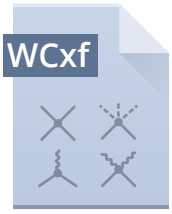
IFIC/17-61
KA-TP-38-2017
TUM-HEP-1117/17
LMU-ASC 74/17



`WCxf`: an exchange format for Wilson coefficients beyond the Standard Model

Jason Aebischer[a], Ilaria Brivio[b], Alejandro Celis[c], Jared A. Evans[d], Yun Jiang[b], Jacky Kumar[l], Xuanyou Pan[a], Werner Porod[e], Janusz Rosiek[f], David Shih[g], Florian Staub[h,i], David M. Straub[a], Danny van Dyk[j], Avelino Vicente[k]

[a] *Excellence Cluster Universe, TUM, Boltzmannstr. 2, 85748 Garching, Germany*
[b] *Niels Bohr International Academy and Discovery Center, Niels Bohr Institutet, Københavns Universitet, Blegdamsvej 17, 2100 København Ø, Denmark*
[c] *Ludwig-Maximilians-Universität München, Fakultät für Physik, Arnold Sommerfeld Center for Theoretical Physics, 80333 München, Germany*
[d] *Department of Physics, University of Cincinnati, Cincinnati, Ohio 45221, USA*
[e] *Institut für Theoretische Physik und Astrophysik, Universität Würzburg, 97074 Würzburg, Germany*
[f] *Faculty of Physics, University of Warsaw, Pasteura 5, 02-093 Warsaw, Poland*
[g] *Department of Physics, Rutgers University, Piscataway, New Jersey 08854, USA*
[h] *ITP, Karlsruhe Institute of Technology, Engesserstraße 7, 76128 Karlsruhe, Germany*
[i] *IKP, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Pl. 1, 76344 Eggenstein-Leopoldshafen, Germany*
[j] *Physik Department, TU München, James-Franck-Straße 1, 85748 Garching, Germany*
[k] *Instituto de Física Corpuscular, Universitat de València - CSIC, 46071 València, Spain*
[l] *Harish-Chandra Research Institute, Chhatnag Road, Jhusi, Allahabad 211019, India*

Abstract

We define a data exchange format for numerical values of Wilson coefficients of local operators parameterising low-energy effects of physics beyond the Standard Model. The format facilitates interfacing model-specific Wilson coefficient calculators, renormalisation group (RG) runners, and observable calculators. It is designed to be unambiguous (defining a non-redundant set of operators with fixed normalisation in each basis), extensible (allowing the addition of new EFTs or bases by the user), and robust (being based on industry standard file formats with parsers implemented in many programming languages). We have implemented the format for the Standard Model EFT (SMEFT) and for the weak effective theory (WET) below the electroweak scale and have added interfaces to a number of public codes dealing with SMEFT or WET. We also provide command-line utilities and a Python module for convenient manipulation of `WCxf` files, including translation between different bases and matching from SMEFT to WET.

1. Introduction and Motivation

Indirect effects of physics beyond the Standard Model (SM) at energies much lower than the mass scale of the new particles can be described by effective field theories (EFTs) extending the SM by new local operators. EFT methods can be used either as a means to perform model-independent analyses of new physics (NP), or merely as a tool to separate the model-independent low-energy phenomenology from the model-dependent short-distance physics when studying specific NP models.

Depending on the problem, different EFTs are appropriate. For very low-energy processes like flavour physics, the renormalisable part of the theory only contains QCD and QED interactions and light quarks as well as leptons. For electroweak-scale energies, the Standard Model Effective Field Theory (SMEFT) is appropriate for theories with a linearly realised breaking of the electroweak symmetry. Other EFTs relevant for physics beyond the SM include the "Higgs EFT" (HEFT) extending the SM with a non-linear realisation of electroweak symmetry (see e.g. [1, 2] for reviews on SMEFT and HEFT), or EFTs with new light degrees of freedom – e.g. sterile neutrinos or light dark matter particles. In a phenomenological analysis, typically the Wilson coefficients are predicted by a NP model at some high scale. Before predicting experimental observables, the Wilson coefficients have to be run down to low energies using the renormalisation group (RG), and if necessary they have to be matched onto a different EFT appropriate at lower energies.

Although several public codes exist dealing with various aspects of this problem, a practical hurdle is the exchange of numerical values of Wilson coefficients between them. The challenges include

- the generally very large number of operators[1],
- the many different bases used in the literature, often geared for a specific problem,
- ambiguities related to redundant operators or different normalisations,
- different programming languages used and different quality of parsers for program-specific file formats.

The `WCxf` is an attempt at overcoming these challenges by defining a common file format that is

- *unambiguous*, by uniquely fixing the set of non-redundant operators and their normalisation in a given basis of a given EFT,
- *extensible*, by allowing the user to define new EFTs or new bases for existing EFTs,
- *robust* by using industry standard file formats with parsers implemented in many programming languages.

[1] The Flavour Les Houches Accord (FLHA) [3] is a file format for (among other things) the exchange of numerical Wilson coefficients relevant for flavour physics that solves this problem by defining a numeric code representing any given operator. While FLHA aims to be a very general exchange format for flavour physics codes using the WET, the scope of `WCxf` is different by focusing only on new physics Wilson coefficients, but also allowing different EFTs and multiple bases. `WCxf` Wilson coefficient files in the WET can in principle be converted to FLHA format (but not vice versa) and we envisage an implementation of this in the future.

Crucially, the format does not fix a single basis that has to be used by all codes – acknowledging the fact that different bases can be more convenient for different problems – but rather provides a way to *define* a basis (or EFT) by uploading a definition file to a public repository. The translation between different bases[2] (as well as the matching between different EFTs) can then be performed by a single tool, i.e. in principle has to be implemented only once, while individual codes only have to export or import data using their internal EFT and basis. Robustness is achieved by using the well-established `YAML` and `JSON` formats.

The rest of this document is organised as follows. In section 2, we define the `WCxf` format. In section 3, we discuss the EFTs and bases that we have defined so far. Section 4 describes the command-line tool and Python package that we provide for the basis translation, matching, and validation of `WCxf` files. Section 5 contains brief descriptions of the implementation of a `WCxf` interface in several existing public codes. Section 6 contains our conclusions and outlook.

2. Definition of the format

The `WCxf` defines three different types of data files:

- The **EFT file**, fixing the theory the Wilson coefficients are defined in;
- The **basis file**, defining the basis used and listing all operators defined in the basis;
- The **Wilson coefficient file**, containing the actual numerical values of the Wilson coefficients.

The EFT and basis files are meant to be immutable definitions deposited in a public repository, whereas the Wilson coefficient file contains the actual data to be exchanged between different programs.

The data structure of these files consists of key-value pairs and lists, but allows for different *metaformats* to ship this data. The two recommended formats are

- `JSON`, which can be imported and exported without third-party tools in Python and Mathematica and is very fast to parse due to its rigorous format,
- `YAML`, which is a superset of `JSON` (i.e. `JSON` files can be parsed by a `YAML` parser but not vice versa), but tends to be more human readable/editable and allows the use of comments.

In principle, `WCxf` is not necessarily a *file* format but can also be used to directly exchange data structures between programs, bypassing the file system (e.g., in the case of Python this could be a `wcxf.WC` instance discussed in section 4.2 or simply a dictionary). The rest of this document will use the `YAML` format in all examples.

2.1. The EFT file

The EFT file only defines the name of the EFT and lists the named **sectors**, which are sets of operators with definite quantum numbers under global symmetries preserved by the RG evolution. In the case of SMEFT, the RGEs preserve baryon number B and lepton number L separately, so

[2] For translating between different bases, the approach is similar to the Rosetta SMEFT translator [4], but not restricted to SMEFT.

the sectors correspond to transitions with fixed ΔB and ΔL. At dimension 5, there is a single operator with $\Delta L = 2$; at dimension 6, there are B- and L-conserving operators as well as operators with $\Delta B = \Delta L = 1$. A minimal EFT file could thus look like

```yaml
eft: SMEFT
sectors:
  dB=dL=0:
  dB=dL=1:
  dL=2:
```

In the weak effective theory, flavour quantum numbers are conserved by the QED and QCD RG evolution, such that there are several sectors with definite flavour quantum numbers, e.g. $\Delta S = 2$ operators contributing to neutral kaon mixing.

The rationale of defining sectors is to simplify basis translations and RG evolution, as they can be performed sector by sector.

2.2. The basis file

The basis file contains all the names of the operators grouped by sectors as defined above. Operators are not allowed to have flavour indices; i.e., Wilson coefficients are always scalars. The basis must not contain redundant operators, but does not necessarily have to be complete (partial bases may be useful for specific physical problems or for observable calculators only considering processes sensitive to a subset of all operators). A file minimally defining the "Warsaw basis" of SMEFT could look like

```yaml
name: Warsaw
eft: SMEFT
sectors:
  dB=dL=0:
    G:
      real: true
    Gtilde:
      real: true
    W:
      real: true
    [...]
    uphi_11:
    uphi_12:
    uphi_13:
    [...]
```

The flavour indices (`11` etc.) are part of the name. The rationale for not allowing matrix- or tensor-valued Wilson coefficients is that this would make it very difficult to define non-redundant bases. Unless otherwise specified, Wilson coefficients are assumed to be complex-valued. For Hermitian operators, only allowing real-valued Wilson coefficients, the key `real` should be added and set to `true`.

As a general convention, the Wilson coefficients are defined as coefficients of operators in the effective Lagrangian

$$\mathcal{L}_{\text{eff}} = -\mathcal{H}_{\text{eff}} = \sum_{O_i = O_i^\dagger} C_i\, O_i + \sum_{O_i \neq O_i^\dagger} \left(C_i\, O_i + C_i^*\, O_i^\dagger \right), \tag{1}$$

i.e., the Hermitian conjugate is added only for non-Hermitian operators. When prefactors are factored out of the Wilson coefficients (e.g. the Fermi constant or CKM elements), these should be understood as part of the operator.

To define bases that are just a minor modification of an existing basis, basis files can use *inheritance* by specifying a parent basis name in the top-level `parent` key. All sectors omitted from these child basis files are assumed to be the same[3] as for the parent basis.

2.3. The Wilson coefficient file

While the EFT and basis files only have to be defined once (and are already pre-defined for a number of standard EFTs and bases), the Wilson coefficient file contains the actual numerical data to be exchanged between different codes. Minimally, it just defines the EFT, the basis, the renormalisation scale, and lists the numerical values of the coefficients. Unless otherwise specified, Wilson coefficients are assumed to be renormalised in the $\overline{\text{MS}}$ scheme. Dimensionful coefficients must be given in appropriate powers of GeV. Any coefficient defined in the basis but not present in the data file is assumed to vanish. If the value is a number, the Wilson coefficient is assumed to be real. If it is complex, it must be given as a mapping with keys `Re` and `Im`. Dimensionful numbers have to be given in units of GeV to the appropriate power. A minimal example could look like

```yaml
eft: SMEFT
basis: Warsaw
scale: 1e16
values:
    Gtilde: 3.1e-6
    uphi11:
      Re: 0
      Im: 0.0001
```

2.4. Metadata

While the minimal examples above are sufficient for machine-readable data exchange, the `WCxf` format also allows for the addition of metadata. Metadata can contain textual descriptions of EFTs or bases, information about the software used to generate the file, or LaTeX code that can be used for a more readable representation of the basis.

All three file types allow for a top-level metadata key that can contain arbitrary key-value pairs below it. Recommended metadata keys include

- `description` in the case of EFT and basis files, containing a textual description of the basis and its characteristics,

[3] The *name* and *form* of the operators in a sector omitted from a child basis definition is assumed to be the same as in the parent basis, but not necessarily the *numerical values*: for instance, a child basis could inherit all operators from a parent basis, but differ due to a rephasing of fermion fields.

- `generator` in the case of the Wilson coefficient file, specifying the software that generated the file.

The basis file, defining the non-redundant set of operators in a given basis, can additionally contain metadata for each operator. This is facilitated by the definition in the form of a mapping key. For instance, the `tex` key can be used to define the LaTeX representation of an operator:

```
    [...]
    H:
      tex: (H^\dagger H)^3
```

As a convention, math mode delimiters are omitted.

3. Pre-defined EFTs and bases

In this section we describe the EFTs that we have already defined, namely SMEFT and WET, and the *complete* bases defined therein. In section 5, we will discuss more specific bases for WET appropriate for individual codes.

3.1. SMEFT

At dimension 5, SMEFT contains a single operator: the lepton number violating Weinberg operator generating Majorana masses for the neutrinos [5]. At dimension 6, it contains baryon- and lepton number conserving operators as well as operators with $\Delta B = \Delta L = 1$ [6, 7]. At dimension 7, there are additional operators with $\Delta L = 2$ as well as with $\Delta B = -\Delta L = 1$ [8, 9]. We limit ourselves to dimension-6 operators for the time being.

The first non-redundant basis for the $\Delta B = \Delta L = 0$ operators at dimension 6 was derived in [7] and for baryon number violating ones in [10]. This basis is often called the "Warsaw basis". The counting of all non-redundant operators in the presence of three fermion generations was first performed in [11, 12]. Recently, a specific choice of a complete set of non-redundant operators was suggested as part of the `DsixTools` package [13]. We adhere to this choice in our basis file for the Warsaw basis.

A subtlety of choosing a basis for flavoured SMEFT operators is the choice of the weak basis in the space of the three fermion generations for each of the quark and lepton fields. Since all fermions are massless in the $SU(2)_L \times U(1)_Y$-symmetric phase, there is no a priori preferred basis, such as the mass basis in the weak effective theory below the electroweak scale. Concretely, the theory is invariant under a $U(3)^5$ rotation,

$$\psi \to U_\psi \psi, \qquad \psi = q, u, d, l, e\,. \tag{2}$$

While specifying the complete Lagrangian including the Yukawa matrices along with the Wilson coefficients would fix this basis, this conflicts with the specification of `WCxf` as being a Wilson coefficient-only data exchange format. Instead, our choice for the Warsaw basis is to define a default weak basis in which the fermion mass matrices have a specific form. The running fermion mass matrices at the dimension-6 level, after a general $U(3)^5$ rotation, can be written as (see e.g. [14])

$$M_d = \frac{v}{\sqrt{2}} U_q^\dagger \left(Y_d - \frac{v^2}{2} C_{d\phi} \right) U_d\,, \tag{3}$$

$$M_u = \frac{v}{\sqrt{2}} U_q^\dagger \left(Y_u - \frac{v^2}{2} C_{u\phi} \right) U_u \,, \tag{4}$$

$$M_e = \frac{v}{\sqrt{2}} U_l^\dagger \left(Y_e - \frac{v^2}{2} C_{e\phi} \right) U_e \,, \tag{5}$$

$$M_\nu = -v^2 \, U_l^T \, C_{ll\phi\phi} \, U_l \,. \tag{6}$$

We use the freedom of $U(3)^5$ field rotations to choose a basis where

- M_d and M_e are diagonal with real positive ascending entries,
- M_u has the form $V^\dagger \hat{M}_u$, where $\hat{M}_u$ is diagonal with real positive ascending entries and V has the form of the CKM matrix in standard phase convention.
- M_ν has the form $U^* \hat{M}_\nu U^\dagger$, where $\hat{M}_\nu$ is diagonal with real positive ascending[4] entries and U has the form of the PMNS matrix in standard phase convention.

This basis choice is convenient as it removes all unphysical parameters present in the Yukawa couplings and it allows to easily translate to the mass basis at the scale of electroweak symmetry breaking. However, we stress that this diagonality is not invariant under SMEFT renormalisation group evolution and only holds at a single scale.

As a variant of the default "Warsaw" basis, we also define a basis denoted `Warsaw up` that uses a weak basis where the up-type quark mass matrix M_u rather than M_d is diagonalised, while M_d has the form $V \hat{M}_d$.

Finally, we also define a "Warsaw mass" basis coinciding with the basis choice in [14] and the $\widetilde{C}$ basis in [15]. Due to our weak basis choice in the Warsaw basis, this basis differs from the "Warsaw" basis only by a rotation by the CKM matrix in the operators O_{uX}, with $X = \phi, G, W$, or B, and by the fact that the neutrino mass operator $O_{ll\phi\phi}$ is diagonal in this basis.

3.2. Weak effective theory

The effective theory of the SM below the electroweak scale, in the phase where the electroweak symmetry is broken, is often called "weak effective theory" (WET) as it represents the appropriate EFT for describing weak interactions of leptons and quarks.[5] The dynamical fields in WET contain: all leptons; all quarks except the top; the photon; and the gluon. Since QED and QCD conserve flavour quantum numbers, the "sectors" are simply collections of operators with fixed flavour quantum numbers that do not mix under RG evolution. A complete, non-redundant basis and the full 1-loop QED and QCD RG equations for *flavour-changing operators* were recently presented in [17]. The complete basis including flavour-conserving operators and the one-loop anomalous dimension matrix was derived in [16, 18]. We provide basis files for both these bases, that we dub "Bern" and "JMS" bases, respectively. For the JMS basis, our basis file defines a complete set of non-redundant operators. We have checked that the number of non-redundant elements coincides with the counting in [16].

[4] Our definition of $\hat{M}_\nu$ having *ascending* masses on the diagonal means that the matrix U, while defined as having the standard phase convention of the PMNS matrix, only coincides with the PMNS matrix for a normally ordered neutrino mass spectrum. In the case of inverse ordering, $m_{\nu_3} < m_{\nu_1} < m_{\nu_2}$, the matrix U corresponds to a PMNS-like matrix with permuted angles.

[5] Alternatively, the WET was dubbed LEFT (low-energy EFT) in [16].

EFT	quarks	leptons
`WET`	u, d, s, c, b	e, μ, τ
`WET-4`	u, d, s, c	e, μ, τ
`WET-3`	u, d, s	e, μ
`WET-2`	u, d	e

Table 1: Four EFTs below the electroweak scale and the dynamical quark and lepton fields they contain.

For processes below the b-quark scale, EFTs with a smaller number of dynamical quark (and lepton) fields are appropriate. We define four different EFTs with names and dynamical fields listed in table 1.

Additional WET bases have been defined for specific codes as described in section 5.

4. Python and command-line interface

To facilitate the basis translation and matching of Wilson coefficients we provide a Python package `wcxf` including a command-line tool that can be used by different codes or on its own. Having this central tool allows in particular to define translation and matching functions only once and make them available to different programs. We have already implemented the matching from SMEFT (in the Warsaw basis) to the WET (in the JMS basis) as well as translations between most of the bases discussed here. The tool also allows validating the format of `WCxf` files. Here we briefly describe its main features and refer to the project web site [19] for detailed documentation.

4.1. Installation

The Python package and command line interface require Python version 3.5 or above as well as the `NumPy` package. With these prerequisites, they can be installed by the command[6]

```
python3 -m pip install wcxf --user
```

This will also download the EFT and basis files from the public repository. When a new version is available, the package can be upgraded with

```
python3 -m pip install --upgrade wcxf --user
```

A development version of `WCxf` can also be installed directly from a Git working directory using

```
python3 -m pip install --user -e .
```

[6]The name of the Python 3 executable might differ depending on the system.

4.2. Python package

The `wcxf` package provides three classes representing EFT, basis, and Wilson coefficient files, aptly named `wcxf.EFT`, `wcxf.Basis`, and `wcxf.WC`. All three classes provide a `load` method to load `YAML` or `JSON` files as well as a `validate` method that performs various checks on the file and raises an exception if a problem is found. For instance, a Wilson coefficient file can be read and validated via

```
import wcxf

with open('my_wcxf_input_file.yml', 'r') as f:
  wc = wcxf.WC.load(f)
wc.validate()
```

The `WC` class also has a `dict` property that returns the numerical values of all Wilson coefficients as a dictionary, already representing complex coefficients by complex numbers rather than a dictionary of real and imaginary parts.

All EFTs and bases present in the public repository are part of the package and are automatically read in at import time. Existing EFTs and bases can be accessed by their names as follows,

```
smeft = wcxf.EFT['SMEFT']
warsaw_basis = wcxf.Basis['SMEFT', 'Warsaw'] # eft, basis
```

Translators between different bases of the same EFT can be defined by writing a function accepting a dictionary of Wilson coefficient name-value pairs in the source basis, and returning a corresponding dictionary in the target basis. The translator can then be made known to the package simply by decorating it with the `translator` decorator,

```
@wcxf.translator('My source basis', 'My target basis')
def my_translation_function(source_dict):
  # ...
  return target_dict
```

Matchers from one EFT to another are defined analogously by a `matcher` decorator, where both the EFT and the basis of the source and target dictionaries have to be specified,

```
@wcxf.matcher('My UV EFT', 'My UV EFT basis', 'My IR EFT', 'My IR EFT basis')
def my_matching_function(source_dict):
  # ...
  return target_dict
```

If the appropriate translator or matcher exists, a Wilson coefficient instance in the new basis or EFT can be generated by calling the `translate` or `match` methods with the source basis (or EFT and basis) as arguments,

```
wc_new = wc_old.translate('My target basis')
wc_new = wc_old.match('My IR EFT', 'My IR EFT basis')
```

4.3. *Command-line interface*

The validation, translation, and matching functionality of the Python package are also available through a command-line script that is automatically installed along with the package. In addition, the script can also convert `YAML` files to `JSON` and vice versa. The script is invoked as[7]

```
wcxf <command> [<options>]
```

where `<command>` can be

- `convert` for converting between the `JSON` and `YAML` formats,
- `validate` for validating that a file adheres to the standard,
- `translate` for translating between different bases,
- `match` for matching between different EFTs.

The available options and arguments can be displayed by calling

```
wcxf <command> -h
```

Here we just list a few common examples,

```
wcxf convert json my_file.yml --output my_file.json      # YAML -> JSON conversion
wcxf validate basis my_basis.yml                         # Basis validation
wcxf validate wc my_coeffs.yml                           # Wilson c. validation
wcxf translate flavio wc_jms.yml --output wc_flavio.json # Basis translation
wcxf match wet jms wc_warsaw.json --output wc_jms.json   # SMEFT -> WET matching
```

5. Implementation in public codes

5.1. *`flavio`*

`flavio` [20] is a Python package for flavour physics phenomenology in the SM and beyond. The features include making predictions for a host of flavour observables in the presence of new physics parameterised by Wilson coefficients of dimension-6 operators in the WET, and fitting Wilson coefficients to experimental data using Bayesian or frequentist methods. The interface for setting values of new physics Wilson coefficients in `flavio` is the `WilsonCoefficient` class. From version 0.25, this class supports loading the initial values of the Wilson coefficients from a `WCxf` file, making use of the `wcxf-python` package (cf. section 4.2). A simple example how to read the Wilson coefficients from a file:

[7] If the command is not found even though the package has been installed, on Linux you might have to add `$HOME/.local/bin` to your `$PATH`.

```python
import flavio, wcxf

with open('my_wcxf_input_file.yml', 'r') as f:
  wc = wcxf.WC.load(f)

fwc = flavio.WilsonCoefficients()
fwc.set_initial_wcxf(wc)
```

The initial values are set at the scale specified in the `WCxf` file; `flavio` takes care of the RG evolution to the scale relevant for each process automatically. The EFT of the input must be WET, but the basis is arbitrary, as long as a translator to the `flavio` basis is defined in the `wcxf-python` package or the user's script.

5.2. EOS

The `EOS` software fulfills two use cases: First, computation of flavour physics observables within the SM or the WET; and second, inference of parameters based on experimental constraints in a Bayesian framework. `EOS` provides command-line clients for the most common tasks, as well as library interfaces in both C++ and Python; see [21] for an introduction and user manual.

The command-line script `wcxf2eos`, that is shipped with the `wcxf` Python package, converts a `WCxf` file to a `EOS` parameter `YAML` file

```
wcxf2eos my_wcxf_input_file.yml --output eos.yaml
```

The use of this parameter file is most convenient when examining a single point in the WET parameter space. For examining more than a few points, we recommend using the `EOS` Python interface instead:

```python
import eos, wcxf

with open ('my_wcxf_input_file.yml', 'r') as f:
    wc = wcxf.WC.load(f)

parameters = eos.Parameters.FromWCxf(wc)
```

At the time of writing this document, `EOS` implements $b \to s\{qq, \ell\ell, \gamma, G\}$ as well as $b \to u\ell\nu$ transitions in a variety of exclusive and inclusive decay modes. For $b \to s\ell\ell$ transitions, violation of lepton flavour universality is implemented. For $b \to u\ell\nu$ transitions, Wilson coefficients with $\ell = e$ and $\ell = \mu$ must be equal, otherwise an error is raised.

5.3. FlavorKit

`FlavorKit` [22, 23] is an extension of the `Mathematica` package `SARAH` [24–28] that increases its capability to handle flavour observables. Since it is based on `SARAH`, `FlavorKit` is not restricted to a specific model, but can be used to obtain analytical and numerical predictions for quark and lepton flavour observables in a wide range of models. `FlavorKit` adds to the `SPheno` [29, 30] interface of `SARAH` all necessary routines for the numerical calculation of Wilson coefficients at the full one-loop level in a given theory. In previous versions of `FlavorKit`, the numerical values of

these Wilson coefficients were written by SPheno in a standard output file in FLHA format. With version 4.12.3 of SARAH, the support of the WCxf format has also been added.

SPheno will export the numerical values of all calculated Wilson coefficients into JSON output files following the WCxf format if the flag

```
Block SPhenoInput #
...
79 1    # Write WCXF files (exchange format for Wilson coefficients)
```

is set in the Les Houches input file. The generated files are called WC.$MODEL_$X.json, where $MODEL is the name of the considered model and $X counts the scales for which the coefficients are written out. SPheno computes the Wilson coefficients at two different scales: the coefficients for quark flavour violating operators are calculated at $Q = 160$ GeV whereas those for lepton flavour violating operators are calculated at $Q = 91$ GeV.[8] Therefore, two JSON files are written by SPheno:

- WC.$MODEL_1.json: this contains the Wilson coefficients for quark flavour violating operators calculated at $Q = 160$ GeV
- WC.$MODEL_2.json: this contains the Wilson coefficients for lepton flavour violating operators calculated at $Q = 91$ GeV

These output files contain the numerical results for the Wilson coefficients in the FlavorKit basis [22]. This WET basis consists of a non-redundant set of operators for quark and lepton flavour physics. In contrast to the JMS basis [16], which is very similar, the FlavorKit basis is not complete and can be further extended with the addition of new operators. This can be done by means of the PreSARAH package, which interfaces FlavorKit with FeynArts/FormCalc [31–36] to fully automate the one-loop calculation of the Wilson coefficients. We refer to the FlavorKit manual [22] for more details. Once the new operators are added with PreSARAH, FlavorKit must be adapted to generate output WCxf files containing the new Wilson coefficients. This is achieved by modifying the file

SARAH-X.Y.Z/FlavorKit/WCXF_WilsonCoefficients.m

This modification must be applied before generating the SPheno code for the model of interest. The standard format of the WCXF_WilsonCoefficients.m file is the following:

[8] The internal classification of the effective operators into quark and lepton flavour violating is explicitly given in Appendix A of the FlavorKit manual [22]. We note that some of the quark flavour violating operators violate lepton flavour as well.

```
WCXF`Outputs = 2;

(* Output for QFV at Q=160 GeV *)
WCXF`EFT[1]= "WET";
WCXF`Basis[1]= "FlavorKit";
WCXF`Scale[1]= 160.0;

WCXF`Values[1]={

(* SECTOR: SBSB *)
(* 4D *)
{{"DVLL_2323",O4dVLL[2,3,2,3], Complex}},

....

(* SECTOR: UBENU *)
(* UDENU *)
Table[{"GVLL_"<>ToString[3]<>ToString[1]<>ToString[i]<>ToString[1],OdulvVLL[3,1,i
    ,1], Complex},{i,1,3}],

....

};
WCXF`Values[1]=Flatten[WCXF`Values[1],1];

(* Output for LFV at Q=91 GeV *)
WCXF`EFT[2]= "WET";
WCXF`Basis[2]= "FlavorKit";
WCXF`Scale[2]= 91.0;

WCXF`Values[2]={

....

};
WCXF`Values[2]=Flatten[WCXF`Values[2],1];
```

First, `WCXF`Outputs` is used to define the number of energy scales for which an output `JSON` file shall be written. For each of these scales, the EFT (`WCXF`EFT`), the basis (`WCXF`Basis`) and the value of energy scale in GeV (`WCXF`Scale`) are set. Finally, the Wilson coefficients in this basis are related to the coefficients internally calculated by `FlavorKit`. The general syntax for each coefficient is:

```
{Name WCXF, Name FlavorKit, Complex or Real}
```

One should note that while the name of the coefficient used in the `WCxf` files is a string, the `FlavorKit` name is a `Mathematica` symbol. Furthermore, the conventions for the `FlavorKit` name described in Appendices A and B of [22] must be taken into account. For instance, in the example given above, the first Wilson coefficient corresponds to the operator $(\bar{b}\gamma^\mu P_L s)(\bar{b}\gamma_\mu P_L s)$. Since the coefficients are usually defined for all three generations of fermions, it might be helpful to use `Mathematica` functions like `Table` to define them in a compact form. This is also shown in the

example, where the Wilson coefficients of the $(\bar{u}\gamma^\mu P_L b)(\bar{e}\gamma_\mu P_L \nu_i)$ operators, with $i = 1, 2, 3$, are also defined.

5.4. SPheno

The stand-alone `SPheno` package [29, 30] has included the `WCxf` format starting with version 4.1.0 in a similar manner as the `FlavorKit` package [22, 23] described above. `SPheno` will export the numerical values of all calculated Wilson coefficients into `JSON` output files following the `WCxf` format if the flag

```
Block SPhenoInput #
...
79 1    # Write WCXF files (exchange format for Wilson coefficients)
```

is set in the Les Houches input file. The generated files are called `WC.SPheno_$X.json`, where `$X` counts the scales for which the coefficients are written out. `SPheno` computes the Wilson coefficients at two different scales: the coefficients for quark flavour violating operators are calculated at $Q = 160$ GeV whereas those for lepton flavour violating operators are calculated at $Q = m_Z$. Therefore, two `JSON` files are written by `SPheno`:

- `WC.SPheno_1.json`: this contains the Wilson coefficients for quark flavour violating operators calculated at $Q = 160$ GeV
- `WC.SPheno_2.json`: this contains the Wilson coefficients for lepton flavour violating operators calculated at $Q = m_Z$

These output files contain the numerical results for the Wilson coefficients in the `FlavorKit` basis [22] discussed in section 5.3. However, only those coefficients are calculated which are needed for the flavour observables calculated in `SPheno`, see [30] for the corresponding list.

5.5. DsixTools

`DsixTools` [13, 37] is a Mathematica package for the handling of the SMEFT. It includes modules for the one-loop RG evolution from the NP scale down to low energies [11, 38, 39], tree-level matching to the WET at the electroweak scale [15] and RG evolution of the WET coefficients down to low energies [17].

Since version 1.1.2, `DsixTools` supports the handling of input and output files for the Wilson coefficients in `WCxf` format. Regarding the input, this is accomplished by extending the functionality of the usual `DsixTools` routine to read input cards, `ReadInputFiles`. The routine can be used as

```
ReadInputFiles["Options.dat", "WCsInput.json", "SMInput.dat"];
```

The `WCxf` input file can be provided in `JSON` or `YAML` formats. Note however that reading input files in `YAML` format requires previous installation of a `YAML` importer for Mathematica [40]. This file should contain the Wilson coefficients in the Warsaw basis as defined in section 3.1, and it must be supplemented with the files `Options.dat` and `SMInput.dat`, containing the working options and SM parameters, respectively. Note that in this case the SM parameters must be supplied in the weak basis defined in section 3.1. Internally, this basis is known in `DsixTools` as *WCXF basis*

and is different from the *fermion mass basis*.[9] `ReadInputFiles` first translates the `WCxf` input file to `SLHA` format, the default format in `DsixTools`, inspired by the Supersymmetry Les Houches Accord [41, 42], and then proceeds as usual. As a result of this, the file `WCsInput.dat`, with the resulting Wilson coefficients in `SLHA` format, is also produced when executing `ReadInputFiles`. Additionally, the user can also use `DsixTools` to convert a `WCxf` file to `SLHA` format simply as

```
WCXFtoSLHA["WCs.json", "WCs.dat", HIGHSCALE];
```

where in this case `WCs.json` and `WCs.dat` are the input `WCxf` and output `SLHA` files, respectively, and the high scale Λ, at which the coefficients are generated, is given in GeV in the argument `HIGHSCALE`. `DsixTools` can also export the SMEFT Wilson coefficients in `WCxf` format. First, the user must load the `EWmatcher` module of `DsixTools`, where all the required routines are contained. This is done with

```
LoadModule["EWmatcher"]
```

Once this module is loaded, one must rotate the SMEFT Wilson coefficients to the WCXF basis, ensuring that the standard CKM phase conventions are followed, see section 3.1. This can be done by executing the routine

```
RotateToWCXFBasis;
```

This routine creates the replacement rule `ToWCXFBasis`, which can then be used to obtain the values of specific Wilson coefficients in the WCXF basis. For instance,

```
LQ1[2, 2, 2, 3] /. ToWCXFBasis
```

would give the numerical value of the $\left(C_{\ell q}^{(1)}\right)_{2223}$ coefficient in this basis. One can proceed analogously with any SMEFT Wilson coefficient. After running `RotateToWCXFBasis`, the user can export all the Wilson coefficients in the WCXF basis to a `WCxf` file with the command

```
WriteWCsOutputFile["WCs.json", "WCXF", "JSON"];
```

or similarly

```
WriteWCsOutputFile["WCs.yaml", "WCXF", "YAML"];
```

where the second argument of the routine indicates the fermion basis and the third the output file format. Finally, one can also convert a Wilson coefficients file in `SLHA` format to `WCxf` format using

```
SLHAtoWCXF["WCs.dat", "WCs.json", CPV, SCALE, HIGHSCALE];
```

[9] One should also keep in mind that Wilson coefficients are dimensionless in `DsixTools`, which implies $C_{\texttt{WCxf}} = \frac{1}{\Lambda^2} C_{\texttt{DsixTools}}$, with Λ the high scale at which the Wilson coefficients are generated.

Note that the SLHA file does not contain information about whether CP violating entries are allowed in the Wilson coefficients, about the value of the scale at which these are given (SCALE) or about the high scale (HIGHSCALE) at which they are generated. This is why the user must provide these three details when running this routine.

5.6. python-smeftrunner

python-smeftrunner [43] is a Python package implementing approximately the same functionality as the SMEFTrunner module of the DsixTools Mathematica package, i.e. one-loop RG evolution in SMEFT. The main interface to the package is the SMEFT class, which is instantiated as

```
from smeftrunner import SMEFT
smeft = SMEFT()
```

Since version 2.0, python-smeftrunner supports importing Wilson coefficients in WCxf format, using the SMEFT Warsaw basis as defined in section 3.1, as follows,

```
with open('my_wcxf_input_file.yml', 'r') as f:
  smeft.load_wcxf(f)
```

Both JSON and YAML files are supported. Note that the SM parameters, which are not contained in the WCxf input file, have to be supplied by the user manually or using a DsixTools input file, and they have to be specified in the weak basis defined in section 3.1. For exchanging Wilson coefficients between WCxf-compatible Python packages, there is also a method directly taking an instance of wcxf.WC (cf. sec. 4.2) as input,

```
import wcxf
wc = wcxf.WC(...)
smeft.set_initial_wcxf(wc)
```

After the RG evolution, the Wilson coefficients at the output scale can be exported to WCxf, e.g.

```
with open('my_wcxf_output_file.yml', 'w') as f:
  smeft.dump_wcxf(C_out, scale_out=80, stream=f, fmt='yaml')
```

where the fmt parameter allows to specify the output format as YAML or JSON. For processing the Wilson coefficients with a WCxf-compatible Python package, an instance of wcxf.WC (cf. sec. 4.2) can also be exported instead,

```
wc = smeft.get_wcxf(C_out, scale_out=80)
```

5.7. FormFlavor

FormFlavor [44, 45] is a Mathematica package for deriving Wilson coefficients from scratch in a general new physics model. The code is modular across several pieces: CalcAmps for deriving analytic expressions for Wilson coefficients with the aid of the packages FeynArts [32] and FormCalc [31]; FFWilson for converting analytic expressions into numerical Wilson coefficients;

FFObservables for converting numerical Wilson coefficients into flavour observables; and FFModel, containing model specific files that interface with CalcAmps and FFWilson.

As of version 1.2.0, FormFlavor has the ability to read and write WCxf files. The output of FFWilson (and associated routines such as FFWilsonfromSLHA2) can be written to a JSON file with the command

```
WriteWilsonToJSON[WILSON, "path/filename.json"];
```

where WILSON is the output of FFWilson and path/filename.json is the desired output file and path. All Wilson coefficients are output in the FormFlavor basis [44] at the scale (normally, a high scale) where the coefficients have been evaluated. Information contained in the FFWilson output on the origin of the contribution (i.e., which diagrams provide the information) is lost in this output. JSON files can also be read for use with FFObservables by the command

```
ReadWilsonFromJSON["path/filename.json"];
```

where the input file must be in the formflavor basis.

5.8. SMEFTsim

SMEFTsim [46, 47] is a package containing a set of FeynRules [48] and UFO [49] models implementing the complete set of dimension-six operators in the Warsaw basis that conserve B and L. It allows symbolical calculations in Mathematica as well as numerical simulations e.g. in MadGraph5_aMC@NLO [50]. SMEFTsim contains 6 different models, corresponding to 3 possible flavour assumptions (a flavour general case, a $U(3)^5$ symmetric limit and a linear MFV case[10]) and 2 input parameter scheme choices, $\{\alpha_{ew}, m_Z, G_F\}$ and $\{m_W, m_Z, G_F\}$.

The basis used in SMEFTsim coincides with the "Warsaw mass" basis predefined in WCxf, up to small notational differences: all the Wilson coefficients are adimensional and a cutoff scale Λ is defined as an independent parameter with a default value of 1 TeV. Moreover, complex Wilson coefficients are parameterised defining their absolute value and complex phase instead of the real and imaginary parts.

The WCxf Python package will contain a command-line script wcxf2smeftsim, allowing to convert a JSON or YAML file into a param_card.dat, that can be used to run numerical simulations in MadGraph5_aMC@NLO with any of the flavour-general SMEFTsim models. The script allows to specify two optional parameters: the desired input scheme (which can be either alpha or mw) and the value to be assigned to the cutoff scale Λ, that can be an arbitrary number in GeV. For instance:

```
wcxf2smeftsim my_wcxf_input_file.json --output my_param_card.dat \
  --inputscheme alpha --cutoff-scale 1000
```

If inputscheme or cutoff-scale are not specified, they are assigned to the default values alpha and 1000 respectively.

[10] This case is based on the MFV paradigm [51], that assumes that the only source of CP violation in the $d \leq 6$ Lagrangian is the CP-odd phase of the CKM matrix and that a $U(3)^5$ flavour symmetry is present, broken only by insertions of the Yukawa couplings. The automated implementation contains all the relevant flavour spurion insertions up to linear order in $(Y_f Y_f^\dagger)$, $(Y_f^\dagger Y_f)$.

5.9. SMEFT Feynman Rules

`SMEFT Feynman Rules` [14] is a Mathematica package evaluating the Feynman rules for the SMEFT in terms of the physical (mass-eigenstates) fields of the theory. It works using the `FeynRules` package [52]. Depending on the users' choice, the package is able to calculate Feynman rules analytically for the unitary or for a general R_ξ gauge (including ghost interactions), for the full model or for a chosen field sector and/or subset of higher dimension operators. The results are available in Mathematica or LaTeX formats. Using `FeynRules` interfaces they can also be exported to formats accepted by other symbolic packages (e.g. `FeynCalc`) or to the UFO format [49] used by Monte Carlo generators like MadGraph.

As the `SMEFT Feynman Rules` package is designed to produce the analytical form of the SMEFT Lagrangian and interaction vertices in terms of the physical fields, the package itself does not need the numerical values of the Wilson coefficients. However, the derived Feynman rules are just the intermediate step in the calculations of physical observables and usually are fed as input to other programs, calculating such quantities and using the numerical input for Wilson coefficients. Thus, starting from version 1.1, `SMEFT Feynman Rules` is able to import and export values of the Wilson coefficients in `WCxf` format.

The import routine reads the `WCxf` input from the Wilson coefficient file in the `JSON` format (see Sec. 2.3) and produces a model file containing the SMEFT parameters in the `FeynRules` format. It assumes that the imported values of Wilson coefficients are defined in the "Warsaw mass" basis defined at the end of Section 3.1 (see also Refs. [14, 15]). The transcription from the `WCxf` format can be done using the following command (in the Mathematica notebook):

```
<< smeft_wcxf.m

WCXFtoSMEFT["wcxf_input_file", "smeft_par_model_file" ];
```

The '`smeft_par_model_file`' is generated in the '`definitions`' sub-directory of the `SMEFT Feynman Rules` package tree. Next, the user should edit the '`smeft_control_variables.m`' file accordingly, defining the correct name of the model for the file for the UFO format generator:

```
...
$UFOModelFile = "smeft_par_model_file";
...
```

In order to use this model file, the user should first run the '`smeft_initialize.m`' program (see program web page [53] for more detailed instructions):

```
<< smeft_initialize.m;
```

which calculates the analytical form of physical SMEFT Lagrangian, without assigning numerical values to the Wilson coefficients, and stores it to disk files. Next, in the new Mathematica notebook (the `FeynRules` package cannot reload model file without being restarted), the user should run

```
<< smeft_UFO.m;
```

The '`smeft_UFO.m`' program reads the physical SMEFT Lagrangian with the Wilson coefficients initialised to the numerical values defined in the model file generated from `WCxf` input, generates the

Feynman rules in the form of the UFO output and, if necessary, could also be adapted to produce other types of output formats supported by `FeynRules`. `'smeft_UFO.m'` can be executed multiple times (always in new Mathematica notebooks), after each modification of the Wilson coefficients values, without rerunning the `'smeft_initialize.m'` program.

The inverse translation from `FeynRules` to `WCxf JSON` format can be done using the commands (files should be in the current directory or full paths must be specified in their names):

```
<< smeft_wcxf.m

SMEFTtoWCXF["smeft_par_model_file", "wcxf_output_file"];
```

6. Conclusions and outlook

We have defined a new data exchange format for Wilson coefficients of local operators parameterising low-energy effects of physics beyond the Standard Model. By implementing this format both for the Standard Model Effective Field Theory (SMEFT) above and the Weak Effective Theory (WET) below the electroweak scale in several public codes, we have demonstrated that this format facilitates interfacing different codes dealing with precision tests of the Standard Model. This makes it more realistic to construct global likelihoods – including flavour physics, electroweak precision tests, and Higgs physics – for model-independent analyses of new physics in the future. Additionally, model-specific analyses can be greatly simplified, since the EFT can serve as as a tool to separate the model-independent low-energy phenomenology from the model-dependent short-distance physics. This can even be automatised, as demonstrated by `FlavorKit` and `FormFlavor` for WET, and more recently by a complete tree-level matching in SMEFT [54, 55].

Since the format is extensible, we encourage the submission of new bases or EFTs to the public repository[11] via a pull request. The documentation of the bases and associated software is available on the project website.[12]

Acknowledgements

The work of D. S., J. A., and X. P. is supported by the DFG cluster of excellence "Origin and Structure of the Universe". The work of A. C. is supported by the DFG grant BU 1391/2-1. I. B. and Y. J. are supported by the Villum Foundation, NBIA, the Discovery Centre at Copenhagen University and the Danish National Research Foundation (DNRF91). A. V. acknowledges financial support from the "Juan de la Cierva" program (27-13-463B-731) funded by the Spanish MINECO as well as from the grants FPA2014-58183-P, FPA2017-85216-P and SEV-2014-0398 (MINECO), and PROMETEOII/ 2014/084 (Generalitat Valenciana). D. v. D. is supported by the Emmy Noether programme of the Deutsche Forschungsgemeinschaft (DFG) under grant DY 130/1-1. The work of J.R. is supported in part by the National Science Centre, Poland, under research grants DEC-2015/19/B/ST2/02848 and DEC-2015/18/M/ST2/00054. W. P. is supported by the DFG, project nr. PO 1337-7/1. F. S. is supported by the ERC Recognition Award ERC-RA-0008 of the Helmholtz Association. I. B. and Y. J. also thank Michael Trott for valuable discussions and suggestions in

[11] https://github.com/wcxf/wcxf-bases
[12] https://wcxf.github.io

designing the `SMEFTsim` interface. A. C. and A. V. thank Javier Fuentes-Martín and Javier Virto for their collaboration in the development of `DsixTools`. D. S. thanks Alex Arbey, Christoph Bobeth, Nazila Mahmoudi, Ayan Paul, and Mauro Valli for useful discussions.

References

[1] D. de Florian, et al., Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector, arXiv:1610.07922, doi:10.23731/CYRM-2017-002.
[2] I. Brivio, M. Trott, The Standard Model as an Effective Field Theory, arXiv:1706.08945.
[3] F. Mahmoudi, et al., Flavour Les Houches Accord: Interfacing Flavour related Codes, Comput. Phys. Commun. 183 (2012) 285–298. arXiv:1008.0762, doi:10.1016/j.cpc.2011.10.006.
[4] A. Falkowski, B. Fuks, K. Mawatari, K. Mimasu, F. Riva, V. Sanz, Rosetta: an operator basis translator for Standard Model effective field theory, Eur. Phys. J. C75 (12) (2015) 583. arXiv:1508.05895, doi:10.1140/epjc/s10052-015-3806-x.
[5] S. Weinberg, Baryon and Lepton Nonconserving Processes, Phys. Rev. Lett. 43 (1979) 1566–1570. doi:10.1103/PhysRevLett.43.1566.
[6] W. Buchmuller, D. Wyler, Effective Lagrangian Analysis of New Interactions and Flavor Conservation, Nucl. Phys. B268 (1986) 621–653. doi:10.1016/0550-3213(86)90262-2.
[7] B. Grzadkowski, M. Iskrzynski, M. Misiak, J. Rosiek, Dimension-Six Terms in the Standard Model Lagrangian, JHEP 10 (2010) 085. arXiv:1008.4884, doi:10.1007/JHEP10(2010)085.
[8] L. Lehman, Extending the Standard Model Effective Field Theory with the Complete Set of Dimension-7 Operators, Phys. Rev. D90 (12) (2014) 125023. arXiv:1410.4193, doi:10.1103/PhysRevD.90.125023.
[9] B. Henning, X. Lu, T. Melia, H. Murayama, 2, 84, 30, 993, 560, 15456, 11962, 261485, ...: Higher dimension operators in the SM EFT, JHEP 08 (2017) 016. arXiv:1512.03433, doi:10.1007/JHEP08(2017)016.
[10] L. F. Abbott, M. B. Wise, The Effective Hamiltonian for Nucleon Decay, Phys. Rev. D22 (1980) 2208. doi:10.1103/PhysRevD.22.2208.
[11] R. Alonso, E. E. Jenkins, A. V. Manohar, M. Trott, Renormalization Group Evolution of the Standard Model Dimension Six Operators III: Gauge Coupling Dependence and Phenomenology, JHEP 04 (2014) 159. arXiv:1312.2014, doi:10.1007/JHEP04(2014)159.
[12] R. Alonso, H.-M. Chang, E. E. Jenkins, A. V. Manohar, B. Shotwell, Renormalization group evolution of dimension-six baryon number violating operators, Phys. Lett. B734 (2014) 302–307. arXiv:1405.0486, doi:10.1016/j.physletb.2014.05.065.
[13] A. Celis, J. Fuentes-Martin, A. Vicente, J. Virto, DsixTools: The Standard Model Effective Field Theory Toolkit, Eur. Phys. J. C77 (6) (2017) 405. arXiv:1704.04504, doi:10.1140/epjc/s10052-017-4967-6.
[14] A. Dedes, W. Materkowska, M. Paraskevas, J. Rosiek, K. Suxho, Feynman rules for the Standard Model Effective Field Theory in R_ξ -gauges, JHEP 06 (2017) 143. arXiv:1704.03888, doi:10.1007/JHEP06(2017)143.
[15] J. Aebischer, A. Crivellin, M. Fael, C. Greub, Matching of gauge invariant dimension-six operators for $b \to s$ and $b \to c$ transitions, JHEP 05 (2016) 037. arXiv:1512.02830, doi:10.1007/JHEP05(2016)037.
[16] E. E. Jenkins, A. V. Manohar, P. Stoffer, Low-Energy Effective Field Theory below the Electroweak Scale: Operators and Matching, arXiv:1709.04486.
[17] J. Aebischer, M. Fael, C. Greub, J. Virto, B physics Beyond the Standard Model at One Loop: Complete Renormalization Group Evolution below the Electroweak Scale, JHEP 09 (2017) 158. arXiv:1704.06639, doi:10.1007/JHEP09(2017)158.
[18] E. E. Jenkins, A. V. Manohar, P. Stoffer, Low-Energy Effective Field Theory below the Electroweak Scale: Anomalous Dimensions, arXiv:1711.05270.
[19] WCxf web site. https://wcxf.github.io
[20] D. M. Straub, et al., flavio – flavour phenomenology in the standard model and beyond. doi:10.5281/zenodo.594587. https://flav-io.github.io
[21] D. van Dyk, et al., EOS – A HEP Programm for Flavour Observables. https://eos.github.io
[22] W. Porod, F. Staub, A. Vicente, A Flavor Kit for BSM models, Eur. Phys. J. C74 (8) (2014) 2992. arXiv:1405.1434, doi:10.1140/epjc/s10052-014-2992-2.
[23] W. Porod, F. Staub, A. Vicente, FlavorKit. http://sarah.hepforge.org/FlavorKit.html
[24] F. Staub, SARAH, arXiv:0806.0538.

[25] F. Staub, From Superpotential to Model Files for FeynArts and CalcHep/CompHep, Comput. Phys. Commun. 181 (2010) 1077–1086. arXiv:0909.2863, doi:10.1016/j.cpc.2010.01.011.
[26] F. Staub, Automatic Calculation of supersymmetric Renormalization Group Equations and Self Energies, Comput. Phys. Commun. 182 (2011) 808–833. arXiv:1002.0840, doi:10.1016/j.cpc.2010.11.030.
[27] F. Staub, SARAH 3.2: Dirac Gauginos, UFO output, and more, Comput. Phys. Commun. 184 (2013) 1792–1809. arXiv:1207.0906, doi:10.1016/j.cpc.2013.02.019.
[28] F. Staub, SARAH 4 : A tool for (not only SUSY) model builders, Comput. Phys. Commun. 185 (2014) 1773–1790. arXiv:1309.7223, doi:10.1016/j.cpc.2014.02.018.
[29] W. Porod, SPheno, a program for calculating supersymmetric spectra, SUSY particle decays and SUSY particle production at e+ e- colliders, Comput. Phys. Commun. 153 (2003) 275–315. arXiv:hep-ph/0301101, doi:10.1016/S0010-4655(03)00222-4.
[30] W. Porod, F. Staub, SPheno 3.1: Extensions including flavour, CP-phases and models beyond the MSSM, Comput. Phys. Commun. 183 (2012) 2458–2469. arXiv:1104.1573, doi:10.1016/j.cpc.2012.05.021.
[31] T. Hahn, M. Perez-Victoria, Automatized one loop calculations in four-dimensions and D-dimensions, Comput. Phys. Commun. 118 (1999) 153–165. arXiv:hep-ph/9807565, doi:10.1016/S0010-4655(98)00173-8.
[32] T. Hahn, Generating Feynman diagrams and amplitudes with FeynArts 3, Comput. Phys. Commun. 140 (2001) 418–431. arXiv:hep-ph/0012260, doi:10.1016/S0010-4655(01)00290-9.
[33] T. Hahn, Automatic loop calculations with FeynArts, FormCalc, and LoopTools, Nucl. Phys. Proc. Suppl. 89 (2000) 231–236. arXiv:hep-ph/0005029, doi:10.1016/S0920-5632(00)00848-3.
[34] T. Hahn, New features in FormCalc 4, Nucl. Phys. Proc. Suppl. 135 (2004) 333–337, [,333(2004)]. arXiv:hep-ph/0406288, doi:10.1016/j.nuclphysbps.2004.09.018.
[35] T. Hahn, New developments in FormCalc 4.1, eConf C050318 (2005) 0604. arXiv:hep-ph/0506201.
[36] B. Chokoufe Nejad, T. Hahn, J. N. Lang, E. Mirabella, FormCalc 8: Better Algebra and Vectorization, J. Phys. Conf. Ser. 523 (2014) 012050. arXiv:1310.0274, doi:10.1088/1742-6596/523/1/012050.
[37] A. Celis, J. Fuentes-Martín, A. Vicente, J. Virto, DsixTools. https://dsixtools.github.io/
[38] E. E. Jenkins, A. V. Manohar, M. Trott, Renormalization Group Evolution of the Standard Model Dimension Six Operators I: Formalism and lambda Dependence, JHEP 10 (2013) 087. arXiv:1308.2627, doi:10.1007/JHEP10(2013)087.
[39] E. E. Jenkins, A. V. Manohar, M. Trott, Renormalization Group Evolution of the Standard Model Dimension Six Operators II: Yukawa Dependence, JHEP 01 (2014) 035. arXiv:1310.4838, doi:10.1007/JHEP01(2014)035.
[40] Z. Bjornson, MYaml. https://github.com/zbjornson/MYaml
[41] P. Z. Skands, et al., SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators, JHEP 07 (2004) 036. arXiv:hep-ph/0311123, doi:10.1088/1126-6708/2004/07/036.
[42] B. C. Allanach, et al., SUSY Les Houches Accord 2, Comput. Phys. Commun. 180 (2009) 8–25. arXiv:0801.0045, doi:10.1016/j.cpc.2008.08.004.
[43] D. M. Straub, X. Pan, python-smeftrunner. https://github.com/DsixTools/python-smeftrunner
[44] J. A. Evans, D. Shih, FormFlavor Manual, arXiv:1606.00003.
[45] J. A. Evans, D. Shih, FormFlavor. https://formflavor.hepforge.org
[46] I. Brivio, Y. Jiang, M. Trott, The SMEFTsim package, theory and tools, arXiv:1709.06492.
[47] I. Brivio, Y. Jiang, M. Trott, SMEFTsim. http://feynrules.irmp.ucl.ac.be/wiki/SMEFT
[48] N. D. Christensen, C. Duhr, FeynRules - Feynman rules made easy, Comput. Phys. Commun. 180 (2009) 1614–1641. arXiv:0806.4194, doi:10.1016/j.cpc.2009.02.018.
[49] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, T. Reiter, UFO - The Universal FeynRules Output, Comput. Phys. Commun. 183 (2012) 1201–1214. arXiv:1108.2040, doi:10.1016/j.cpc.2012.01.022.
[50] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, M. Zaro, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, JHEP 07 (2014) 079. arXiv:1405.0301, doi:10.1007/JHEP07(2014)079.
[51] G. D'Ambrosio, G. F. Giudice, G. Isidori, A. Strumia, Minimal flavor violation: An Effective field theory approach, Nucl. Phys. B645 (2002) 155–187. arXiv:hep-ph/0207036, doi:10.1016/S0550-3213(02)00836-2.
[52] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, B. Fuks, FeynRules 2.0 - A complete toolbox for tree-level phenomenology, Comput. Phys. Commun. 185 (2014) 2250–2300. arXiv:1310.1921, doi:10.1016/j.cpc.2014.04.012.
[53] A. Dedes, W. Materkowska, M. Paraskevas, J. Rosiek, K. Suxho, SMEFT Feynman rules web site. http://www.fuw.edu.pl/smeft

[54] J. C. Criado, MatchingTools: a Python library for symbolic effective field theory calculations, arXiv:1710.06445.

[55] J. de Blas, J. C. Criado, M. Perez-Victoria, J. Santiago, Effective description of general extensions of the Standard Model: the complete tree-level dictionary, arXiv:1711.10391.