IAEA 60 Years
Atoms for Peace and Development

An EXFOR parser prototype for layer 0 and about Transformers

Georg Schnabel
g.schnabel [at] iaea.org

Nuclear Data Section
Division of Physical and Chemical Sciences NAPC
Department for Nuclear Sciences and Applications
IAEA, Vienna

WPEC-SG50 meeting

5 May 2022

Outline

- Design considerations of an Python EXFOR parser for layer 0 and implementation details
- About transformers and how they can be used to interlink with codes at higher layers

Basic guiding principle

- Python data structure should contain all EXFOR information
- We may also want to go back to EXFOR master files
- The data structure should not be “too far away” from EXFOR

Usage example of parser

Reading an EXFOR master file

```
from exfor_parser import ExforBaseParser
parser = ExforBaseParser()
exfor_dic = parser.readfile('testdata/entry_21308.txt')
```

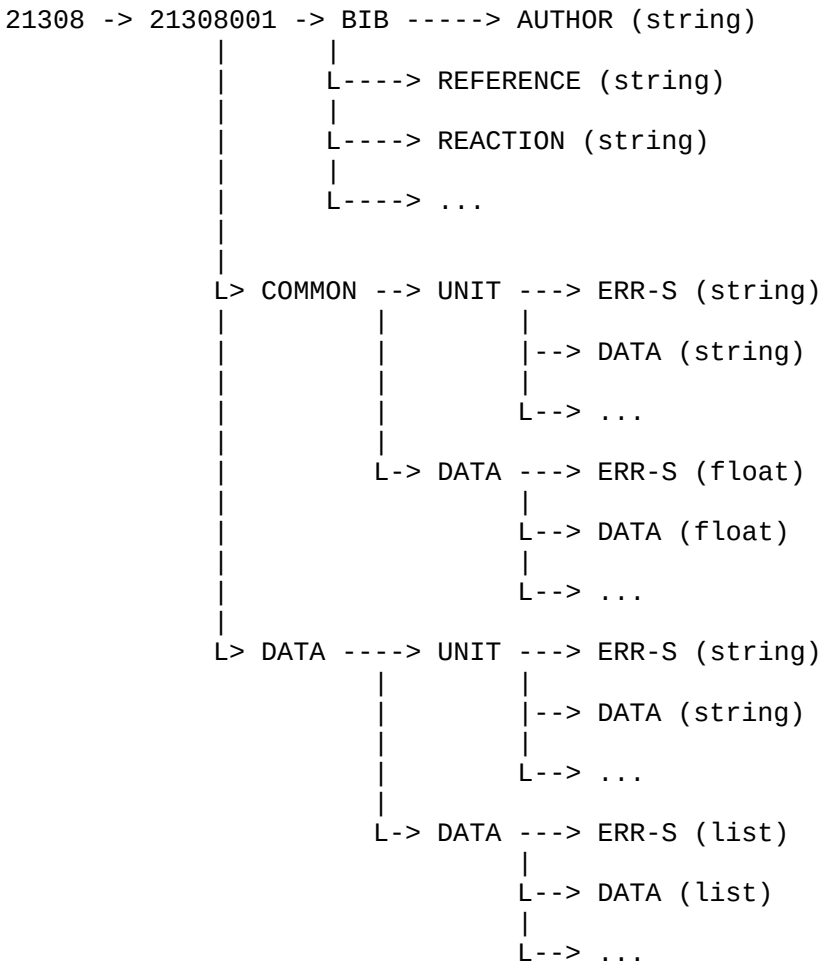
Change something

```
exfor_dic['21308']['21308001']['BIB']['AUTHOR'] = 'Mr. Anonymous'
```

Writing back an EXFOR master file

```
parser.writefile('testoutput.x4', exfor_dic)
```

Structure of the Python data structure: nested dictionaries



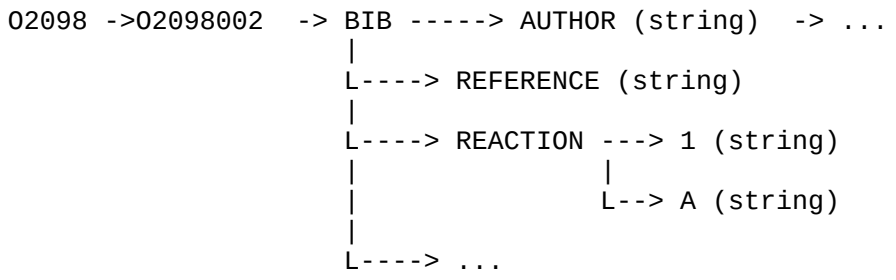
Example:

```
exfor_dic['21308']['21308001']['BIB']['AUTHOR']

# returns: (D.B.GAYTHER,M.C.MOXON,B.W.THOMAS,R.B.THOM,
            J.B.BRISLAND)
```

Handling of pointers

If a field contains pointers, the content of the field will not be a string but a dictionary with the keys given by the pointers, e.g.,



Example:

```
exfor_entry['O2098']['O2098002']['BIB']['REACTION']
# returns
# {  '1': '(8-O-17(P,G)9-F-18,,SIG,,SFC)S factors of primary transitions',
     '2': '(8-O-17(P,G)9-F-18,,SIG,,SFC)S factors of secondary transitions' }
```

Design principles #2

- All text strings in fields are preserved (EXFOR codes + free text)
- Line breaks are preserved
- (Different) units are preserved
- COMMON blocks are preserved
- etc. etc.

→ Even though the EXFOR entry is now in a Python data structure, we still have a rather unhandy format.

Transformers

- The idea is to apply functions that take the EXFOR dictionary as returned by the *ExforBaseParser* as argument, perform some transformations on the data and/or the structure and return a modified data structure
- We can call such functions *transformers*
- A variety of transformers can be conceived to prepare the data structure in ways more pertinent for processing codes at higher layers (1,2,3) and/or the end-user

Example of a transformer: unitfy

- Bring all the energy units to **MeV** and all cross section-like units to **mbarn** in the **DATA** and **COMMON** dictionaries. Also handle compound units such as B*GeV for xs integrals and B/SR for angular distributions

```
from exfor_parser.trafos import unify
transformed_exfor_dic = unitfy(exfor_dic)
```

Ideas for more transformers

- **Uncommonify**: Integrate the data in the COMMON block into the DATA block
- **Detextify:** Remove free-form text from fields
- **Reactify:** Unify the representation in the REACTION string
- **Unpointerfy:** Split a subentry with pointers into virtual subentries with an augmented subentry id, e.g., O2098002 → O20980021 and O20980022 (as in the IAEA-EXFOR system by V. Zerkin)
- **Tablify:** Get rid of all the nested dictionary structures and condense the information into a single table (similar to C4)
- etc. etc.

Summary

- Basic EXFOR parsing implemented
- More complex transformations afterwards by *transformers*
- Source code [(c) IAEA and MIT license] available at

  https://github.com/iaea-nds/exfor-parserpy.git
- Feedback and contributions appreciated
- My wish/hope: We build something together and release it open-source for everyone (NEA + IAEA Member States)