A Proposal for an OpenMath JSON Encoding
OpenMath workshop CICM 2018

Tom Wiesing

October 2, 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. 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.

JSON, short for **J**ava**S**cript **O**bject **N**otation, is a lightweight data-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`),

[1] EdNote: Citations
[2] EdNote: Flow

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, and 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.

1.1 Existing Approaches

There are existing approaches for encoding OpenMath as JSON. These have been discussed and suggested on the OpenMath mailing list, we will discuss two particular ones here.

XML as JSON The JSONML standard [3] 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 plus$(x, 5)$ would correspond to: EdN:3

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

[3] EdNote: http://www.jsonml.org/

```
    ]
]
```

This translation has the advantage that it is near-trivial to translate between the XML and JSON encodings of OpenMath. However, it also comes with 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 [4] encoding takes a different approach. EdN:4
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.

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

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

[4] EdNote: 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.

1.2 Goals for our OpenMath encoding