hline\alpha&\mu\\\hline\beta&\\\hline\end{array}}\gamma^\mu{}_A{}^B t^{\alpha\beta}_B = \frac{2}{3}\gamma^\mu{}_A{}^B t^{\alpha\beta}_B - \frac{2}{3}\gamma^{[\alpha}{}_A{}^B t^{\beta]\mu}_B\,,
\tag{B.3b}
$$

---

[10]See for example [7] or the very readable lecture notes [48] and references therein.

$$\gamma^{\mu\nu}{}_A{}^B t_B^{\alpha\beta} = \left(P_{\boxed{\substack{\alpha\,\mu \\ \beta\,\nu}}} + P_{\boxed{\substack{\alpha\,\mu \\ \beta \\ \nu}}} + P_{\boxed{\substack{\alpha\,\nu \\ \beta \\ \mu}}} \right) \gamma^{\mu\nu}{}_A{}^B t_B^{\alpha\beta}$$

$$= \frac{5}{6}\gamma^{\mu\nu}{}_A{}^B t_B^{\alpha\beta} - \frac{1}{3}\gamma^{\mu[\alpha}{}_A{}^B t_B^{\beta]\nu} + \frac{1}{3}\gamma^{\nu[\alpha}{}_A{}^B t_B^{\beta]\mu} - \frac{1}{6}\gamma^{\alpha\beta}{}_A{}^B t_B^{\mu\nu} \,,$$
(B.3c)

$$\gamma^{\mu\nu\rho}{}_A{}^B t_B^{\alpha\beta} = \left(P_{\boxed{\substack{\alpha\,\mu \\ \beta\,\nu \\ \rho}}} + P_{\boxed{\substack{\alpha\,\mu \\ \beta\,\rho \\ \nu}}} + P_{\boxed{\substack{\alpha\,\nu \\ \beta\,\rho \\ \mu}}} \right) \gamma^{\mu\nu\rho}{}_A{}^B t_B^{\alpha\beta}$$

$$= \frac{1}{2}\gamma^{\mu\nu\rho}{}_A{}^B t_B^{\alpha\beta} + \frac{1}{2}\gamma^{\alpha\beta[\mu}{}_A{}^B t_B^{\nu\rho]} + \frac{1}{2}\gamma^{\alpha[\mu\nu}{}_A{}^B t_B^{\rho]\beta} - \frac{1}{2}\gamma^{\beta[\mu\nu}{}_A{}^B t_B^{\rho]\alpha} \,,$$
(B.3d)

$$\gamma^{\mu\nu\rho\sigma}{}_A{}^B t_B^{\alpha\beta} = \left(P_{\boxed{\substack{\alpha\,\mu \\ \beta\,\nu \\ \rho \\ \sigma}}} + P_{\boxed{\substack{\alpha\,\mu \\ \beta\,\rho \\ \nu \\ \sigma}}} + P_{\boxed{\substack{\alpha\,\nu \\ \beta\,\rho \\ \mu \\ \sigma}}} + P_{\boxed{\substack{\alpha\,\mu \\ \beta\,\sigma \\ \nu \\ \rho}}} + P_{\boxed{\substack{\alpha\,\nu \\ \beta\,\sigma \\ \mu \\ \rho}}} + P_{\boxed{\substack{\alpha\,\rho \\ \beta\,\sigma \\ \mu \\ \nu}}} \right) \gamma^{\mu\nu\rho\sigma}{}_A{}^B t_B^{\alpha\beta}$$

$$= \frac{3}{5}\gamma^{\mu\nu\rho\sigma}{}_A{}^B t_B^{\alpha\beta} - \frac{3}{5}\gamma^{\alpha[\mu\nu\rho}{}_A{}^B t_B^{\sigma]\beta} + \frac{3}{5}\gamma^{\beta[\mu\nu\rho}{}_A{}^B t_B^{\sigma]\alpha} + \frac{3}{5}\gamma^{\alpha\beta[\mu\nu}{}_A{}^B t_B^{\rho\sigma]} \,.$$
(B.3e)

FIELDSX applies these identities to the invariant tensors

$$\Psi^{(3,2d(d+1)(d+2))}{}_{\boxed{1\,2\,3}}{}_{\nu A}^{abc} \,, \qquad \Psi^{(3,4d(d+1)(d-1))}{}_{\boxed{\substack{1\,2 \\ 3}}}{}_{\nu A}^{abc} \,, \qquad \Psi^{(3,4d(d+1)(d+2)/3)}{}_{\boxed{1\,2\,3}}{}_{\mu\nu A}^{abc}$$
(B.4)

for commuting spinors and

$$\Psi^{(3,2d(d-1)(d-2))}{}_{\boxed{\substack{1 \\ 2 \\ 3}}}{}_{\nu A}^{abc} \,, \qquad \Psi^{(3,4d(d+1)(d-1))}{}_{\boxed{\substack{1\,3 \\ 2}}}{}_{\nu A}^{abc} \,, \qquad \Psi^{(3,4d(d-1)(d-2)/3)}{}_{\boxed{\substack{1 \\ 2 \\ 3}}}{}_{\mu\nu A}^{abc}$$
(B.5)

for anticommuting spinors.

In the same way, the products of the totally antisymmetric ϵ tensor and generalised γ matrices can be replaced by simpler expressions, by multiplying $\gamma^{\mu\cdots}{}_A{}^B$ with $\delta_C^A = \gamma_{*C}{}^D \gamma_{*D}{}^A$ and expanding $\gamma_{*D}{}^A \gamma_{\mu\cdots}{}_A{}^B = -\frac{\mathrm{i}}{24}\epsilon_{\alpha\beta\gamma\delta}\gamma^{\alpha\beta\gamma\delta}{}_D{}^A \gamma_{\mu\cdots}{}_A{}^B$ in generalised γ matrices, or alternatively (in odd dimensions) using that the antisymmetrisation over more than d indices vanishes in d dimensions. This results in the identities

$$\epsilon_{\mu_1\cdots\mu_d}\gamma^{\nu_1\cdots\nu_k} = (-1)^{k(d+1)}\frac{d!}{k!(d-k)!}\delta_{[\mu_1}^{\nu_1}\cdots\delta_{\mu_k}^{\nu_k}\epsilon_{\mu_{k+1}\cdots\mu_d]\rho_{d-k+1}\cdots\rho_d}\gamma^{\rho_{d-k+1}\cdots\rho_d} \,,$$
(B.6)

where now all indices of the generalised γ matrices are contracted with the ϵ tensor. In particular in $d = 4$ dimensions, we have

$$\epsilon_{\mu\nu\rho\sigma}\gamma^{\alpha\beta\gamma\delta} = \delta_{[\mu}^\alpha\delta_\nu^\beta\delta_\rho^\gamma\delta_{\sigma]}^\delta\epsilon_{\kappa\lambda\upsilon\tau}\gamma^{\kappa\lambda\upsilon\tau} \,,$$
(B.7a)

$$\epsilon_{\mu\nu\rho\sigma}\gamma^{\alpha\beta\gamma} = -4\delta_{[\mu}^\alpha\delta_\nu^\beta\delta_{\rho]}^\gamma\epsilon_{\sigma]\kappa\lambda\upsilon}\gamma^{\kappa\lambda\upsilon} \,,$$
(B.7b)

$$\epsilon_{\mu\nu\rho\sigma}\gamma^{\alpha\beta} = 6\delta_{[\mu}^\alpha\delta_\nu^\beta\epsilon_{\rho\sigma]\kappa\lambda}\gamma^{\kappa\lambda} \,,$$
(B.7c)

$$\epsilon_{\mu\nu\rho\sigma}\gamma^\alpha = -4\delta_{[\mu}^\alpha\epsilon_{\nu\rho\sigma]\kappa}\gamma^\kappa \,.$$
(B.7d)

Lastly, also the product of the totally antisymmetric ϵ tensor and an arbitrary number of other tensors can be projected onto the corresponding Young tableaux. FieldsX implements the following projections:

$$\epsilon_{\mu\nu\rho\sigma}t_{\alpha} = P_{\tiny\begin{array}{|c|c|}\hline \mu & \alpha \\\hline \nu \\\cline{1-1} \rho \\\cline{1-1} \sigma \\\cline{1-1}\end{array}}\epsilon_{\mu\nu\rho\sigma}t_{\alpha} = \frac{4}{5}\epsilon_{\mu\nu\rho\sigma}t_{\alpha} - \frac{4}{5}\epsilon_{\alpha[\mu\nu\rho}t_{\sigma]}\,, \tag{B.8a}$$

$$\epsilon_{\mu\nu\rho\sigma}t_{\alpha\beta} = \left(P_{\tiny\begin{array}{|c|c|c|}\hline \mu & \alpha & \beta \\\hline \nu \\\cline{1-1} \rho \\\cline{1-1} \sigma \\\cline{1-1}\end{array}} + P_{\tiny\begin{array}{|c|c|}\hline \mu & \alpha \\\hline \nu & \beta \\\hline \rho \\\cline{1-1} \sigma \\\cline{1-1}\end{array}} \right)\epsilon_{\mu\nu\rho\sigma}t_{\alpha\beta} = \frac{3}{5}\epsilon_{\alpha\beta[\mu\nu}t_{\rho\sigma]} + \frac{1}{30}\epsilon_{\mu\nu\rho\sigma}t_{\beta\alpha} - \frac{1}{30}\epsilon_{\beta[\mu\nu\rho}t_{\sigma]\alpha}$$
$$- \frac{1}{30}\epsilon_{\alpha[\mu\nu\rho}t_{|\beta|\sigma]} + \frac{19}{30}\epsilon_{\mu\nu\rho\sigma}t_{\alpha\beta} - \frac{19}{30}\epsilon_{\alpha[\mu\nu\rho}t_{\sigma]\beta} - \frac{19}{30}\epsilon_{\beta[\mu\nu\rho}t_{|\alpha|\sigma]}\,. \tag{B.8b}$$

References

[1] J. M. Martín-García, R. Portugal and L. R. U. Manssur, *The Invar Tensor Package*, *Comput. Phys. Commun.* **177** (2007) 640–648, [`0704.1756`].

[2] J. M. Martín-García, D. Yllanes and R. Portugal, *The Invar tensor package: Differential invariants of Riemann*, *Comput. Phys. Commun.* **179** (2008) 586–590, [`0802.1274`].

[3] J. M. Martín-García, *xPerm: fast index canonicalization for tensor computer algebra*, *Comput. Phys. Commun.* **179** (2008) 597–603, [`0803.0862`].

[4] D. Brizuela, J. M. Martín-García and G. A. Mena Marugan, *xPert: Computer algebra for metric perturbation theory*, *Gen. Rel. Grav.* **41** (2009) 2415–2431, [`0807.0824`].

[5] A. G.-P. Gómez-Lobo and J. M. Martín-García, *Spinors: a Mathematica package for doing spinor calculus in General Relativity*, *Comput. Phys. Commun.* **183** (2012) 2214–2225, [`1110.2662`].

[6] C. Pitrou, X. Roy and O. Umeh, *xPand: An algorithm for perturbing homogeneous cosmologies*, *Class. Quant. Grav.* **30** (2013) 165002, [`1302.6174`].

[7] T. Nutma, *xTras: A field-theory inspired xAct package for mathematica*, *Comput. Phys. Commun.* **185** (2014) 1719–1738, [`1308.3493`].

[8] xAct-contrib, *Community contributions to xAct*, *contrib.xact.es* .

[9] K. Peeters, *A Field-theory motivated approach to symbolic computer algebra*, *Comput. Phys. Commun.* **176** (2007) 550–558, [`cs/0608005`].

[10] K. Peeters, *Cadabra2: computer algebra for field theory revisited*, *J. Open Source Softw.* **3** (32) 1118, [`hep-th/0701238`].

[11] S. Ferrara and B. Zumino, *Supergauge invariant Yang-Mills theories*, *Nucl. Phys. B* **79** (1974) 413.

[12] A. Salam and J. Strathdee, *Super-symmetry and non-Abelian gauges*, *Phys. Lett. B* **51** (1974) 353.

[13] B. de Wit and D. Z. Freedman, *Combined supersymmetric and gauge-invariant field theories*, *Phys. Rev. D* **12** (1975) 2286.

[14] M. Fierz, *Zur Fermischen Theorie des β-Zerfalls*, *Z. Physik* **104** (1937) 553.

[15] D. Z. Freedman and A. van Proeyen, *Supergravity*. Cambridge University Press, Cambridge, UK, 2012.

[16] R. D'Auria, P. Fré, E. Maina and T. Regge, *A new group theoretical technique for the analysis of Bianchi identities and its application to the auxiliary field problem of $D = 5$ supergravity*, *Annals Phys.* **139** (1982) 93.

[17] M. Forger and H. Römer, *Currents and the energy-momentum tensor in classical field theory: a fresh look at an old problem*, *Annals Phys.* **309** (2004) 306, [`hep-th/0307199`].

[18] C. Becchi, A. Rouet and R. Stora, *Renormalization of Gauge Theories*, *Ann. Phys.* **98** (1976) 287.

[19] I. V. Tyutin, *Gauge Invariance in Field Theory and Statistical Physics in Operator Formalism*, `0812.0580`.

[20] I. Batalin and G. Vilkovisky, *Gauge algebra and quantization*, *Phys. Lett. B* **102** (1981) 27.

[21] I. Batalin and G. Vilkovisky, *Quantization of gauge theories with linearly dependent generators*, *Phys. Rev. D* **28** (1983) 2567.

[22] I. Batalin and G. Vilkovisky, *Erratum: Quantization of gauge theories with linearly dependent generators*, *Phys. Rev. D* **30** (1984) 508.

[23] M. Henneaux, *Lectures on the antifield-BRST formalism for gauge theories*, *Nucl. Phys. B Proc. Suppl.* **18A** (1990) 47.

[24] J. Gomis, J. Paris and S. Samuel, *Antibracket, antifields and gauge-theory quantization*, *Phys. Rept.* **259** (1995) 1, [`hep-th/9412228`].

[25] G. Barnich, F. Brandt and M. Henneaux, *Local BRST cohomology in gauge theories*, *Phys. Rept.* **338** (2000) 439, [`hep-th/0002245`].

[26] F. Brandt, M. Henneaux and A. Wilch, *Global symmetries in the antifield formalism*, *Phys. Lett. B* **387** (1996) 320, [`hep-th/9606172`].

[27] F. Brandt, M. Henneaux and A. Wilch, *Extended antifield formalism*, *Nucl. Phys. B* **510** (1998) 640, [`hep-th/9705007`].

[28] P. K. Townsend, *Covariant quantization of antisymmetric tensor gauge fields*, *Phys. Lett. B* **88** (1979) 97.

[29] M. A. Namazie and D. Storey, *On secondary and higher-generation ghosts*, *J. Phys. A* **13** (1980) L161.

[30] J. Thierry-Mieg, *BRS structure of the antisymmetric tensor gauge theories*, *Nucl. Phys. B* **335** (1990) 334.

[31] W. Siegel, *Hidden Ghosts*, *Phys. Lett. B* **93** (1980) 170.

[32] T. Kimura, *Counting of ghosts in quantized antisymmetric tensor gauge field of third rank*, *J. Phys. A* **13** (1980) L353.

[33] T. Kimura, *Quantum Theory of Antisymmetric Higher Rank Tensor Gauge Field in Higher Dimensional Space-Time*, *Prog. Theor. Phys.* **65** (1981) 338.

[34] O. Piguet and S. P. Sorella, *Algebraic Renormalization: Perturbative Renormalization, Symmetries and Anomalies.* Springer-Verlag, Berlin/Heidelberg, Germany, 1995.

[35] N. Dragon and F. Brandt, *BRST Symmetry and Cohomology*, in *Strings, gauge fields, and the geometry behind: The legacy of Maximilian Kreuzer* (A. Rebhan, L. Katzarkov, J. Knapp, R. Rashkov and E. Scheidegger, eds.), pp. 3–86. 5, 2012. `1205.3293`. 10.1142/9789814412551_0001.

[36] J. Wess and B. Zumino, *Consequences of anomalous Ward identities*, *Phys. Lett. B* **37** (1971) 95–97.

[37] G. Barnich and M. Henneaux, *Consistent couplings between fields with a gauge freedom and deformations of the master equation*, *Phys. Lett. B* **311** (1993) 123–129, [`hep-th/9304057`].

[38] G. Barnich, M. Henneaux and R. Tatar, *Consistent interactions between gauge fields and the local BRST cohomology: The Example of Yang-Mills models*, *Int. J. Mod. Phys. D* **3** (1994) 139–144, [`hep-th/9307155`].

[39] O. Piguet and K. Sibold, *Renormalization of $N = 1$ supersymmetric Yang-Mills theories (I). The classical theory*, *Nucl. Phys. B* **197** (1982) 257–271.

[40] O. Piguet and K. Sibold, *Renormalization of $N = 1$ supersymmetric Yang-Mills theories (II). The radiative corrections*, *Nucl. Phys. B* **197** (1982) 272–289.

[41] J. A. Dixon, *Calculation of BRS cohomology with spectral sequences*, *Commun. Math. Phys.* **139** (1991) 495–526.

[42] T. Clark, O. Piguet and K. Sibold, *Supercurrents, Renormalization and Anomalies*, *Nucl. Phys. B* **143** (1978) 445–484.

[43] O. Piguet and K. Sibold, *The anomaly in the Slavnov identity for $N = 1$ supersymmetric Yang-Mills Theories*, *Nucl. Phys. B* **247** (1984) 484.

[44] K.-i. Konishi and K.-i. Shizuya, *Functional-Integral Approach to Chiral Anomalies in Supersymmetric Gauge Theories*, *Nuovo Cim. A* **90** (1985) 111.

[45] E. Guadagnini and M. Mintchev, *Chiral anomalies and supersymmetry*, *Nucl. Phys. B* **269** (1986) 543–556.

[46] A. Sudbery, *Computer-friendly d-tensor identities for SU(n)*, *J. Phys. A: Math. Gen.* **23** (1990) L705–L709.

[47] J. A. de Azcárraga, A. J. Macfarlane, A. J. Mountain and J. C. Pérez Bueno, *Invariant tensors for simple groups*, *Nucl. Phys. B* **510** (1998) 657–687, [`physics/9706006`].

[48] J. M. Alcock-Zeilinger, *The Special Unitary Group, Birdtracks, and Applications in QCD*, *Lecture Notes Universität Tübingen* (2018) .