A Proposal for an OpenMath JSON Encoding
OpenMath workshop CICM 2018

Tom Wiesing

October 4, 2018

Abstract

OpenMath is a semantic representation of mathematical objects.

There are several encodings of OpenMath Objects, most notably the XML and Binary encodings. JSON is a lightweight data-interchange format that is present natively in many programming languages.

A few OpenMath JSON encodings already exist, which all have their advantages and disadvantages. These commonly correspond to a naive representation of the XML encoding and thus do not make use of some of the features that JSON offers.

In this paper we propose a new OpenMath JSON encoding, which combines the advantages of the above.

[1] [2] EdN:1
EdN:2

1 Introduction

OpenMath is a semantic representation of mathematical objects. [3] Because this paper is submitted to the OpenMath workshop, we will assume that reader is familar with OpenMath and will not introduce it further here. EdN:3

JSON [4], short for **J**ava**S**cript **O**bject **N**otation, is a lightweight data- EdN:4

[1] EdNote: Citations
[2] EdNote: Flow
[3] EdNote: Is this OK?
[4] EdNote: cite json

interchange format. While being a subset of JavaScript, it is defined independently. JSON can represent both primitive types and composite types.

Primitive JSON types are strings (e.g. `"Hello␣world"`), Numbers (e.g. `42` or `3.14159265`), Booleans (`true` and `false`) and `null`. Composite JSON types are either (non-homogeneous) arrays (e.g. `[1, "two", false]`) or key-value pairs called objects (e.g. `{"foo": "bar", "answer": 42}`).

Constructs corresponding to JSON objects are found in most programming languages. Futhermore, the syntax is very simple; hence many languages have built-in facilities for translating their existing data structures to and from JSON. The use for an OpenMath JSON encoding is clear: It would enable easy use of OpenMath across many languages.

There are existing approaches for encoding OpenMath as JSON. We will discuss two particular ones here.

XML as JSON The JSONML standard [5] allows generic encoding of arbitrary XML as JSON. This can easily be adapted to the case of OpenMath. To encode an OpenMath object as JSON, one first encodes it as XML and then makes use of JSONML in a second step. Using this method, the term $\mathrm{plus}(x, 5)$ would correspond to: EdN:5

```
[
    "OMOBJ",
    {"xmlns":"http://www.openmath.org/OpenMath"},
    [
        "OMA",
        [
            "OMS",
            {"cd": "arith1", "name": "plus"}
        ],
        [
            "OMV",
            {"name": "x"}
        ],
        [
            "OMI",
            "5"
        ]
    ]
```

[5] EdNote: cite http://www.jsonml.org/

```
]
```

This translation has the advantage that it is near-trivial to translate between the XML and JSON encodings of OpenMath. It also has some disadvantages:

- The encoding does not use the native JSON datatypes. One of the advantages of JSON is that it can encode most basic data types directly, without having to turn the data values into strings. To encode the floating point value `1e-10` (a valid JSON literal) using the JSONML encoding, one can not directly place it into the result. Instead, one has to turn it into a string first. Despite many JSON implementations providing such a functionality, in practice this would require frequent translation between strings and high-level datatypes. This is not what JSON is intended for, instead the provided data types should be used.
- The akwardness of some of the XML encoding remains. Due to the nature of XML the XML encoding sometimes needs to introduce elements that do not directly correspond to any OpenMath objects. For example, the *OMATP* element is used to encode a set of attribute / value pairs. This introduces unnecessary overhead into JSON, as an array of values could be used instead.
- Many languages use JSON-like structures to implement structured data types. Thus it stands to reason that an OpenMath JSON encoding should also provide a schema to allow languages to implement OpenMath easily. This is not the case for a JSONML encoding.

OpenMath-JS The openmath-js [6] encoding takes a different approach. It is an (incomplete) implementation of OpenMath in JavaScript and was developed by Nathan Carter for use with Lurch Math on the web. It is written in literate coffee script, a derivative language of JavaScript. EdN:6

In this encoding, the term plus$(x, 5)$ would correspond to:

```
{
  "t":"a",
  "c": [
    {
```

[6] EdNote: cite https://github.com/lurchmath/openmath-js

```
        "t":"sy",
        "cd":"arith1",
        "n":"plus"
      },
      {
        "t":"v",
        "n":"x"
      },
      {
        "t":"i",
        "v":"5"
      }
   ]
}
```

This encoding solves some of the disadvantages of the JSONML encoding, however it still has some drawbacks:

- It was written as a JavaScript, not JSON, encoding. The existing library provides JavaScript functions to encode OpenMath objects. However, the resulting JSON has only minimal names. This makes it difficult for humans to read and write directly.
- No formal schema exists, like in the JSONML encoding.

[7]Given these disadvantages, it makes sense to instead create a new OpenMath JSON encoding. This encoding should combine the advantages of the above two. EdN:7

In particular, it should be close to the XML encoding, and at the same time make use of native JSON concepts. Furthermore, we want to formalize this encoding, which is not achieved by any existing approach, and thus enable easy validation.

[7]EdNote: Re-formulate this

2 The OpenMath-JSON encoding

2.1 Architecture

2.2 Encoding Details

2.3 The Demo Site

To demonstrate our OpenMath-JSON encoding, we have created a demo site which can be found at [8]. This site is implemented in TypeScript[9] and encapsulated using Docker[10]. EdN:8 EdN:9

The demo site serves three purposes. EdN:10

Primarily, it serves as a presentation of the encoding, providing examples and documenting it's usage.

Secondly, it enables validation of OpenMath JSON objects. This can be seen in Figure[11]. The user can enter some JSON, press the *Validate JSON* button, and receive immediate feedback if their JSON is a valid OpenMath object or not. In particular, the user can also see a detailed error message if their object is not valid OpenMath JSON. EdN:11

This makes use of the OpenMath JSON schema, and validates the users' JSON using a generic JSON Schema Validator. Furthermore, this is also exposed using a REST API, enabling easy validation of OpenMath JSON in other applications.

Thirdly, it enables translation between XML and JSON OpenMath objects. Like for validation, the site enables the user to enter some JSON and be presented with some XML and vice-versa. This can be seen in Figure[12]. EdN:12

As we designed our encoding with this translatability goal in mind, the implementation of it was straight-forward. Nonetheless, this translation is also exposed using a REST API. [13] EdN:13

[8] EdNote: cite http://omjson.kwarc.info
[9] EdNote: cite typescript
[10] EdNote: cite docker
[11] EdNote: Make figure
[12] EdNote: Make figure + screenshot
[13] EdNote: More

3 Conclusion

In this paper we have established that an OpenMath JSON encoding enables using OpenMath in many programming languages. The existing approaches for such an encoding did not make use of many of the native JSON features, hence we have developed our own encoding. This encoding is both easily translatable to and from the JSON encoding and makes use of native JSON features.

[14] EdN:14

[14] EdNote: Write more