Comparison of Syntax Formats for the Physics Derivation Graph

Ben Payne[1]*, Michael Goff[2]
[1]*Department of Fun, University Name & Town, city, State Zip*
[2]*Department of Mathematics, University of Maryland & Baltimore 21228*

January 2, 2015

Abstract

A comparison of candidates for how to express elements of the graph

1 Introduction

The Physics Derivation Graph is an project designed to capture mathematical physics knowledge.

In this report, we compare methods of capturing mathematical syntax required to describe derivations in physics. This survey covers LaTeX, Mathematical Markup Language (MathML)[1], Mathematica[2], and SymPy[3]. For MathML, both Presentation and Content forms are included.

This comparison is between syntax methods which do not serve the same purpose. LaTeXis a typesetting language, while Mathematica and SymPy are Computer Algebra Systems (CAS). The reason these were picked is twofold: what is typically used in the scientific community and what is needed for the Physics Derivation Graph.

Consumer cares about previous experience, how wide spread in their community, speed, ease of input, presentation (rendering), ability to access content across devices, OS independence, ease of setup.

Evaluate criteria relevant to users, including the ability to manually input syntax (section 3), the ability to transform between representations (section 4.1), and the ability to audit correctness (section 4.2).

We can ignore syntax methods which do not support notation necessary for describing physics. Example of this include ASCII[4] and HTML[5]. Storage of the generated content (essentially a knowledge base for all of physics) isn't expected to exceed a Gigabyte, so compactness in terms of storage isn't a criterion in this evaluation.

2 Test Cases

In order to demonstrate the variety of uses in distinct domains of Physics, a set of test cases are provided in this section. These cases are not meant to be exhaustive of either the syntax or the scientific domain. Rather, they are examples of both capability requirements of the Physics Derivation Graph and of the syntax methods.

Case 1 is a second order polynomial. Algebra

$$ax^2 + bx + c = 0 \tag{1}$$

Case 2, Stoke's theorem, includes integrals, cross products, and vectors. Calculus

$$\int\int_{\sum} \vec{\nabla} \times \vec{F} d\sum = \oint_{\partial \sum} \vec{F} d\vec{r} \tag{2}$$

Case 3: Tensor analysis. Einstein notation: contravariant = superscript, covariant = subscript. Used in electrodynamics

$$Y^i(X_j) = \delta^i{}_j \tag{3}$$

Case 4 covers notation used in Quantum Mechanics. Symbols such as $\hbar$ and Dirac notation are typically used.

Case 4a is the creation operator

$$\hat{a}^+|n\rangle = \sqrt{n+1}|n+1\rangle \tag{4}$$

Case 4b is the uncertainty principle

$$\sigma_x \sigma_p \geq \frac{\hbar}{2} \tag{5}$$

Case 4c: Lüders projection

$$|\psi\rangle \rightarrow \sum_n |c_n|^2 P_n, \text{ where } \mathrm{P_n} = \sum_{\mathrm{i}} |\psi_{\mathrm{ni}}\rangle\langle\psi_{\mathrm{ni}}| \tag{6}$$

3 Quantitative Comparison of Test Cases

*Corresponding author: ben.is.located@gmail.com

Table 1: Character Count of Test Cases

Name	case 1	case 2	case 3	case 4a	case 4b	case 4c
Latex	20	101	26	45	39	110
PMathML	324	538	348	372	250	
CMathML	381					
Mathematica						
SymPy						

Character count for the MathML was carried out using

```
wc -m mathML_presentation_case*.xml
```

4 Qualitative Comparisons of Syntax Methods

LaTeX, MathML, and SymPy are free and open source. Mathematica is proprietary and not free.

For Physicists comfortable writing journal articles in LaTeXor exploring ideas in Mathematica, these are natural syntax methods. Both LaTeXand Mathematica are concise, making them intuitive to read and quick to enter. MathML is a verbose syntax which is lengthy to manually enter and yield difficult to read the native XML.

Unicode is needed to support Dirac notation and any other non-ASCII text in MathML

4.1 Transform Syntax

Wolfram Research offers the ability to convert from Mathematica expressions to MathML on their site www.mathmlcentral.com

A CAS typically produces output syntax such as LaTeXor MathML in a single format. However, there are often many ways to represent the same math, *e.g.* Eq. 7.

4.2 Audit Correctness of Derivations

One reason Computer Algebra Systems such as Mathematica and SymPy were included in this survey was to address the requirement of checking correctness of derivations.

LaTeXand Presentation MathML are intended for rendering equations and are not easily parsed consistently by a CAS. For example, scientists and mathematicians often render the same partial differential operation in multiple ways:

$$\frac{\partial^2}{\partial t^2}F = \frac{\partial}{\partial t}\frac{\partial F}{\partial t} = \frac{\partial^2 F}{\partial t^2} = \frac{\partial}{\partial t}\dot{F} = \frac{\partial \dot{F}}{\partial t} = \ddot{F}. \quad (7)$$

All of these are equivalent.

5 Conclusions

This survey covered LaTeX, Mathematica, MathML, and SymPy as candidates for syntax methods to be used for the Physics Derivation Graph. Although LaTeXis intuitive for scientists and is concise, it is a typesetting language and not well suited for the web or use in Computer Algebra Systems (CAS). Mathematica is also concise and has wide spread use by scientists, though its cost limits accessibility. Mathematica is also proprietary, which limits the ability to explore the correctness of this CAS

6 Bibliography

References

[1] Mathematical markup language (mathml) version 3.0 2nd edition, 2014.

[2] Inc. Wolfram Research. *Mathematica*. Champaign, Illinois, version 10.0 edition, 2014.

[3] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2014.

[4] American standard code for information interchange, 1968.

[5] Html 4.01 specification, 1999.

A Test cases in Latex and MathML

A.1 Case 1: polynomial

$$ax^2 + bx + c = 0 \tag{8}$$

Latex:

```
a x^2 + b x + c = 0
```

SymPy:

```
#!/bin/python
import sympy
a = sympy.Symbol('a')
b = sympy.Symbol('b')
c = sympy.Symbol('c')
x = sympy.Symbol('x')

a*(x**2) + b*x + c == 0
```

Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<mrow>
  <mrow>
    <mn>a</mn>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mo>+</mo>
    <mrow>
      <mn>b</mn>
      <mo>&InvisibleTimes;</mo>
      <mi>x</mi>
    </mrow>
    <mo>+</mo>
    <mn>c</mn>
  </mrow>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
</math>
```

Content MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<apply>
 <eq/>
  <apply>
   <plus/>
    <apply>
     <ci>a</ci>
     <power/><ci> x </ci><ci> 2 </ci>
     </apply>
    <apply>
```

```
     <ci>b</ci> <ci> x </ci>
     </apply>
    <apply>
    <ci>c</ci>
    </apply>
   </apply>
   <apply>
    <cn>0</cn>
  </apply>
</apply>
</math>
```

A.2 Case 2: Stoke's theorem

$$\int \int_{\sum} \vec{\nabla} \times \vec{F} \dot{d} \sum = \oint_{\partial \sum} \vec{F} \dot{d\vec{r}} \tag{9}$$

Latex:

```
\int \int_{\sum} \vec{\nabla} \times \vec{F} \dot d\sum =
\oint_{\partial \sum} \vec{F}\dot d\vec{r}
```

SymPy:

```
#!/bin/python
import sympy
# \int \int_{\sum} \vec{\nabla} \times \vec{F} \dot d\sum = \oint_{\partial \sum} \vec{F}\dot d\vec{r}
# http://docs.sympy.org/0.7.3/tutorial/calculus.html#integrals
x = sympy.Symbol('x')
sympy.integrate(x,x) == sympy.integrate(x,x)
```

Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<mo>&#x222B;</mo>
<msub>
    <mo>&#x222B;</mo>
    <mo>&#x3A3;</mo>
</msub>
<mover>
  <mo>&#x2207;</mo>
  <mo>&rarr;</mo>
</mover>
<mo>&#x2A2F;</mo>
<mover>
<mi>F</mi>
<mo>&rarr;</mo>
 </mover>
<mi>&#xB7;</mi>
<mi>d</mi>
<mo>&#x3A3;</mo>
<mo>=</mo>
<msub>
    <mo>&#x222E;</mo>
    <mrow>
        <mo>&#x2202;</mo>
```

```
        <mo>&#x3A3;</mo>
    </mrow>
</msub>
 <mover>
<mi>F</mi>
<mo>&rarr;</mo>
 </mover>
<mi>&#xB7;</mi>
<mi>d</mi>
<mover>
<mi>r</mi>
<mo>&rarr;</mo>
</mover>
</math>
```

Content MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">

</math>
```

A.3 Case 3: Tensor analysis

$$Y^i(X_j) = \delta^i{}_j \tag{10}$$

Latex:

```
Y^i(X_j) = \delta^i_{\ j}
```

SymPy:

```
#!/bin/python
import sympy
# Y^i(X_j) = \Delta^i_{\ j}
# http://docs.sympy.org/dev/modules/tensor/tensor.html
Y = sympy.Symbol('Y')
X = sympy.Symbol('X')
```

Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<msup>
    <mi>Y</mi>
    <mi>i</mi>
</msup>
<mo maxsize='1'>(</mo>
<msub>
    <mi>X</mi>
    <mi>j</mi>
</msub>
<mo maxsize='1'>)</mo>
<mo>=</mo>
<msubsup>
    <mtext>&#x394;</mtext>
    <mrow>
        <mspace width='1em'></mspace>
        <mi>j</mi>
```

```
    </mrow>
    <mi>i</mi>
</msubsup>
</math>
```

Content MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">

</math>
```

A.4 Case 4a: creation operator

$$\hat{a}^+|n\rangle = \sqrt{n+1}|n+1\rangle \tag{11}$$

```
\hat{a}^+ |n\rangle = \sqrt{n+1} |n+1\rangle
```

SymPy:

```
#!/bin/python
import sympy
# \hat{a}^+ |n\rangle = \sqrt{n+1} |n+1\rangle
# http://docs.sympy.org/dev/modules/physics/secondquant.html
n = sympy.Symbol('n')
sympy.sqrt(n+1)
```

Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<msup>
<mover>
    <mtext>a</mtext>
<mo>&#x5E;</mo>
</mover>
    <mo>&#x2020;</mo>
</msup>
<mo maxsize='1'>|</mo>
<mi>n</mi>
<mo maxsize='1'>&#x232A;</mo>
<mo>=</mo>
<msqrt>
    <mi>n</mi>
    <mo>+</mo>
    <mn>1</mn>
</msqrt>
<mo maxsize='1'>|</mo>
<mi>n</mi>
<mo>+</mo>
<mn>1</mn>
<mo maxsize='1'>&#x232A;</mo>
</math>
```

Content MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">

</math>
```

A.5 Case 4b: uncertainty principle

$$\sigma_x \sigma_p \geq \frac{\hbar}{2} \tag{12}$$

```
\sigma_x \sigma_p \geq \frac{\hbar}{2}
```

SymPy:

```
#!/bin/python
import sympy
# \sigma_x \sigma_p \geq \frac{\hbar}{2}
# http://docs.sympy.org/dev/_modules/sympy/physics/qho_1d.html
```

Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<msub>
    <mtext>&#x3C3;</mtext>
    <mi>x</mi>
</msub>
<msub>
    <mtext>&#x3C3;</mtext>
    <mi>p</mi>
</msub>
<mo>&#x2265;</mo>
<mfrac>
    <mtext>&#x210F;</mtext>
    <mn>2</mn>
</mfrac>
</math>
```

Content MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">

</math>
```

A.6 Case 4c: Lüders projection

$$|\psi\rangle \rightarrow \sum_n |c_n|^2 P_n, \text{ where } \mathrm{P_n} = \sum_{\mathrm{i}} |\psi_{\mathrm{ni}}\rangle\langle\psi_{\mathrm{ni}}| \tag{13}$$

```
 |\psi\rangle \rightarrow \sum_n  |c_n|^2 P_n,\ \rm{where}\ P_n = \sum_i |\psi_{ni}\rangle \langle \psi_
```

SymPy:

```
#!/bin/python
import sympy
#  |\psi\rangle \rightarrow \sum_n  |c_n|^2 P_n,\ \rm{where}\ P_n =
 \sum_i |\psi_{ni}\rangle \langle \psi_{ni}|
omega = sympy.Symbol('omega') # from outputs
f = sympy.Symbol('f') # from inputs
pi = sympy.pi
```

Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">

</math>
```

Content MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">

</math>
```