Interactive Variable Network: Tool Specification

Synopsis

The idea is to create a visually pleasing frontend to be able to interact with a system of equations relating variables representing aspects of a business process (examples below). The user should be able to modify the values of the variables, and observe the consequential value changes on other variables.

Simple example

Probably better that I explain this starting with a simple example. The formula for this example is:

$z = x + y$

So there's three variables (x, y, and z). If you modify x and y, z will change to the value x + y. (Like in http://stackoverflow.com/questions/15648711/angular-js-adding-two-numbers-issue for example). How it would actually work is if you change the value of x, z will be reset to be x + y, where the current value of y would be used, and then you would change y if you want.

Only thing, is we want to be able to change any of the three variables. So what happens when the user changes z? Well, x = z - y, and y = z - x, but the question is: Do we change x (using the fixed value of y to compute it) or do we change (using the value of x to compute it)?

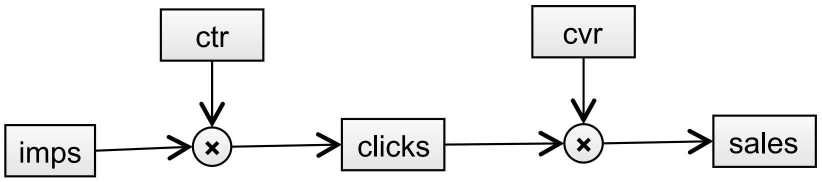
There's several ways to solve this. I propose (for now), to allow each "variable box" to be clicked & dragged (like in http://jqueryui.com/sortable/#default or http://jqueryui.com/sortable/#placeholder) to allow the user to sort the variables.This order then specifies what to do in such ambiguous cases. The rule being that the variables that are lower should be changed in priority. So in the case just explained, if the order was [x,y,z], changing z would change y (with formula y = z - x) since y is "lower" in the list. (I can send you an pseudo-code algorithm for this if we get to that...)

Now that's the three variable case, but I'd like to have it with more variables.

A bit bigger (and useful) example

Let me give you a real life example. Take a network that describes how you get a sale from an ad. The use case this network models is on where an ad is displayed, a user clicks on it and lands on a page where he buys something. So... an ad has some number of impressions (imps) and a click-through rate (ctr), producing a number of clicks (clicks = imps * ctr). Clicks lead to a page with some things to buy in them, with a conversion rate (cvr). You get your number of sales as sales = cvr * clicks.

Below is a visual representation of this situation:



The whole network is specified by two equations:

clicks = ctr * imps

and

sales = cvr * clicks

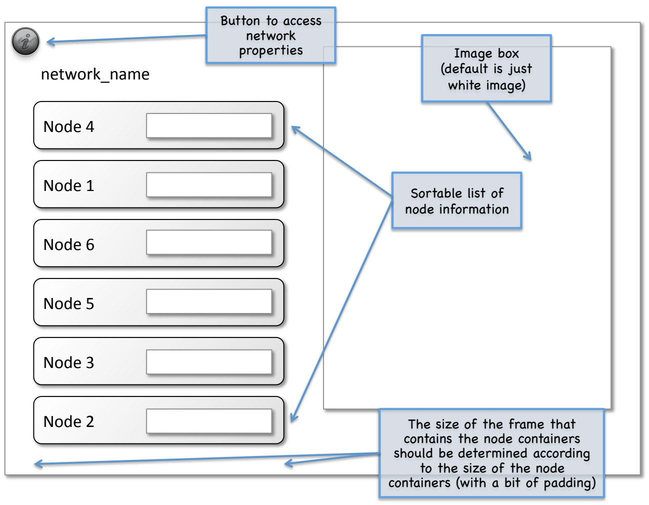
But to function, the network will also have to able to compute, say, ctr from imps and clicks.

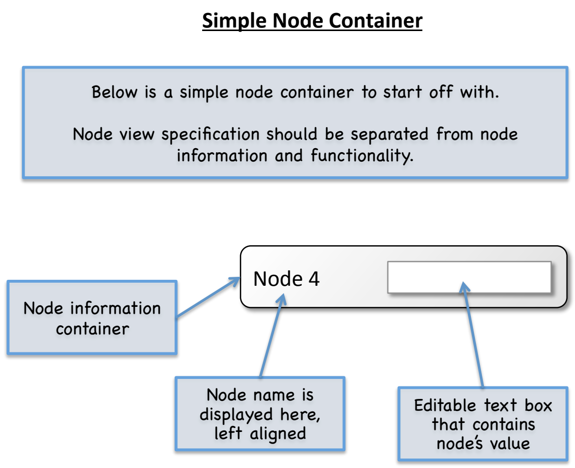
We could have the code figure it out, but to stay simple, for now, we will explicitly specify each equation, namely:

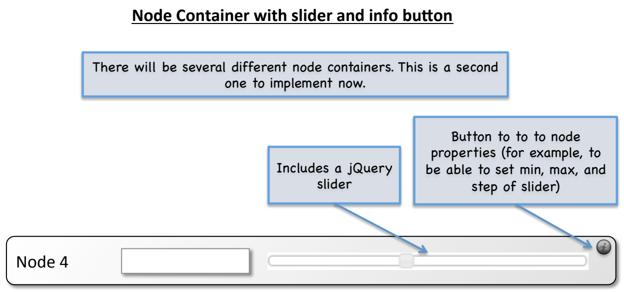
- clicks = ctr * imps, and therefore
  - ctr = clicks / imps
  - imps = clicks / ctr
- sales = cvr * clicks, and therefore
  - cvr = sales / clicks
  - clicks = sales / cvr

Further, you can see a bigger (and even more useful) example.

View


Button to access network properties
network_name
Image box (default is just white image)
Node 4
Node 1
Node 6
Node 5
Node 3
Node 2
Sortable list of node information
The size of the frame that contains the node containers should be determined according to the size of the node containers (with a bit of padding)


Simple Node Container
Below is a simple node container to start off with.
Node view specification should be separated from node information and functionality.
Node 4
Node information container
Node name is displayed here, left aligned
Editable text box that contains node's value


Node Container with slider and info button
There will be several different node containers. This is a second one to implement now.
Includes a jQuery slider
Button to to to node properties (for example, to be able to set min, max, and step of slider)
Node 4

Proposed Control/Model Implementation

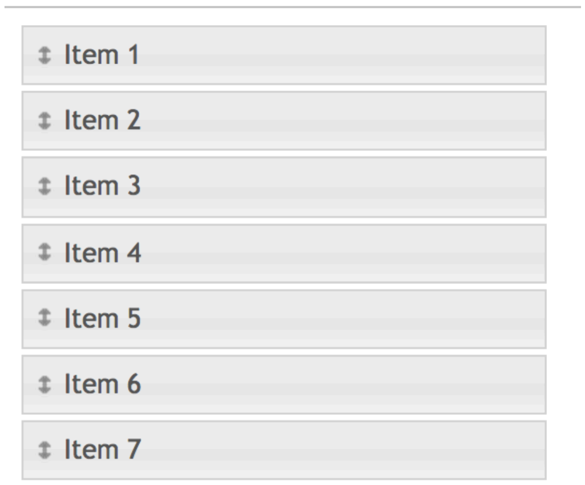
It's more important now to get a network that work (that is, that computes), so we won't spend any time on an interface for the construction of networks. Rather, a network will be specified by a json. I propose the following (example based on the five variable example above):

```
{
        'clicks': {'links': [{'inputs': ['imps', 'ctr'], 'relation': 'mu
ltiply'},    {'inputs': ['sales', 'cvr'], 'relation': 'divide'}],   'va
lue': 50.0},
        'ctr': {'links': [{'inputs': ['clicks', 'imps'], 'relation': 'di
vide'}],   'value': 0.05},
        'cvr': {'links': [{'inputs': ['sales', 'clicks'], 'relation': 'd
ivide'}],   'value': 0.02},
        'imps': {'links': [{'inputs': ['ctr', 'clicks'], 'relation': 'mu
ltiply'}],   'value': 1000.0},
        'sales': {'links': [{'inputs': ['clicks', 'cvr'], 'relation': 'm
ultiply'}],   'value': 1.0}
}
```

Note that the "links" for clicks has two entries because it's connected to two variable clusters.

From the point of view of the View, there should be a function that specifies how to display a variable. I propose some kind of box containing the variable name, and it's value (later, I would like to have a slider to change the value, and a "settings" button to be able to change the sliders dimensions).

The variables are displayed in some order, as in a typical jQuery sortable list (http://jqueryui.com/sortable/):



The order of the variables will determine what functions will be used to update the values of variables when the user updates the value of some other variable.

Here’s the algorithm (or at least one possible one):

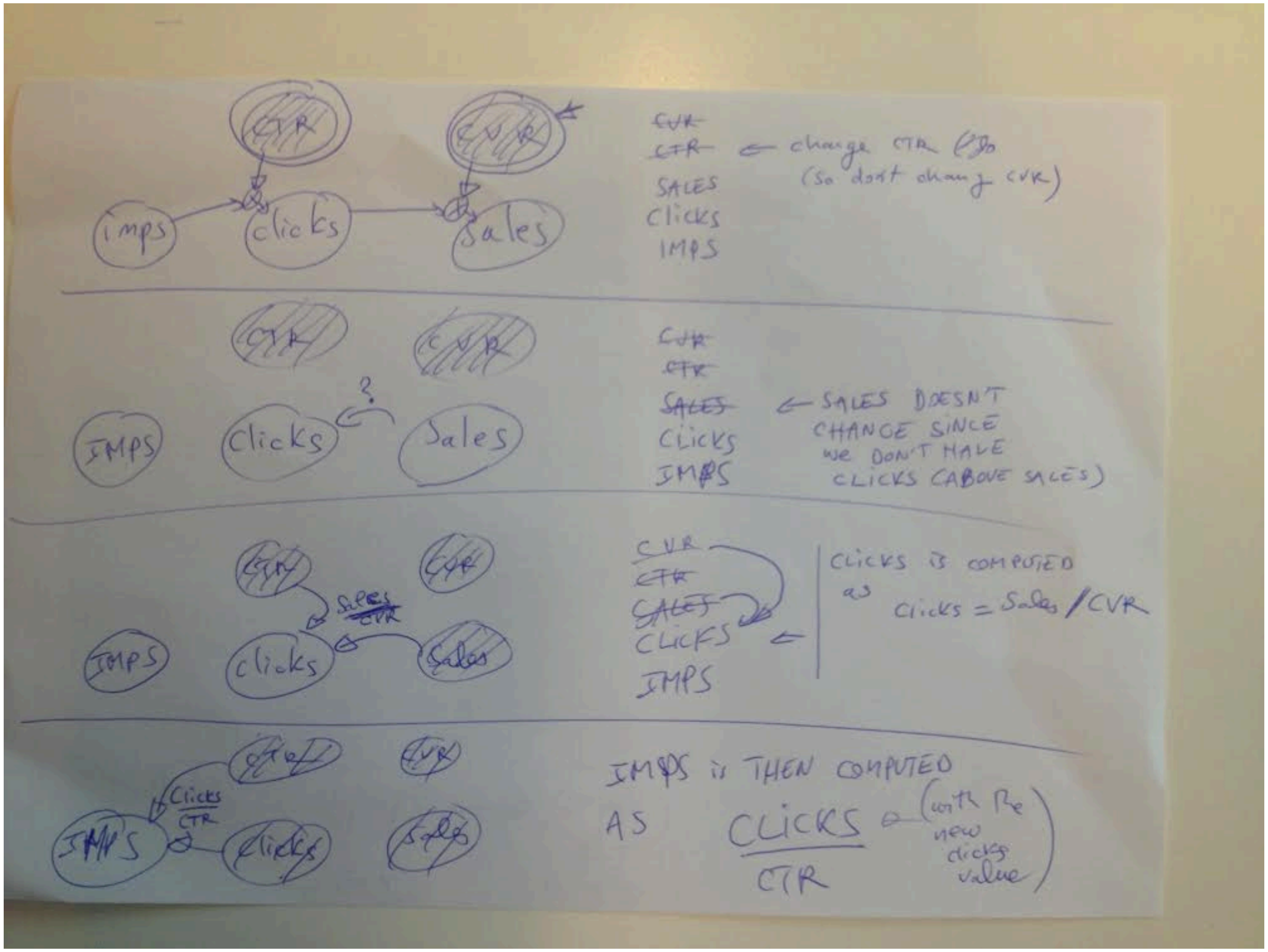
Go through each node starting from the node just below the one that changed (so all the nodes above should not change), and for each such node X, if the “inputs” X[‘input’] are all “above” X (in the current node order), you change the value of X to be the result of applying X[‘function’] to the values of X[‘input’], and then continue looping.

So in the loop there’s something like

```
        if X[‘inputs’] subset of all_nodes_above(X):
                change the value of X
        else continue looping
```

See below a workout example of the algorithm:

The graph below is a larger example of a (useful) network of variables



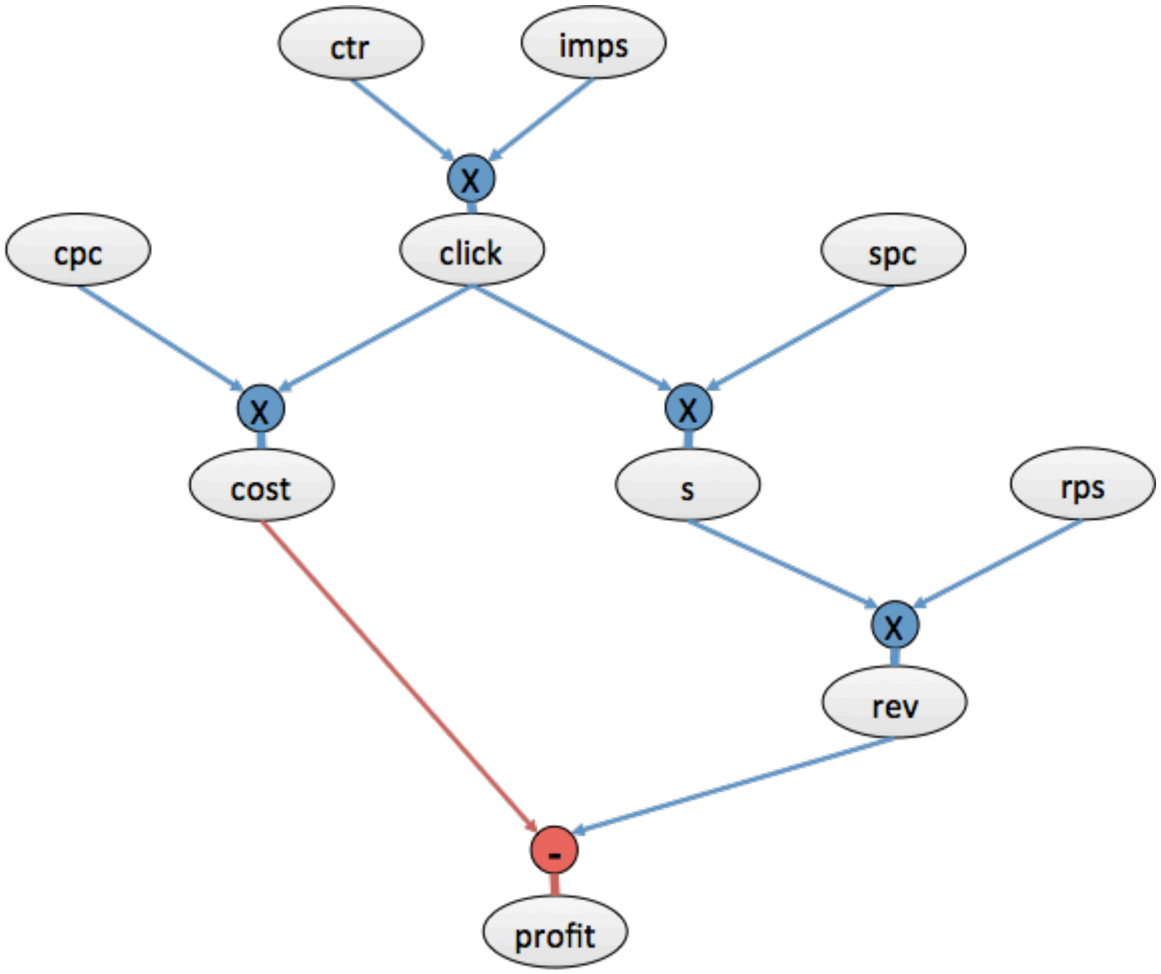
~~CVR~~
~~CTR~~ ← change CTR [illegible]
(so don't change CVR)
SALES
Clicks
IMPS

~~CVR~~
~~CTR~~
~~SALES~~ ← SALES DOESN'T CHANGE SINCE WE DON'T HAVE CLICKS (ABOVE SALES)
CLICKS
IMPS

~~CVR~~
~~CTR~~
~~SALES~~
CLICKS ←
IMPS

CLICKS IS COMPUTED AS Clicks = Sales / CVR

IMPS IS THEN COMPUTED AS $\frac{\text{CLICKS}}{\text{CTR}}$ ← (with the new clicks value)



Where:

- imps: impressions
- spc: sale per click (e.g. conversion rate)
- s: sale (e.g. actual sale, or paid click-out)
- rps: revenue per sale (for example “average basket”, commission obtain therefrom, or revenue obtained from a click-out, etc.)