AEI 2010-127

PSS: A FORM Program to Evaluate Pure Spinor Superspace Expressions

Carlos R. Mafra[1]

Max-Planck-Institut für Gravitationsphysik

Albert-Einstein-Institut, 14476 Golm, Germany

A `FORM` program which is used to efficiently expand in components pure spinor superfield expressions of kinematic factors is presented and comments on how it works are made. It is highly customizable using the standard features of `FORM` and can be used to help obtaining superstring effective actions from the scattering amplitudes computed with the pure spinor formalism.

July 2010

[1] email: crmafra@aei.mpg.de

1. Introduction

Since the discovery of the pure spinor formalism [1] the computation of manifestly supersymmetric superstring scattering amplitudes became possible[2]. At first the results were limited to tree-level, where it was shown that amplitudes with an arbitrary number of bosonic and up to four fermionic states were equivalent to the standard results from the Ramond-Neveu-Schwarz (RNS) and Green-Schwarz (GS) formulations [5]. Explicit four-point tree computations were first performed in [6], while in [7] those results were streamlined in a superspace derivation which also made manifest its relation with one- and two-loop amplitudes. The five-point amplitude was computed in [8], providing a compact superspace representation which contrasts with the bosonic-only result from [9]. In addition, an OPE identity related to the Bern-Carrasco-Johansson kinematic relations [10] was uncovered[3], which led to further developments discussed in [13]. Furthermore, it was shown in [14] that there is a BRST-equivalent superspace expression for the field theory limit of the superstring amplitude from [8] which provides hints of a direct mapping between Feynman diagrams with cubic vertices and pure spinor superspace expressions.

After being extended in [15,16], higher-loop amplitude computations using the pure spinor formalism also became a reality. At one-loop, the massless four-point [15,17], five-point [18] and the gauge variation of the six-point amplitude [19] were obtained. At two-loops, the massless four-point amplitude was computed in [20,21]. Using formulæ for integration over pure spinor space, the overall coefficients of the one-loop [22] and two-loop [23] were also computed and shown to agree with S-duality conjecture expectations [24].

Besides streamlining amplitude computations avoiding supermoduli spaces and sums over spins structures altogether, the pure spinor formalism naturally gives rise to manifestly supersymmetric kinematic factors in *pure spinor superspace*.

Pure spinor superspace expressions are correlation functions written in terms of ten-dimensional super-Yang-Mills superfields [25] and three pure spinors λ^α normalized as[4]

$$\langle(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\lambda\gamma^p\theta)(\theta\gamma_{mnp}\theta)\rangle = 1. \tag{1.1}$$

The simplest example of a pure spinor superspace expression is provided by the massless three-point scattering amplitude [1]

$$K_3^{(0)} = \langle(\lambda A^1)(\lambda A^2)(\lambda A^3)\rangle. \tag{1.2}$$

[2] For reviews see [2,3,4].

[3] See also [11,12] for string theory monodromy explanations of the BCJ relations.

[4] The precise overall coefficients of [22,23] will not be needed here.

The four-point kinematic factors are given by

$$K_4^{(0)} = \langle (\lambda A^1)(\lambda\gamma^m W^2)(\lambda\gamma^n W^3)\mathcal{F}^4_{mn}\rangle, \tag{1.3}$$

$$K_4^{(1)} = \langle (\lambda A^1)(\lambda\gamma^m W^2)(\lambda\gamma^n W^3)\mathcal{F}^4_{mn}\rangle, \tag{1.4}$$

$$K_4^{(2)} = \langle (\lambda\gamma^{mnpqr}\lambda)(\lambda\gamma^s W^4)\mathcal{F}^1_{mn}\mathcal{F}^2_{pq}\mathcal{F}^3_{rs}\rangle \tag{1.5}$$

for the tree-level [7], one- [15] and two-loop (1.5) amplitudes, respectively. Another example is provided by the one-loop five-point kinematic factor of [18], whose expression for the (12) and (25) "channels" read

$$L_{12} = \langle \Big[(\lambda A^1)(k^1\cdot A^2) + A^1_p(\lambda\gamma^p W^2)\Big](\lambda\gamma^m W^5)(\lambda\gamma^n W^3)\mathcal{F}^4_{mn}\rangle$$
$$K_{25} = \langle (\lambda A^1)\Big[(\lambda\gamma^m W^2)(k^2\cdot A^5) - \frac{1}{4}(\lambda\gamma^m\gamma^{rs}W^5)\mathcal{F}^2_{rs}\Big](\lambda\gamma^n W^3)\mathcal{F}^4_{mn}\Big\rangle - (2\leftrightarrow 5).$$

The above pure spinor superspace representations provide compact information about the amplitudes, but it may be convenient to evaluate those expressions in terms of familiar component expansions. These component expansions are written in terms of polarization vectors e^I_m and spinors χ^α_I with momenta k^I_m, where $I = 1,\ldots,N$ are the particle labels and $m = 0,\ldots,9$ $\alpha = 1,\ldots,16$ are the Lorentz and Weyl indices of ten dimensional Minkowski space.

The general method to evaluate these expressions in components was explained in the appendix of [19]. One uses the θ-expansions [26,6] of the SYM superfields

$$A_\alpha(x,\theta) = \frac{1}{2}a_m(\gamma^m\theta)_\alpha - \frac{1}{3}(\xi\gamma_m\theta)(\gamma^m\theta)_\alpha - \frac{1}{32}F_{mn}(\gamma_p\theta)_\alpha(\theta\gamma^{mnp}\theta) + \ldots$$

$$A_m(x,\theta) = a_m - (\xi\gamma_m\theta) - \frac{1}{8}(\theta\gamma_m\gamma^{pq}\theta)F_{pq} + \frac{1}{12}(\theta\gamma_m\gamma^{pq}\theta)(\partial_p\xi\gamma_q\theta) + \ldots$$

$$W^\alpha(x,\theta) = \xi^\alpha - \frac{1}{4}(\gamma^{mn}\theta)^\alpha F_{mn} + \frac{1}{4}(\gamma^{mn}\theta)^\alpha(\partial_m\xi\gamma_n\theta) + \frac{1}{48}(\gamma^{mn}\theta)^\alpha(\theta\gamma_n\gamma^{pq}\theta)\partial_m F_{pq} + \ldots$$

$$\mathcal{F}_{mn}(x,\theta) = F_{mn} - 2(\partial_{[m}\xi\gamma_{n]}\theta) + \frac{1}{4}(\theta\gamma_{[m}\gamma^{pq}\theta)\partial_{n]}F_{pq} + \ldots \tag{1.6}$$

where $a_m(x) = e_m e^{ikx}$ and $\xi^\alpha(x) = \chi^\alpha e^{ikx}$. After that, only terms containing five θ's are kept. Using symmetry alone it is possible to rewrite arbitrary combinations of $\langle\lambda^3\theta^5\rangle$ in terms of Kronecker deltas and epsilon tensors [21,27]. For example

$$\langle(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\lambda\gamma^p\theta)(\theta\gamma_{qrs}\theta)\rangle = \frac{1}{120}\delta^{mnp}_{qrs}. \tag{1.7}$$

Substituting the various correlators by their corresponding tensors as above, the component expansion of pure spinor superspace expressions is obtained.

Several different computer-aided procedures were used along the past years for the above steps in superspace derivations of scattering amplitudes. At first, those pure spinor correlators of [19] were obtained with the help[5] of the `GAMMA` package [28]. With some effort, the superspace expressions were expanded in θ's by hand, the corresponding correlators identified from a catalog composed of entries like (1.7) and the resulting tensors were typed in Mathematica or `FORM` [29]. When the scattering involved fermionic states, various Fierz identities were usually necessary at intermediate stages. After all these steps, the output consisted of several terms composed of Kronecker deltas and Levi-Civita epsilon tensors contracted with momenta and polarizations. Those terms constituted the final answer.

However the semi-automated method described above does not scale very well for higher-point amplitudes or when there are many expressions to evaluate in sequence. This fact provided the motivation to write the program presented here. It was developed mainly to help the author's own workflow during computations, and therefore it reflects his priorities (and it is a continuous work in progress). It is called `PSS` and it is written in the interesting language of `FORM` [29].

2. How PSS works

The goal is to be able to obtain component expansions in a fully automated process – all that is required is the pure spinor superspace expression to be expanded and whether the external states are bosonic or fermionic. The rest must be done by the computer.

For example, the superspace expression for [17] is typed in `PSS` as

```
Local [4-pts_one-loop] = la*A1*la*ga(m)*W2*la*ga(n)*W3*cF4(m,n);
```

[5] A small C program was also written to deal with anti-symmetrization of huge tensors during consistency checks.

Choosing all fields to be bosonic results in the following *ipsis litteris* output,

```
    [4-pts_one-loop] =
        - 1/5760*k1.e2*k1.e3*e1.e4*t
        + 1/5760*k1.e2*k1.e4*e1.e3*t
        + 1/5760*k1.e2*k2.e3*e1.e4*u
        + 1/5760*k1.e2*k2.e4*e1.e3*t
        + 1/5760*k1.e2*k3.e1*e3.e4*t
        + 1/5760*k1.e3*k2.e1*e2.e4*t
        - 1/5760*k1.e3*k2.e4*e1.e2*t
        - 1/5760*k1.e3*k3.e2*e1.e4*u
        - 1/5760*k1.e3*k3.e2*e1.e4*t
        + 1/5760*k1.e4*k2.e1*e2.e3*u
        - 1/5760*k1.e4*k2.e3*e1.e2*u
        + 1/5760*k1.e4*k3.e2*e1.e3*u
        + 1/5760*k1.e4*k3.e2*e1.e3*t
        - 1/5760*k2.e1*k2.e3*e2.e4*u
        + 1/5760*k2.e1*k2.e4*e2.e3*u
        + 1/5760*k2.e1*k3.e2*e3.e4*u
        - 1/5760*k2.e3*k3.e1*e2.e4*u
        - 1/5760*k2.e3*k3.e1*e2.e4*t
        + 1/5760*k2.e4*k3.e1*e2.e3*u
        + 1/5760*k2.e4*k3.e1*e2.e3*t
        + 1/5760*k3.e1*k3.e2*e3.e4*u
        + 1/5760*k3.e1*k3.e2*e3.e4*t
        + 1/11520*e1.e2*e3.e4*t*u
        - 1/11520*e1.e3*e2.e4*t*u
        - 1/11520*e1.e3*e2.e4*t^2
        - 1/11520*e1.e4*e2.e3*u^2
        - 1/11520*e1.e4*e2.e3*t*u
       ;
  Momentum conservation: k4 eliminated
  Gauge invariance: not tested
    0.05 sec + 0.08 sec: 0.13 sec out of 0.15 sec
```

which is the result obtained in [17]. One should notice in the final statistics displayed by `FORM` how quickly the answer is obtained.

The program is composed of one main FORM script called `pss.frm` and four header files: `pss_header.h`, `kin_factor.h`, `pss.h` and `ps_tensors.h`. They contain the definitions of indices, vectors, tensors, superfields etc (`pss_header.h`), the pure spinor superspace expressions to be evaluated (`kin_factor.h`) and the procedures which actually do the computations (`pss.h`). The database of pure spinor correlators is contained in `ps_tensors.h`. There is also a small `sed` script (FORM2tex.sed) to help translating the result into $T_{\rm E}X$.

To use the program one has to write down the kinematic factor in `kin_factor.h`. In the beginning of the main file `pss.frm`, the number of points must be defined (e.g. `#define Npts ``4''`) and whether the external states are bosonic or fermionic (`#define field1 ``0''` and so forth). After that one executes `pss.frm` using either form or tform (for multiprocessor computers) and a result like the one written above is obtained. Optionally one can select which momentum to be eliminated by setting the dollar variable `$kn` and test for gauge invariance by uncommenting the line containing `id e1 = k1; #gauge = e1;` close to the end of the file. Several other things can be done, depending on the problem at hand and how one chooses to manipulate it. There are also a few debug options (-d psonly, sfexpand, nofierz) which help in case something goes wrong and one has to check where. They will be explained below.

2.1. User input and notation

The superspace expression is written in terms of the super-Yang-Mills superfields $A^I_\alpha(\theta,x)$, $A^I_m(\theta,x)$, $W^\alpha_I(\theta,x)$ and $\mathcal{F}^I_{mn}(\theta,x)$ [25]. Their definitions are contained in the file `pss_header.h` and they correspond to,

$$\texttt{A1},\ \texttt{B1},\ \texttt{W1},\ \texttt{cF1(m,n)}$$

The pure spinor λ^α and θ^α are denoted by `la` and `th`. Note that PSS does not know about spinor indices, but that does not cause problems as long as one writes down correct superspace expressions. However, that also means one has to take care whether a fermionic superfield is contracted from the left or right, because the θ-expansions to be used differ

in this case[6]. For this situation one has to use the "left" version of the superfields, AL1 and WL1 etc. For example, the factor $(W^1\gamma_{mnp}W^2)$ must be written as

$$\texttt{WL1} * \texttt{ga}(\texttt{m},\texttt{n},\texttt{p}) * \texttt{W2}$$

The 16×16 gamma matrices $\gamma^m, \gamma^{mn}, \gamma^{mnp}, \ldots$ are denoted by

$$\texttt{ga}(\texttt{m}),\ \ \texttt{ga}(\texttt{m},\texttt{n}),\ \ \texttt{ga}(\texttt{m},\texttt{n},\texttt{p}),\ \ \ldots$$

Another convention to be followed is to write the pure spinor λ^α to the left of fermionic bilinears, so $(\lambda\gamma^m W^2)$ or $(W^2\gamma^m\lambda)$ must be written as

$$\texttt{la} * \texttt{ga}(\texttt{m}) * \texttt{W2}$$

and not

$$\texttt{WL2} * \texttt{ga}(\texttt{m}) * \texttt{la}$$

although PSS can be easily modified to accept the latter version. Of course, there is no problem to write the factor $(\lambda\gamma^{mnpqr}\lambda)$ and the procedure to identify correlators is aware of it. The generalized Kronecker delta is defined by[7]

$$N!\delta^{m_1\ldots m_N}_{n_1\ldots n_N} = \mathrm{da}(m_1,\ldots,m_N,n_1,\ldots,n_N)$$

2.2. The computation

It is important to understand how PSS actually obtains the component expansions, so that modifications can be easily done. All the action happens inside `pss.frm`, where it calls the procedures from `pss.h`. Let us now follow some of the steps[8].

When PSS is executed it loads the headers and the kinematic factors. The kinematic factors are local variables which will be manipulated by the `FORM` program. The first part of the manipulations transform the superspace input into an expression suitable for

[6] This is an artifact of how PSS was designed and this distinction is meaningless in real life.

[7] Unfortunately `FORM` has no notion of a generalized Kronecker delta, so it had to be defined by PSS. Note that the usual Kronecker delta is defined by `FORM` as δ^m_n = d_(m,n).

[8] One should also read the source code, as some details will be skipped.

identifying the required pure spinor correlators in terms of tensors to arrive at the final answer. The (slightly simplified) sequence of procedures is the following:

```
#call chooseMomentum()
#call superfieldExpand()
#call keep5thetas()
#call gammaExpand()
#call deltaExpand()
#call onShell()
#call orderFermions()
#call PSordering()
#call Fierz()
#call earlySimplify()
#call identifyCorrelators()
#call deltaExpand()
#call dualizeGammas()
#call fieldStrength()
#call onShell()
#call gammaExpand()
#call orderFermions()
#call diracEquation()
#call momentumConservation()
```

These steps are almost self-explanatory, and they correspond to what one would actually do in a computation with pen and paper.

The procedure `chooseMomentum()` is called to choose which momentum to eliminate. Although the result does not depend on the choice, there are differences in the number of intermediate terms and computing time. For example, when there are fermionic fields it is convenient to eliminate one momentum whose label is not one of the labels of the fermionic particles. The reason is the increased chance of applying the Dirac equation on fermion bilinears to reduce the rank of the gamma matrix. For example, $(\chi^1\gamma^{mnp}\chi^2)k^1_m = 2(\chi^1\gamma^{[n}\chi^2)k_1^{p]}$.

Then the procedure `superfieldExpand()` is called, which expands the SYM superfields in their θ components using (1.6) and selects the terms according to whether the

particles are bosonic or fermionic. The assignment of particles to each label of the superfields is done at the beginning of `pss.frm` with

```
#define field1 ‘‘0’’
#define field2 ‘‘0’’
...
```

where 0 (1) means bosonic (fermionic). Terms which do not contain five θs are then discarded with `keep5thetas()`. At this point, one of the terms in (1.4) looks like

```
- 1/256*la*ga(e1)*th*la*ga(k4)*ga(m2,n2)*th*la*ga(n)*ga(m3,n3)*th*
      th*ga(n,m4,n4)*th*F2(m2,n2)*F3(m3,n3)*F4(m4,n4)
```

which is the `FORM` output for

$$-\frac{1}{256}\langle(\lambda\gamma^m\theta)(\lambda\gamma^p\gamma^{m_2n_2}\theta)(\lambda\gamma^n\gamma^{m_3n_3}\theta)(\theta\gamma_{nm_4n_4}\theta)\rangle k_p^4 e_m^1 F^2_{m_2n_2}F^3_{m_3n_3}F^4_{m_4n_4} \qquad (2.1)$$

in the schoonschip notation. The gamma matrices in (2.1) are expanded using `gammaExpand()`,

$$\gamma^n\gamma^{m_3n_3} = \gamma^{nm_3n_3} + \eta^{m_3n}\gamma^{n_3} - \eta^{n_3n}\gamma^{m_3}.$$

More complicated expansions may introduce generalized Kronecker deltas, which are then expanded with `deltaExpand()`, e.g., $\delta^{mn}_{pq} = \frac{1}{2}(\delta^m_p\delta^n_q - \delta^m_q\delta^n_p)$. The procedure `onShell()` kills any term which may have been generated at this point containing $(k^i\cdot k^i)$ or $(k^i\cdot e^i)$.

When there are fermionic external states, the procedure `orderFermions()` rewrites the fermionic bilinears $(\chi^i\gamma^{m_1\cdots m_n}\chi^j)$ such that $i<j$. This procedure keeps track of overall minus signs which may be needed due to the Grassmanian nature of χ's and the symmetry properties of the gamma matrices. For example $(\chi^3\gamma^m\chi^1)\to-(\chi^1\gamma^m\chi^3)$ or $(\chi^3\gamma^{mnp}\chi^1)\to+(\chi^1\gamma^{mnp}\chi^3)$.

The procedure `PSordering()` follows a set of conventions on the ordering of fermionic bilinears to minimize the number of pattern matching when trying to identify the pure spinor correlators needed for one particular computation. For example, if the expression $(\lambda\gamma^m\theta)(\theta\gamma_{qrs}\theta)(\lambda\gamma^n\theta)(\lambda\gamma^p\theta)$ is encountered in the middle of the computations, it is first rewritten by `PSordering()` as $(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\lambda\gamma^p\theta)(\theta\gamma_{qrs}\theta)$. Later on there will be only one pattern to match in order to identify the correlator (in `identifyCorrelators()`) and replace it with $\frac{1}{720}$da(m,n,p,q,r,s). For the same reason, the ordering "inside" gamma

matrix bilinears is such that the fields appear as $(\lambda\gamma^{m\cdots}\theta)$, $(\lambda\gamma^{m\cdots}\chi)$ and $(\chi\gamma^{m\cdots}\theta)$, and not for example as $(\theta\gamma^{m\cdots}\lambda)$ etc.

When the computation involves fermionic particles, after processing the expressions with `PSordering()` there may be factors such as $(\chi^1\gamma_m\theta)(\chi^2\gamma^n\theta)$, which will be rewritten by the procedure `Fierz()` as

$$-\frac{1}{96}(\chi^1\gamma^m\gamma_{qrs}\gamma^n\chi^2)(\theta\gamma^{qrs}\theta)$$

again in order to minimize the number of pattern matching to identify correlators. This procedure takes care of various bilinears combinations, and the pair of fermions chosen to be expanded depend on the particular combination being considered, for which the user should read the source code for more information. The general expansions are done with the formula

$$\lambda^\alpha\chi^\beta = \frac{1}{16}(\lambda\gamma^m\chi)\gamma_m^{\alpha\beta} + \frac{1}{96}(\lambda\gamma^{mnp}\chi)\gamma_{mnp}^{\alpha\beta} + \frac{1}{3840}(\lambda\gamma^{mnpqr}\chi)\gamma_{mnpqr}^{\alpha\beta}. \tag{2.2}$$

If the procedure `Fierz()` takes effect, then there will be more gamma matrices to expand, like in the example above $(\chi^1\gamma^m\gamma_{qrs}\gamma^n\chi^2)$. If that is the case, then the extra calling of `gammaExpand()` and `deltaExpand()`, followed by `PSordering()` and `orderFermions()` will let the expressions ready for being identified.

Using the example of the one-loop kinematic factor, at this point one of the terms being dealt with by `PSS` is

```
- 1/512*la*ga(N1_?,N2_?,N3_?)*th*la*ga(N4_?,N5_?,N6_?)*th*
  la*ga(N7_?)*th*th*ga(N7_?,N8_?,N9_?)*th*F1(N8_?,N9_?)*
  F2(N1_?,N2_?)*F3(N4_?,N5_?)*F4(N3_?,N6_?)
```

which is

$$-\frac{1}{512}(\lambda\gamma^{n_1n_2n_3}\theta)(\lambda\gamma^{n_4n_5n_6}\theta)(\lambda\gamma^{n_7}\theta)(\theta\gamma^{n_7n_8n_9}\theta)F^1_{n_8n_9}F^2_{n_1n_2}F^3_{n_4n_5}F^4_{n_3n_6}. \tag{2.3}$$

The next step is to call the procedure which identifies the pure spinor correlator from a catalog of known tensors. This is done with `identifyCorrelators()`, after which the above term is given by

```
- 1/512*F1(N1_?,N2_?)*F2(N3_?,N4_?)*F3(N5_?,N6_?)*F4(N7_?,N8_?)*
  ps331(N3_?,N4_?,N7_?,N5_?,N6_?,N8_?,N9_?,N9_?,N1_?,N2_?)
```

That is, `PSS` identifies the pure spinor correlator $(\lambda\gamma^{n_1n_2n_3}\theta)(\lambda\gamma^{n_4n_5n_6}\theta)(\lambda\gamma^{n_7}\theta)(\theta\gamma^{n_7n_8n_9}\theta)$ with the tensor $\text{ps331}(n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_7,n_8,n_9)$. Looking at the appendix of [19], this tensor is expanded in terms of Kronecker deltas as

$$\langle(\lambda\gamma^{mnp}\theta)(\lambda\gamma^{qrs}\theta)(\lambda\gamma_t\theta)(\theta\gamma_{ijk}\theta)\rangle = \frac{1}{8400}\epsilon^{ijkmnpqrst}+ \tag{2.4}$$

$$+\frac{1}{140}\Big[\delta^{[m}_t\delta^n_{[i}\eta^{p][q}\delta^r_j\delta^{s]}_{k]} - \delta^{[q}_t\delta^r_{[i}\eta^{s][m}\delta^n_j\delta^{p]}_{k]}\Big] - \frac{1}{280}\Big[\eta_{t[i}\eta^{v[q}\delta^r_j\eta^{s][m}\delta^n_{k]}\delta^{p]}_v - \eta_{t[i}\eta^{v[m}\delta^n_j\eta^{p][q}\delta^r_{k]}\delta^{s]}_v\Big]$$

and this is one of the correlators included in the catalog `ps_tensors.h`. So the purpose of `identifyCorrelators()` is to transform an input containing correlators with $\langle\lambda^3\theta^5\rangle$ into an expression written in terms of tensors like ps331(m,n,p,q,r,s,t,u,v,x), which are later substituted by Kronecker deltas and epsilon tensors as in (2.4) using the catalog.

The list of correlators identified by `identifyCorrelators()` is

$$\langle(\lambda\gamma^{t_1\dots t_5}\lambda)(\lambda\gamma^m\theta)(\theta\gamma^{r_1\dots r_3}\theta)(\theta\gamma^{s_1\dots s_3}\theta)\rangle = \text{uind}(t_1,...,t_5,m,r_1,...,r_3,s_1,...,s_3)$$

$$\langle(\lambda\gamma^{t_1\dots t_5}\lambda)(\lambda\gamma^{m_1\dots m_3}\theta)(\theta\gamma^{r_1\dots r_3}\theta)(\theta\gamma^{s_1\dots s_3}\theta)\rangle = \text{tind}(t_1,...,t_5,m_1,...,m_3,r_1,...,r_3,s_1,...,s_3)$$

$$\langle(\lambda\gamma^{t_1\dots t_5}\lambda)(\lambda\gamma^{m_1\dots m_5}\theta)(\theta\gamma^{r_1\dots r_3}\theta)(\theta\gamma^{s_1\dots s_3}\theta)\rangle = \text{cind}(t_1,...,t_5,m_1,...,m_5,r_1,...,r_3,s_1,...,s_3)$$

$$\langle(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\lambda\gamma^p\theta)(\theta\gamma^{abc}\theta)\rangle = \text{ps111}(m,n,p,a,b,c)$$

$$\langle(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\lambda\gamma^p\theta)(\theta\gamma^{r_1\dots r_7}\theta)\rangle = \text{ps111eps}(m,n,p,r_1,...,r_7)$$

$$\langle(\lambda\gamma^{t_1\dots t_3}\theta)(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\theta\gamma^{r_1\dots r_3}\theta)\rangle = \text{ps311}(t_1,...,t_3,m,n,r_1,...,r_3)$$

$$\langle(\lambda\gamma^{t_1\dots t_3}\theta)(\lambda\gamma^m\theta)(\lambda\gamma^n\theta)(\theta\gamma^{r_1\dots r7}\theta)\rangle = \text{ps311eps}(t_1,...,t_3,m,n,r_1,...,r7)$$

$$\langle(\lambda\gamma^{m_1\dots m_3}\theta)(\lambda\gamma^{n_1\dots n_3}\theta)(\lambda\gamma^a\theta)(\theta\gamma^{r_1\dots r_3}\theta)\rangle = \text{ps331}(m_1,...,m_3,n_1,...,n_3,a,r_1,...,r_3)$$

$$\langle(\lambda\gamma^{m_1\dots m_3}\theta)(\lambda\gamma^{n_1\dots n_3}\theta)(\lambda\gamma^a\theta)(\theta\gamma^{r_1\dots r_7}\theta)\rangle = \text{ps331eps}(m_1,...,m_3,n_1,...,n_3,a,r_1,...,r_7)$$

$$\langle(\lambda\gamma^{t_1\dots t_3}\theta)(\lambda\gamma^{m_1\dots m_3}\theta)(\lambda\gamma^{n_1\dots n_3}\theta)(\theta\gamma^{r_1\dots r_3}\theta)\rangle = \text{ps333}(t_1,.,t_3,m_1,.,m_3,n_1,.,n_3,r_1,.,r_3)$$

$$\langle(\lambda\gamma^{t_1\dots t_3}\theta)(\lambda\gamma^{m_1\dots m_3}\theta)(\lambda\gamma^{n_1\dots n_3}\theta)(\theta\gamma^{r_1\dots r_7}\theta)\rangle = \text{ps333eps}(t_1,.,t_3,m_1,.,m_3,n_1,.,n_3,r_1,.,r_7)$$

$$\langle(\lambda\gamma^{t_1\dots t_5}\theta)(\lambda\gamma^m\theta)(\lambda\gamma^a\theta)(\theta\gamma^{r_1\dots r_3}\theta)\rangle = \text{ps511}(t_1,.,t_5,m,a,r_1,.,r_3)$$

$$\langle(\lambda\gamma^{t_1\dots t_5}\theta)(\lambda\gamma^m\theta)(\lambda\gamma^a\theta)(\theta\gamma^{r_1\dots r_7}\theta)\rangle = \text{ps511eps}(t_1,.,t_5,m,a,r_1,.,r_7)$$

$$\langle(\lambda\gamma^{t_1\dots t_5}\theta)(\lambda\gamma^{m_1\dots m_3}\theta)(\lambda\gamma^a\theta)(\theta\gamma^{r_1\dots r_3}\theta)\rangle = \text{ps531}(t_1,.,t_5,m_1,.,m_3,a,r_1,.,r_3)$$

$$\langle(\lambda\gamma^{t_1\dots t_5}\theta)(\lambda\gamma^{m_1\dots m_3}\theta)(\lambda\gamma^a\theta)(\theta\gamma^{r_1\dots r_7}\theta)\rangle = \text{ps531eps}(t_1,...,t_5,m_1,...,m_3,a,r_1,...,r_7)$$

$$\langle(\lambda\gamma^{t_1\ldots t_5}\theta)(\lambda\gamma^{m_1\ldots m_5}\theta)(\lambda\gamma^{a}\theta)(\theta\gamma^{r_1\ldots r_3}\theta)\rangle = \mathrm{ps551}(t_1,...,t_5,m_1,...,m_5,a,r_1,...,r_3)$$

$$\langle(\lambda\gamma^{t_1\ldots t_5}\theta)(\lambda\gamma^{m_1\ldots m_5}\theta)(\lambda\gamma^{a}\theta)(\theta\gamma^{r_1\ldots r_7}\theta)\rangle = \mathrm{ps551eps}(t_1,.,t_5,m_1,.,m_5,a,r_1,.,r_7)$$

$$\langle(\lambda\gamma^{t_1\ldots t_5}\theta)(\lambda\gamma^{m_1\ldots m_5}\theta)(\lambda\gamma^{n_1\ldots n_3}\theta)(\theta\gamma^{r_1\ldots r_3}\theta)\rangle = \mathrm{ps553}(t_1,.,t_5,m_1,.,m_5,n_1,.,n_3,r_1,.,r_3)$$

$$\langle(\lambda\gamma^{t_1\ldots t_5}\theta)(\lambda\gamma^{m_1\ldots m_3}\theta)(\lambda\gamma^{n_1\ldots n_3}\theta)(\theta\gamma^{r_1\ldots r_3}\theta)\rangle = \mathrm{ps533}(t_1,.,t_5,m_1,.,m_3,n_1,.,n_3,r_1,.,r_3)$$

It may happen that one particular computation requires a correlator not in the list. When that happens `PSS` automatically detects the missing correlator and prints it before exiting. For example

```
I do not identify a correlator for:
 + 1/983040*chi1*ga(N1_?,N2_?,N3_?,N4_?,N5_?,N6_?,N7_?)*chi2*
   la*ga(N1_?,N2_?,N3_?,N4_?,N5_?)*th*la*ga(k1,N6_?,N8_?)*th*
   la*ga(N9_?,N10_?,N11_?)*th*
   th*ga(N7_?,N8_?,N9_?,N12_?,N13_?,N14_?,N15_?)*th*
   F3(N10_?,N11_?)*F4(N12_?,N13_?)*F5(N14_?,N15_?)
Add it to the identifyCorrelators() procedure
```

In this case the missing correlator can be obtained from ps533() and its pattern matching added to the procedure `identifyCorrelators()` and its tensor representation added to the file `ps_tensors.h`.

After calling `identifyCorrelators()` the tensors above are substituted by their tensor representations. This is done by including the file `ps_tensors.h`, which contains their expansions in terms of generalized Kronecker deltas and Levi-Civita epsilons. The procedure `deltaExpand()` expands the generalized Kronecker deltas in terms of the antisymmetric combinations of the usual Kronecker delta. Gamma matrices with more than five indices are manipulated with `dualizeGammas()`, where for example the following identity is used

$$\gamma^{m_1\ldots m_7}_{\alpha\beta} = -\frac{1}{3!} i\epsilon_{m_1\ldots m_7 n_1\ldots n_3}\gamma^{n_1\ldots n_3}_{\alpha\beta}.$$

The cases where epsilon tensors are contracted with gamma matrices is also dealt with,

$$\epsilon^{m_1\ldots m_3 n_1\ldots n_7}(\chi^1\gamma_{n_1\ldots n_7}\chi^2) = 5040\, i(\chi^1\gamma^{m_1 m_2 m_3}\chi^2).$$

For completeness, the gamma matrix conventions [30] are such that[9]

$$\gamma^{m_1\ldots m_9}_{\alpha\beta} = i\,\epsilon^{m_1\ldots m_9 n_1}(\gamma_{n_1})_{\alpha\beta}, \quad (\gamma^{m_1\ldots m_8})_{\alpha}^{\ \beta} = \frac{1}{2!} i\,\epsilon^{m_1\ldots m_8 n_1\ldots n_2}(\gamma_{n_1\ldots n_2})_{\alpha}^{\ \beta}$$

[9] The signs change when both spinor indices of the matrix matrices change from Weyl to anti-Weyl.

$$\gamma_{\alpha\beta}^{m_1\ldots m_7} = -\frac{1}{3!} i \epsilon^{m_1\ldots m_7 n_1\ldots n_3} \gamma_{\alpha\beta}^{n_1\ldots n_3}, \quad (\gamma^{m_1\ldots m_6})_\alpha^{\ \beta} = \frac{1}{4!} i\, \epsilon^{m_1\ldots m_6 n_1\ldots n_4} (\gamma_{n_1\ldots n_4})_\alpha^{\ \beta},$$
$$\gamma_{\alpha\beta}^{m_1\ldots m_5} = \frac{1}{5!} i\, \epsilon^{m_1\ldots m_5 n_1\ldots n_5} (\gamma_{n_1\ldots n_5})_{\alpha\beta}, \quad \gamma_{\alpha\beta}^{m_1\ldots m_3} = -\frac{1}{7!} i\, \epsilon^{m_1\ldots m_3 n_1\ldots n_7} (\gamma_{n_1\ldots n_7})_{\alpha\beta} \tag{2.5}$$

It is important to notice that `FORM` uses the convention that $\epsilon_{m_1\ldots m_{10}} \epsilon^{m_1\ldots m_{10}} = 10!$ instead of $-10!$, so that is why there are factors of i together with epsilon tensors in `PSS`.

The procedure `fieldStrength()` substitutes $F_{mn}^I = k_m^I e_n^I - k_n^I e_m^I$ and `onShell()` annihilates terms with $(k^I \cdot k^I)$ and $(k^I \cdot e^I)$. If there are fermionic particles, the procedure `diracEquation()` uses the Dirac equation to reduce the rank of the gamma matrices when there is a momentum of one of the particles in the fermionic bilinear being contracted with one of its indices, for example

$$(\chi^1 \gamma_{mnk_2} \chi^2) = k_m^2 (\chi^1 \gamma_n \chi^2) - k_n^2 (\chi^1 \gamma_m \chi^2).$$

Finally, the procedure `momentumConservation()` applies the conservation of momentum to one of the labels, which can be manually chosen by setting the "dollar" variable `$kn` in the beginning of `pss.frm` (if let at its default value of zero, then an automatic choice is made).

After the above (simplified) sequence of steps the desired component expansion of the superspace expression is printed on the screen.

If one chooses the particles 1 and 2 to be fermionic by using `#define field1 1` etc, rerunning the program results in,

```
[mafra@Pilar:pss] tform -q -w2 pss.frm
   [4pts_one-loop] =
       + 1/11520*chi1*ga(k4,e3,e4)*chi2*u
       + 1/11520*chi1*ga(k4,e3,e4)*chi2*t
       - 1/11520*chi1*ga(k4)*chi2*e3.e4*u
       + 1/11520*chi1*ga(k4)*chi2*e3.e4*t
       + 1/5760*chi1*ga(e3)*chi2*k1.e4*u
       - 1/5760*chi1*ga(e3)*chi2*k2.e4*t
       + 1/11520*chi1*ga(e4)*chi2*k4.e3*u
       - 1/11520*chi1*ga(e4)*chi2*k4.e3*t
      ;
Momentum conservation: k3 eliminated
Gauge invariance: not tested
  0.08 sec + 0.42 sec: 0.50 sec out of 0.32 sec
```

2.3. Debug options

There are three pre-defined debug options which can be used to check intermediate steps in the computation, `sfexpand`, `psonly` and `nofierz`. If `sfexpand` is invoked (using the `-d` flag; see `FORM`'s manual), then only the superfield expansion in terms of θ's is printed,

```
[mafra@Pilar:pss] tform -q -w2 -d sfexpand pss.frm
[4pts_one-loop] =
   - 1/256*la*ga(e1)*th*la*ga(k4)*ga(m2,n2)*th*la*ga(n)*ga(m3,n3)*th*
     th*ga(n,m4,n4)*th*F2(m2,n2)*F3(m3,n3)*F4(m4,n4)
   - 1/384*la*ga(e1)*th*la*ga(m)*ga(k2,N1_?)*th*th*ga(m2,n2,N1_?)*th*
     la*ga(n)*ga(m3,n3)*th*F2(m2,n2)*F3(m3,n3)*F4(m,n)
   + 1/256*la*ga(e1)*th*la*ga(m)*ga(m2,n2)*th*la*ga(k4)*ga(m3,n3)*th*
     th*ga(m,m4,n4)*th*F2(m2,n2)*F3(m3,n3)*F4(m4,n4)
   + 1/384*la*ga(e1)*th*la*ga(m)*ga(m2,n2)*th*la*ga(n)*ga(k1,N1_?)*th*
     th*ga(m3,n3,N1_?)*th*F2(m2,n2)*F3(m3,n3)*F4(m,n)
   + 1/384*la*ga(e1)*th*la*ga(m)*ga(m2,n2)*th*la*ga(n)*ga(k2,N1_?)*th*
     th*ga(m3,n3,N1_?)*th*F2(m2,n2)*F3(m3,n3)*F4(m,n)
   + 1/384*la*ga(e1)*th*la*ga(m)*ga(m2,n2)*th*la*ga(n)*ga(k4,N1_?)*th*
     th*ga(m3,n3,N1_?)*th*F2(m2,n2)*F3(m3,n3)*F4(m,n)
   - 1/512*la*ga(N1_?)*th*th*ga(m1,n1,N1_?)*th*la*ga(m)*ga(m2,n2)*th*
     la*ga(n)*ga(m3,n3)*th*F1(m1,n1)*F2(m2,n2)*F3(m3,n3)*F4(m,n)
   ;
```

which can be useful to check when something goes wrong. The debug option `psonly` prints the expression after the correlators were identified, for example,

```
[mafra@Pilar:pss] tform -q -w2 -d psonly pss.frm
[4pts_one-loop] =
  - 1/128*F1(N1_?,N2_?)*F2(N3_?,N4_?)*F3(N5_?,N6_?)*F4(N3_?,N5_?)*
    ps111(N7_?,N4_?,N6_?,N7_?,N1_?,N2_?)
  - 1/256*F1(N1_?,N2_?)*F2(N3_?,N4_?)*F3(N5_?,N6_?)*F4(N3_?,N7_?)*
    ps311(N5_?,N6_?,N7_?,N8_?,N4_?,N8_?,N1_?,N2_?)
  - 1/256*F1(N1_?,N2_?)*F2(N3_?,N4_?)*F3(N5_?,N6_?)*F4(N5_?,N7_?)*
    ps311(N3_?,N4_?,N7_?,N8_?,N6_?,N8_?,N1_?,N2_?)
  - 1/512*F1(N1_?,N2_?)*F2(N3_?,N4_?)*F3(N5_?,N6_?)*F4(N7_?,N8_?)*
    ps331(N3_?,N4_?,N7_?,N5_?,N6_?,N8_?,N9_?,N9_?,N1_?,N2_?)
    ...
```

where the other terms are similar and were omitted. Furthermore, `nofierz` prints the superspace expansions before any Fierz manipulation is done,

```
[mafra@Pilar:pss] tform -q -w2 -d nofierz pss.frm
[4pts_one-loop] =
 + 1/24*chi1*ga(N1_?)*th*chi2*ga(N2_?)*th*la*ga(k2,N2_?,N3_?)*th*
   la*ga(N1_?)*th*la*ga(N4_?)*th*F3(N4_?,N5_?)*F4(N3_?,N5_?)
 + 1/48*chi1*ga(N1_?)*th*chi2*ga(N2_?)*th*la*ga(k2,N2_?,N3_?)*th*
   la*ga(N4_?,N5_?,N6_?)*th*la*ga(N1_?)*th*F3(N5_?,N6_?)*F4(N3_?,N4_?)
 + 1/24*chi1*ga(N1_?)*th*chi2*ga(N2_?)*th*la*ga(N1_?)*th*la*ga(k2)*th*
   la*ga(N3_?)*th*F3(N3_?,N4_?)*F4(N2_?,N4_?)
   ...
```

There are many possible extensions and optimizations which can be made to `PSS`, as it is available to download at `http://www.aei.mpg.de/~crmafra/pss.tar.gz` under the GPL license. In particular, dealing with four-fermion expansions is still not completely automated (nor guaranteed to be correct). It would be interesting to implement the fermionic methods described in [27] for this purpose. Furthermore, it should be straightforward to write procedures to translate the full ten-dimensional components to four dimensions using the spinor helicity formalism, in order to compare with the results appearing in [31]. The possibilities are many and it is hoped that `PSS` provides a framework for further work.

Acknowledgements: I want to thank Joost Hoogeveend for convincing me that `PSS` may be useful to other people and encouraging me to release it. I also thank Jos Vermaseren for his suggestions on how to deal with generalized Kronecker deltas. I acknowledge support by the Deutsch-Israelische Projektkooperation (DIP H52).

References

[1] N. Berkovits, "Super-Poincare covariant quantization of the superstring," JHEP **0004**, 018 (2000) [arXiv:hep-th/0001035].

[2] N. Berkovits, "ICTP lectures on covariant quantization of the superstring," arXiv:hep-th/0209059.

[3] O. A. Bedoya and N. Berkovits, "GGI Lectures on the Pure Spinor Formalism of the Superstring," arXiv:0910.2254 [hep-th].

[4] C. R. Mafra, "Superstring Scattering Amplitudes with the Pure Spinor Formalism," arXiv:0902.1552 [hep-th].

[5] N. Berkovits and B. C. Vallilo, "Consistency of super-Poincare covariant superstring tree amplitudes," JHEP **0007**, 015 (2000) [arXiv:hep-th/0004171].

[6] G. Policastro and D. Tsimpis, "R**4, purified," [arXiv:hep-th/0603165].

[7] C. R. Mafra, "Pure Spinor Superspace Identities for Massless Four-point Kinematic Factors," JHEP **0804**, 093 (2008) [arXiv:0801.0580 [hep-th]].

[8] C. R. Mafra, "Simplifying the Tree-level Superstring Massless Five-point Amplitude," JHEP **1001**, 007 (2010) [arXiv:0909.5206 [hep-th]].

[9] R. Medina, F. T. Brandt and F. R. Machado, "The open superstring 5-point amplitude revisited," JHEP **0207**, 071 (2002) [arXiv:hep-th/0208121]. ;
L. A. Barreiro and R. Medina, "5-field terms in the open superstring effective action," JHEP **0503**, 055 (2005) [arXiv:hep-th/0503182].

[10] Z. Bern, J. J. M. Carrasco and H. Johansson, "New Relations for Gauge-Theory Amplitudes," Phys. Rev. D **78**, 085011 (2008) [arXiv:0805.3993 [hep-ph]].

[11] N. E. J. Bjerrum-Bohr, P. H. Damgaard and P. Vanhove, "Minimal Basis for Gauge Theory Amplitudes," Phys. Rev. Lett. **103**, 161602 (2009) [arXiv:0907.1425 [hep-th]].

[12] S. Stieberger, "Open & Closed vs. Pure Open String Disk Amplitudes," arXiv:0907.2211

[13] N. E. J. Bjerrum-Bohr, P. H. Damgaard, T. Sondergaard and P. Vanhove, "Monodromy and Jacobi-like Relations for Color-Ordered Amplitudes," arXiv:1003.2403

[14] C. R. Mafra, "Towards Field Theory Amplitudes From the Cohomology of Pure Spinor Superspace," arXiv:1007.3639 [hep-th].

[15] N. Berkovits, "Multiloop amplitudes and vanishing theorems using the pure spinor formalism for the superstring," JHEP **0409**, 047 (2004) [arXiv:hep-th/0406055].

[16] N. Berkovits, "Pure spinor formalism as an N = 2 topological string," JHEP **0510**, 089 (2005) [arXiv:hep-th/0509120].

[17] C. R. Mafra, "Four-point one-loop amplitude computation in the pure spinor formalism," JHEP **0601**, 075 (2006) [arXiv:hep-th/0512052].

[18] C. R. Mafra and C. Stahn, "The One-loop Open Superstring Massless Five-point Amplitude with the Non-Minimal Pure Spinor Formalism," JHEP **0903**, 126 (2009) [arXiv:0902.1539 [hep-th]].

[19] N. Berkovits and C. R. Mafra, "Some superstring amplitude computations with the non-minimal pure spinor formalism," JHEP **0611**, 079 (2006) [arXiv:hep-th/0607187].

[20] N. Berkovits, "Super-Poincare covariant two-loop superstring amplitudes," JHEP **0601**, 005 (2006) [arXiv:hep-th/0503197].

[21] N. Berkovits and C. R. Mafra, "Equivalence of two-loop superstring amplitudes in the pure spinor and RNS formalisms," Phys. Rev. Lett. **96**, 011602 (2006) [arXiv:hep-th/0509234].

[22] H. Gomez, "One-loop Superstring Amplitude From Integrals on Pure Spinors Space," JHEP **0912**, 034 (2009) [arXiv:0910.3405 [hep-th]].

[23] H. Gomez and C. R. Mafra, "The Overall Coefficient of the Two-loop Superstring Amplitude Using Pure Spinors," JHEP **1005**, 017 (2010) [arXiv:1003.0678 [hep-th]].

[24] E. D'Hoker, M. Gutperle and D. H. Phong, "Two-loop superstrings and S-duality," Nucl. Phys. B **722**, 81 (2005) [arXiv:hep-th/0503180].

[25] E. Witten, "Twistor - Like Transform In Ten-Dimensions," Nucl. Phys. B **266**, 245 (1986).

[26] J. P. Harnad and S. Shnider, "Constraints And Field Equations For Ten-Dimensional Superyang-Mills Theory," Commun. Math. Phys. **106**, 183 (1986). ;
P. A. Grassi and L. Tamassia, "Vertex operators for closed superstrings," JHEP **0407**, 071 (2004) [arXiv:hep-th/0405072].

[27] C. Stahn, "Fermionic superstring loop amplitudes in the pure spinor formalism," JHEP **0705**, 034 (2007) [arXiv:0704.0015 [hep-th]].

[28] U. Gran, "GAMMA: A Mathematica package for performing Gamma-matrix algebra and Fierz transformations in arbitrary dimensions," arXiv:hep-th/0105086.

[29] J. A. M. Vermaseren, "New features of FORM," [arXiv:math-ph/0010025] ;
M. Tentyukov and J. A. M. Vermaseren, "The multithreaded version of FORM," [arXiv:hep-ph/0702279].

[30] P. Mukhopadhyay, "On D-brane boundary state analysis in pure-spinor formalism," JHEP **0603**, 066 (2006) [arXiv:hep-th/0505157].

[31] S. Stieberger and T. R. Taylor, "Amplitude for N-gluon superstring scattering," Phys. Rev. Lett. **97**, 211601 (2006) [arXiv:hep-th/0607184]. ;
S. Stieberger and T. R. Taylor, "Multi-gluon scattering in open superstring theory," Phys. Rev. D **74**, 126007 (2006) [arXiv:hep-th/0609175]. ;
S. Stieberger and T. R. Taylor, "Supersymmetry Relations and MHV Amplitudes in Superstring Theory," Nucl. Phys. B **793**, 83 (2008) [arXiv:0708.0574 [hep-th]]. ;
S. Stieberger and T. R. Taylor, "Complete Six-Gluon Disk Amplitude in Superstring Theory," Nucl. Phys. B **801**, 128 (2008) [arXiv:0711.4354 [hep-th]].