Automatic looping in VisTrails

This work follows a comparison of VisTrails with Taverna. The idea is that applying a simple operation to a list of values right now requires the use of the controlflow package, which is awkward (takes modules as input: doesn't feel like a dataflow).

An idea is to allow modules to loop automatically when they are provided with a list of values.

Since we are looking into adding loops into VisTrails's core, another feature we had in mind is streaming for these loops.

Concept

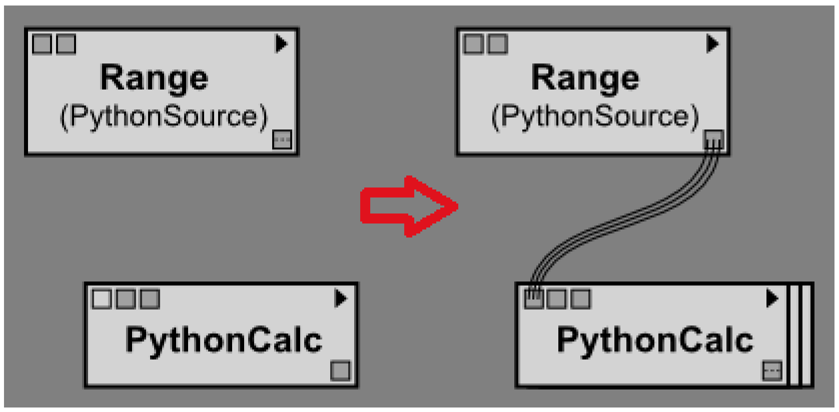
Instead of introducing a List type for ports (or a new one, to represent 'list-of-sometype'), use a flag on the port to define what cardinality the module expects on a port:

- a single value (what every "old" module does). If it gets a list instead, the module will loop (compute() will be called several times), and the outputs will be lists (this is the MAP operation). This would also happen when connecting multiple
- multiple values: mimics the mechanism we have with getInputListFromPort(). The module is still called only once with the whole list. Can be a FOLD operation.

The same thing would go for output ports:

- a single value, which becomes multiple values if this module turns into a map
- multiple values

It should be possible to statically determine what will be carried on a port/connection, and thus to display differently the ports, connections and modules that get loops.



Of course this whole logic can recurse, which means we can get "lists of lists".

Example, a module expecting a list and receiving a list of lists would automatically loop to compute() each list separately.

Implementation

List depths

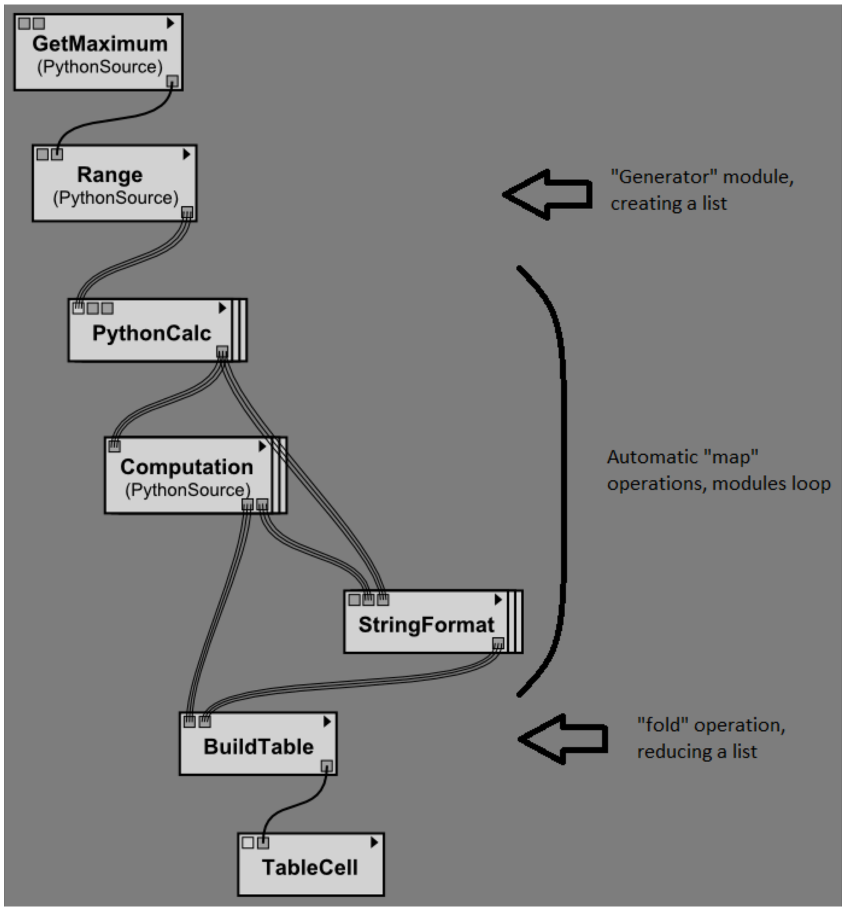
Use a list "depth" parameter on port specs. Current ports would all have depth=0 (single value), a new parameter would allow to choose a different depth. This is what Taverna does.

In the pipeline editor, walk the graph to find the list depth on each connection, and the depth of each MAP operation. For MAP operations (which we need to display differently), we need additional configuration parameters: how to combine the inputs (dot or cartesian product) and in what order (in Taverna: "list handling").

At run-time, we should no longer pass a raw object but rather a wrapper that knows the shape (depth and list of sizes) of the data, so that the receiving module can return an output of the appropriate shape by calling compute() the correct number of times.

Streaming

Because we are now passing specialized objects on the connections, and these objects are not manipulated directly by the package author but iterated over by the interpreter as it calls compute(), it is possible for the values to be lazy generator instead of lists (vectors of ready-for-use values). The only issue here is the logger and propagation of exceptions.


GetMaximum
(PythonSource)
Range
(PythonSource)
PythonCalc
Computation
(PythonSource)
StringFormat
BuildTable
TableCell
"Generator" module, creating a list
Automatic "map" operations, modules loop
"fold" operation, reducing a list