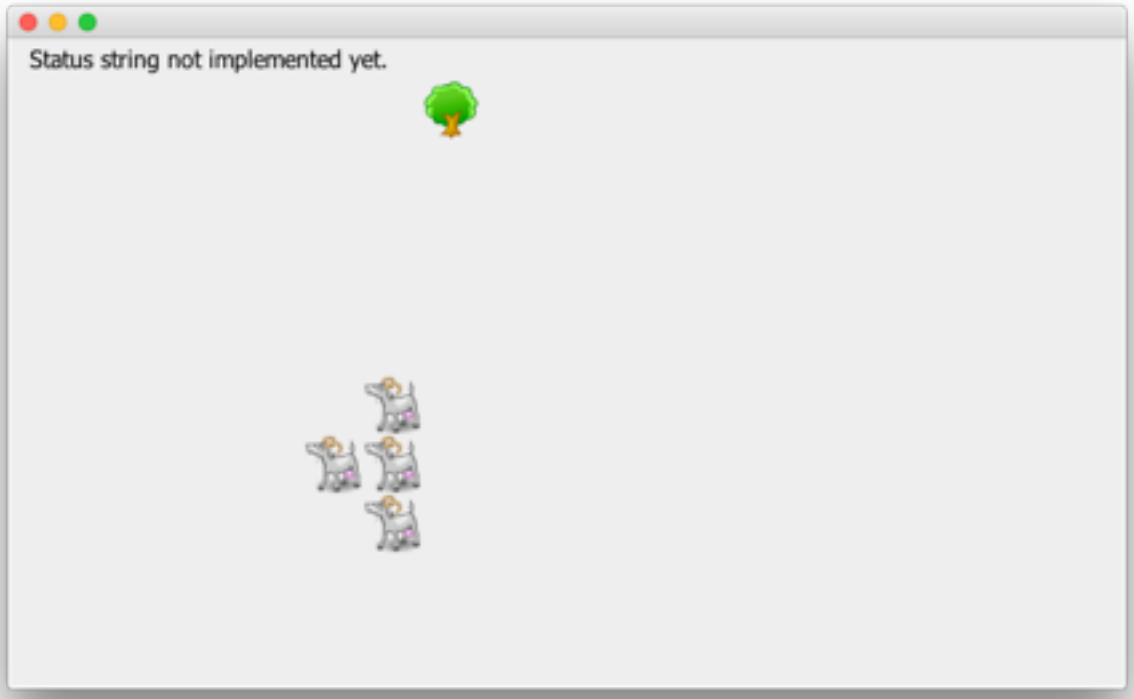
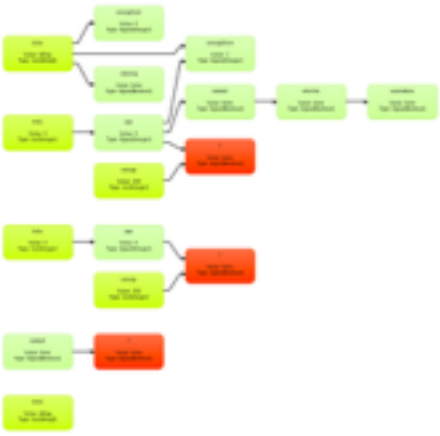
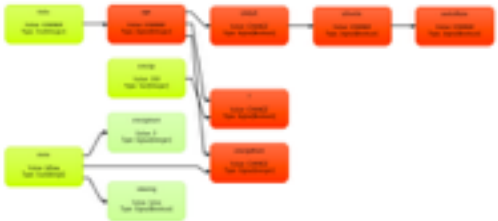
Reactive Tree View

Universe Case Study



General Observations





- The structure works very well. The different reactive variables are sorted by connectedness and are presented in a way to easily understand the dependencies between them.
- The “Highlight Change” function shows how a change in value influences the different variables. This feature is very useful to see how different reactive variables react to the environment in which they are evaluated.
- The live update of the tree view nicely visualizes the correlation between the variables.

Possible Improvements

- Using the scrolling wheel of the mouse to zoom confuses the user when trying to scroll down. The scrolling wheel should be used for scrolling and zooming should only be possible by either double clicking or using the zoom buttons on top of the graph.
- Reactive variables that do not yet influence other variables are shown at the bottom of the graph (picture on the right). Although this is semantically correct, these variables could be either hidden (which would be bad semantically) or be placed on top or right of the main network of nodes.
- The name of the reactive variables in the vertices should be bigger. At the moment, the names cannot be easily differentiated.
- The graph “flickers” when updating. This is due to the fact that it is updated everytime a new reactive variable enters the debugger. Either the process of visualizing the graph is slowed down (wait after every refresh), which mains longer debugging time, or it is only updated once the debugger is done.
- Navigation inside the graph in general feels “murky”. Clicking and dragging the mouse to move around the graph might help with this.
- One has to manually integrate a new REScalaLogger into every implementation to enable debugging with the reactive tree view. Maybe a direct integration would improve the usability.

Strengths

- Although the Universe case study is not complex in itself, the reactive tree view greatly helped to understand the dependencies between the variables. Whereas in the code, one has to hop from definition to defintion to trace one dependency path, the graph allows to understand the relationships in one view.
- Folding vertices in the graph highly reduces its complexity and allows developers to better understand certain parts of the code.
- Showing the values of the reactive variables inside the graph allows developers to understand how one variable changes another even more.