**Notes on JSON AD Graph**

Dr. Jeremy A. Bloom  
[jeremyblmca@gmail.com](mailto:jeremyblmca@gmail.com)  
June 22, 2019

These are my thoughts on *ADFun Object Corresponding to Json AD Graph : Example and Test* <https://coin-or.github.io/CppAD/doc/json2adfun.cpp.htm> by Brad Bell.

An example of the proposed JSON representation of the graph

*An AD graph example*

*node\_1 : x[0]*

*node\_2 : x[1]*

*node\_3 : "x"*

*node\_4 : "y"*

*node\_5 : -2.0*

*node\_6 : x[0] + x[1]*

*node\_7 : (x[0] + x[1]) \* (x[0] + x[1])*

is the following JSON file:

{  
 **"n\_dynamic\_ind"** : 0,  
 **"n\_independent"** : 2,  
 **"string\_vec"** : [ 2, [ **"x"**, **"y"** ] ],  
 **"constant\_vec"** : [ 1, [ -2.0 ] ],  
 **"operator\_vec"** : [ 2, [  
 [ 0, 1, 2, [1, 2], **"add"** ] ,  
 [ 1, 1, 2, [6, 6], **"mul"** ] ]   
 ],  
 **"dependent\_vec"** : [ 1, [7] ]  
}

While this effort is a welcome first step toward an important capability, there are several aspects of this proposal that are problematic.

1. It is difficult to see how it could be adopted within the current proposal for MOSDEX; which should be a goal, since as currently specified, MOSDEX can only handle linear expressions (although extension to quadratic expressions is straightforward).
2. The interpretation of the various elements of the JSON representation is unclear, so a prose explanation is needed.
3. The use of JSON arrays, which don’t have labels on the elements, makes it hard to understand what each element means. These should be changed to JSON Objects, and field names should be added to clarify how each field should be interpreted.
4. On a related issue, key words in the spec, including the field names discussed in item 3, should be distinguished. MOSDEX uses key words written in all caps, with underscores and hyphens permitted inside. Thus **n\_dynamic\_ind** would be written as **N\_DYNAMIC\_IND**.
5. It appears that the JSON Arrays **"string\_vec"**: [ 2, [ **"x"**, **"y"** ] ] and the other vectors have as their first component the number of elements in the array. This is unnecessary, and in fact, it violates model/data separation. Suggest writing instead something like   
   **"VARIABLES"**: [ **"x"**, **"y"** ] and so on.
6. The use of subscripts to index variables and other vector elements is inconsistent with MOSDEX. MOSDEX treats elements with one or more dimensions as tables, in the sense used in the relational data model. Thus, for instance, MOSDEX would define the variable **"x"** like this:  
     
   **"x"**: {  
    **"CLASS"**: **"VARIABLE"**,  
    **"SCHEMA"**: {  
    **"key"**: **"INTEGER"**,  
    ...  
    **"Value"**: **"DOUBLE"** },  
    **"IMPORT\_FROM"**: [...]  
   }
7. How the graph is represented in this example is totally obscure. It appears that it is embodied in the two dimensional array **operator\_vec** but the interpretation of its elements is opaque. It looks like *node\_6 : x[0] + x[1]* is represented by the first row of this vector, but where does this row identify it as node 6? The second row appears to represent *node\_7 : (x[0] + x[1]) \* (x[0] + x[1])* , but where are the other nodes represented?
8. It would seem a better way would be to define a **NODE** class like this:  
     
   **"0"**: {  
    **"CLASS"**: **"NODE"**,  
    **"SCHEMA"**: {  
    **???** },  
    **"INSTANCE"**: [  
    [0, 1, 2, [1, 2], **"add"** ]

]  
}  
  
The Schema needs to identify the items in the Instance records by name and type. Also, for clarity, nodes should be given names rather than index numbers, and Arrays within Arrays like [1, 2] should be avoided; MOSDEX allows only strings, integers, and doubles as items in an Instance record.

1. It would be helpful to have a fully articulated example of how to build a non-linear optimization problem using the proposed graph objects. To this end, there is a quadratic programming example in MOSDEX that could be used as the basis for extension. See <https://github.com/coin-modeling-dev/MOSDEX-Examples/blob/master/trafficNetworkQPNew.json>