PRINCETON UNIVERSITY

Awkward Array

PONDD: Personal Data Delivery

Describe Data to get Science-Data-Ready Tooling:

Awkward as a Target for Kaitai Struct YAML

Manasvi Goyal[1], Ianna Osborne[1], Jim Pivarski[1], Amy Roberts[2], Andrea Zonca[3]

[1]Princeton University *[2]University of Colorado Denver* *[3]San Diego Supercomputer Center*

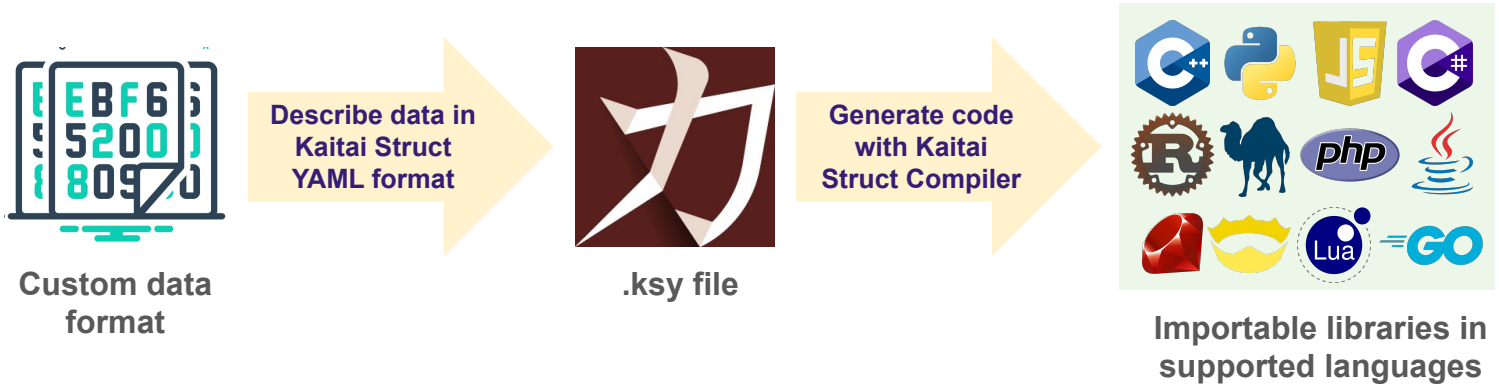
Support for this work was provided by NSF cooperative agreements OAC-1836650 and PHY-2323298 (IRIS-HEP) and grants OAC-2104003 (PONDD) and OAC-2103945 (Awkward Array).

Need and Motivation

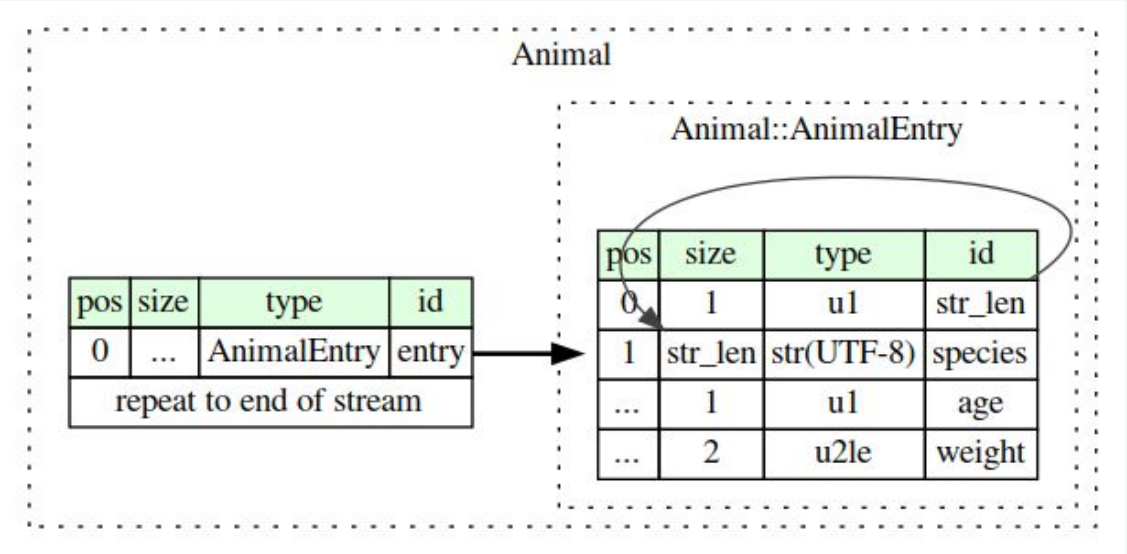
- ➔ Scientific data formats differ across experiments due to specialized hardware and data acquisition systems.
- ➔ Proliferation of custom formats has been a prominent challenge for small to mid-scale experiments such as Cryogenic Dark Matter Search (CDMS).
- ➔ Simplifying the process of converting a custom data format to analysis code still holds immense value for scientific communities even beyond HEP.

Kaitai Struct YAML (KSY)

- ➔ Kaitai Struct is a cross-language and cross-platform declarative language.
- ➔ Describe the very structure of the data, not how you read or write it.
- ➔ Uses a YAML-like description of a arbitrary binary format to generate code for reading a raw data file.



Let's Take an Example…

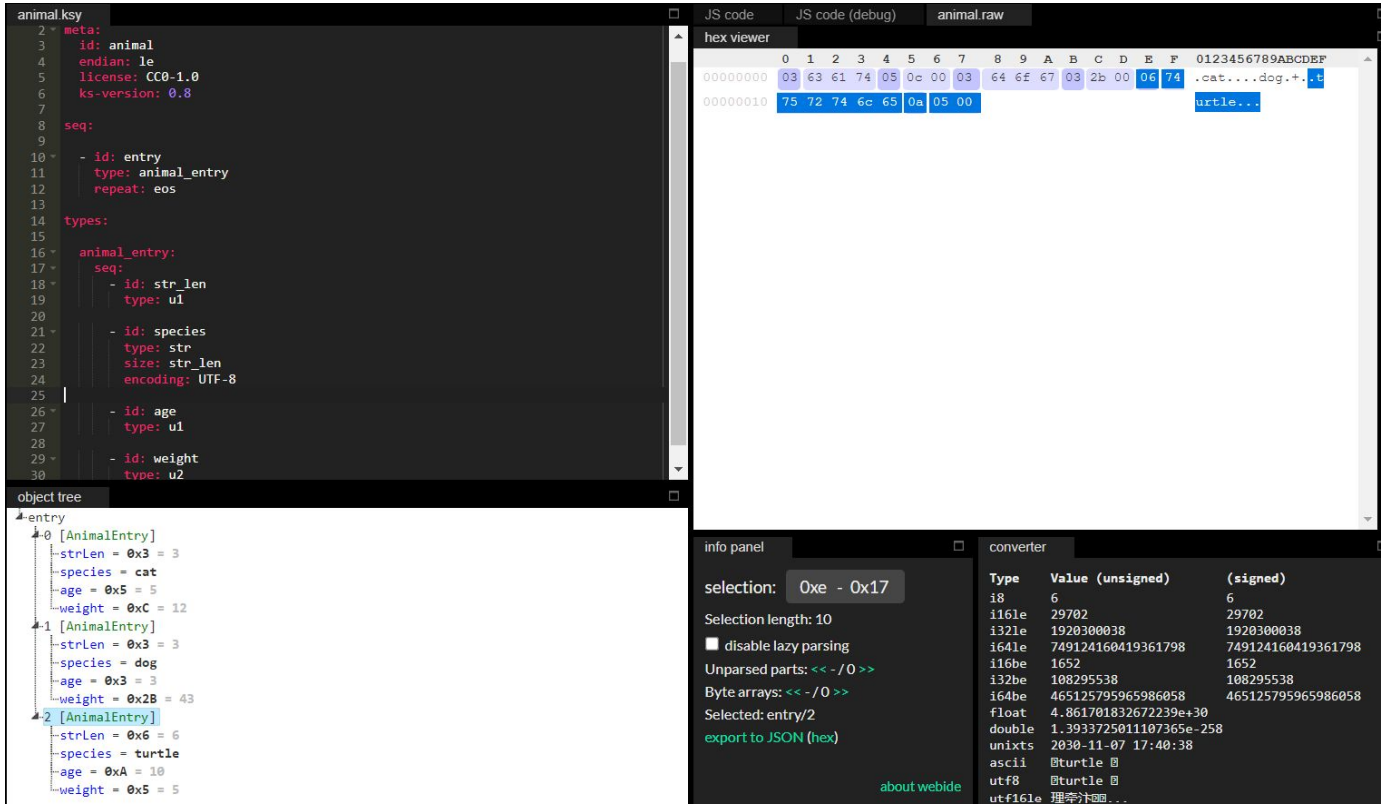


Bits	0 - 7	8 - 15	16 - 23	24 - 31	32 - 39	40 - 55
Value	0x03	0x63	0x6f	0x77	0x06	0xDC05

animal.ksy

```
meta:
  id: animal
  endian: le
seq:
  - id: entry
    type: animal_entry
    repeat: eos
types:
  animal_entry:
    seq:
      - id: str_len
        type: u1
      - id: species
        type: str
        size: str_len
        encoding: UTF-8
      - id: age
        type: u1
      - id: weight
        type: u2
```

Kaitai Web IDE 🔗



Python Parser for animal.ksy

```
class Animal(KaitaiStruct):
    def __init__(self, _io, _parent=None, _root=None):
        self._io = _io
        self._parent = _parent
        self._root = _root if _root else self
        self._read()

    def _read(self):
        self.entry = []
        i = 0
        while not self._io.is_eof():
            self.entry.append(Animal.AnimalEntry(self._io, self, self._root))
            i += 1


    class AnimalEntry(KaitaiStruct):
        def __init__(self, _io, _parent=None, _root=None):
            self._io = _io
            self._parent = _parent
            self._root = _root if _root else self
            self._read()

        def _read(self):
            self.str_len = self._io.read_u1()
            self.species = (self._io.read_bytes(self.str_len)).decode(u"UTF-8")
            self.age = self._io.read_u1()
            self.weight = self._io.read_u2le()
```

```
$ kaitai-struct-compiler -t python
  example_data/schemas/animal.ksy
```

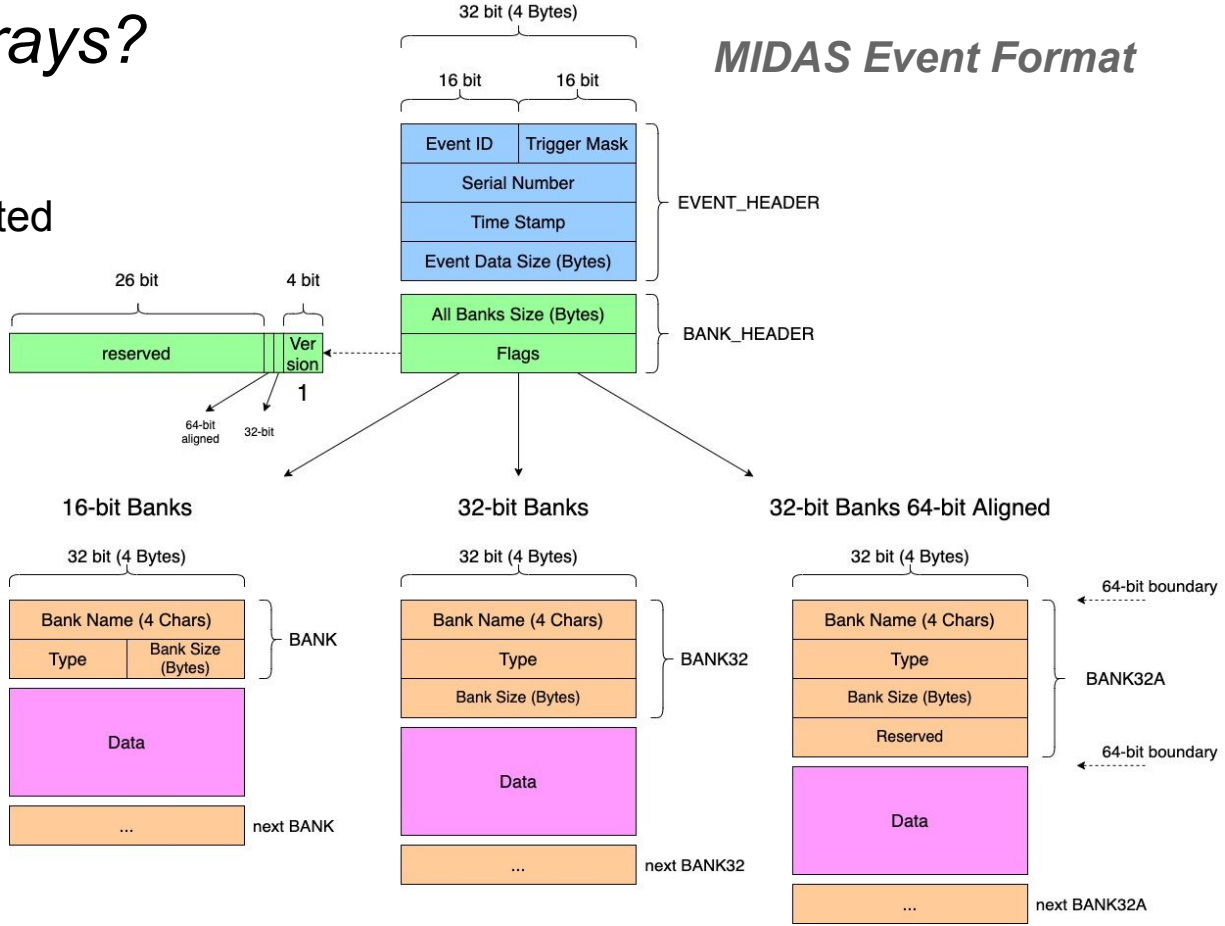

animal.py

```
Species: cat
Age: 5
Weight: 12
Species: dog
Age: 3
Weight: 43
Species: turtle
Age: 10
Weight: 5
```

Why Awkward Arrays?

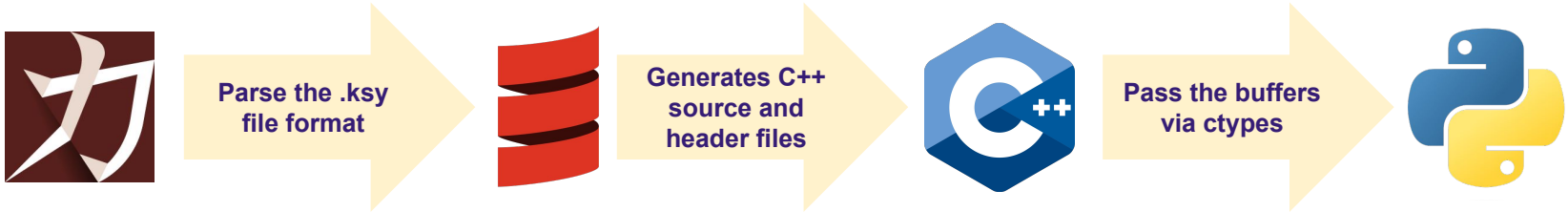
Large files and complicated data structures

- ➔ nested records
- ➔ variable-length lists
- ➔ mixed types
- ➔ missing data.

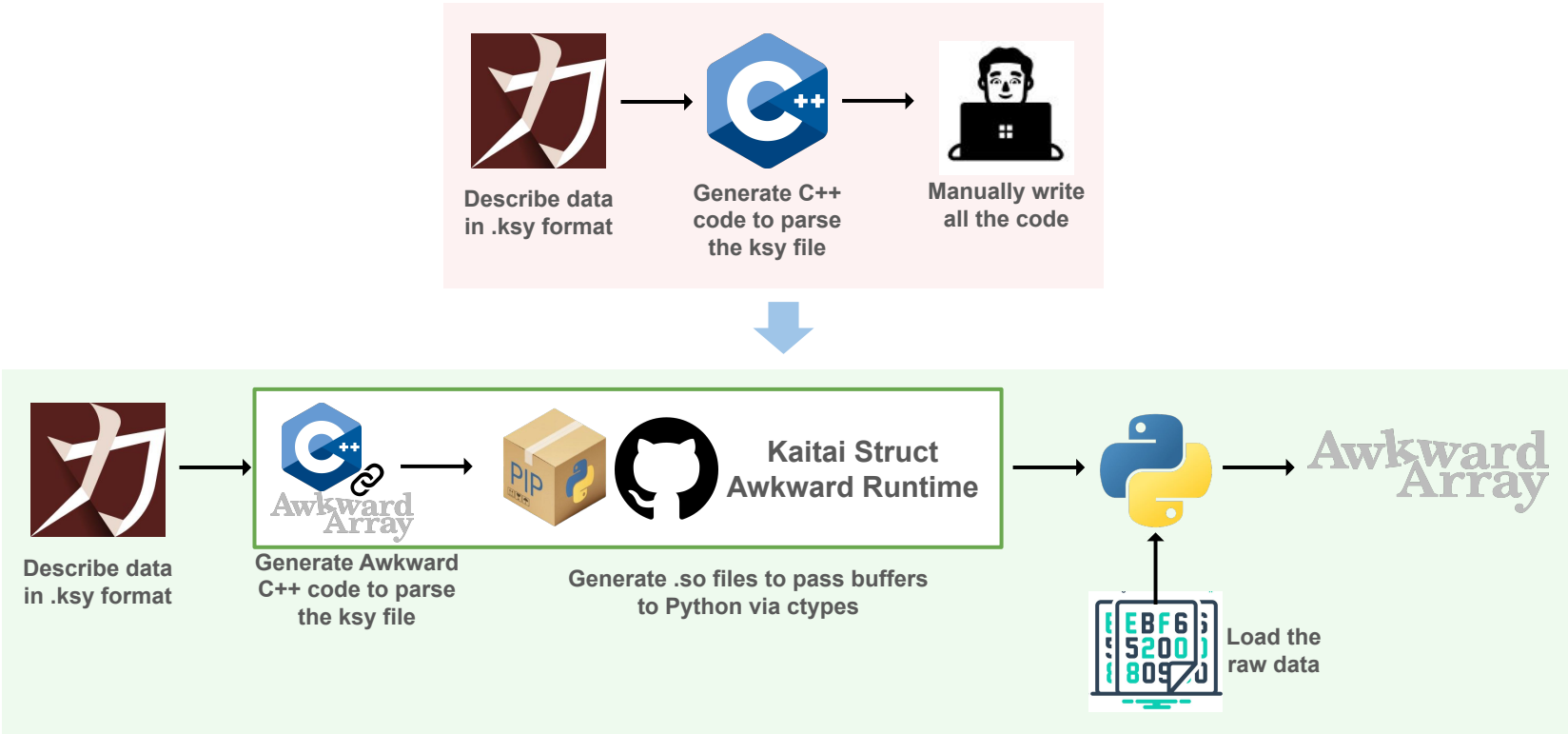


Awkward Target for Kaitai Struct

- ➔ Generates C++ header & source files with LayoutBuilder filling instructions.
- ➔ Users can simply describe their data format in KSY format just once.
- ➔ This KSY file can be converted into a compiled Python module which takes the raw data and converts it into Awkward Arrays.



Kaitai Struct Awkward Runtime: User Interface



Kaitai Struct Awkward Runtime: Steps

Clone **`kaitai_struct_awkward_runtime`** repository.

```
git clone --recursive https://github.com/ManasviGoyal/kaitai_struct_awkward_runtime.git
```

Generate the source and header files for Awkward target.

```
./kaitai-struct-compiler -t awkward --outdir src-animal example_data/schemas/animal.ksy
```

Install the module.

```
pip install .
```

Build **`awkward-kaitai`** to generate **`libanimal.so`**.

```
awkward-kaitai-build src-animal/animal.cpp -b build
```

KSY → LayoutBuilder

```
using AnimalBuilderType =
  RecordBuilder<
    RecordField<Field_animal::animalA__Zentry,
      ListOffsetBuilder<int64_t, RecordBuilder<
        RecordField<Field_animal_entry::animal_entryA__Zstr_len,
          NumpyBuilder<uint8_t>>,
        RecordField<Field_animal_entry::animal_entryA__Zspecies,
          ListOffsetBuilder<int64_t, NumpyBuilder<uint8_t>>>,
        RecordField<Field_animal_entry::animal_entryA__Zage,
          NumpyBuilder<uint8_t>>,
        RecordField<Field_animal_entry::animal_entryA__Zweight,
          NumpyBuilder<uint16_t>>
      >>
    >
  >;
```

animal.ksy

```
meta:
  id: animal
  endian: le

seq:
  - id: entry
    type: animal_entry
    repeat: eos

types:
  animal_entry:
    seq:
      - id: str_len
        type: u1

      - id: species
        type: str
        size: str_len
        encoding: UTF-8

      - id: age
        type: u1

      - id: weight
        type: u2
```

Python Module → Awkward Arrays

```
import awkward_kaitai

animal = awkward_kaitai.Reader("./src-animal/libanimal.so") # pass the shared library

awkward_array = animal.load("example_data/data/animal.raw") # pass the raw data file

awkward_array.to_list()
```

Awkward Array for ***animal.ksy***

```
[{'animalA__Zentry': [
    {'animal_entryA__Zstr_len': 3, 'animal_entryA__Zspecies': 'cat',
     'animal_entryA__Zage': 5, 'animal_entryA__Zweight': 12},
    {'animal_entryA__Zstr_len': 3, 'animal_entryA__Zspecies': 'dog',
     'animal_entryA__Zage': 3, 'animal_entryA__Zweight': 43},
    {'animal_entryA__Zstr_len': 6,'animal_entryA__Zspecies': 'turtle',
     'animal_entryA__Zage': 10, 'animal_entryA__Zweight': 5}
]}]
```

Ongoing and Future Work

- ➔ Support for **`import`**
- ➔ Use **`IndexedBuilder`** for **`EnumType`** cases
- ➔ Add **`awkward-kaitai`** module on **`PyPI`**
- ➔ Merge the changes to add Awkward Target in **`kaitai_struct_compiler`**
- ➔ Integrate **`kaitai_struct_awkward_runtime`** with **`kaitai-io`** project

Andrea Zonca is working on these!

Current Project : Extending Awkward Arrays to GPU

- ➔ Add the remaining 87/145 CUDA kernels for each CPU kernel.
- ➔ Add tests for these CUDA kernels and performance studies.

Progress so far

- ➔ Removed the dead code and obsolete kernels. #2875 #2876
- ➔ Fixed the errors in existing CUDA kernels. #2877
- ➔ Add 15/87 simpler CUDA kernels. #2880 #2930
- ➔ Implementing unit tests for CUDA kernels. #2922 #2938

Awkward Array