IRIS VISUALIZATION

This document describes the results and development process of the *Iris Visualization*, an example of a multi coordinate visualization which can be described by a State Chart.

Introduction

A *multi coordinate view* can be an example of how different widgets can be used within the same visualization. In this particular case, we can recognize three different visual components:

1. *a scatter plot*, where the user can perform a filtering by zoom in / zoom out and panning interactions
2. *a barchart*, where the user can perform a filtering by selecting a particular bin;
3. *a range slider*, where the user can perform a filter by dragging the handlers or the corresponding selected region.

Each visual component corresponds to a particular widget within the finite state machine.

Design

To realize the *Finite State Machine* (*FSM*) of this visualization, we can use different *Hierarchical State Machines* (*HFSM*) to define separately each component: this approach is useful since it can model transitions within a particular widget without exposing them to the others, and also marking the particular transitions that are necessary to reach and subsequently leave a particular widget.

XState

Xstate is a library that can build an *SCXML model* of an automata by using this approach, and with the *Xstate Inspection Tool* it's possible to visualize in real-time the behaviour of the FSM attached to the page. Using an integrated data model, along with guarded/conditional transitions, it is possible to check within the FSM when we can enter/exit different states which require a particular condition to be satisfied. (i.e. to reach the "right" state on the range slider, it’s necessary that the new value dragged by the user with one of the handlers is greater than the previous value).

First Implementation

In the *first implementation* (*I1*) of the visualization (Fig. 1), each time an *HSFM* triggers the event "*mouseout*", it returns to the initial state "*rest*". When a transition "*mouseover*" occurs, along them a string containing the hovered visualization will be passed to the data model. "*mouseover*" is implemented as a self-transition, so when the state is re-entered different

conditions upon the current visualization hovered are checked: If one of them is satisfied, then an eventless-transition will reach the currently hovered visualization.

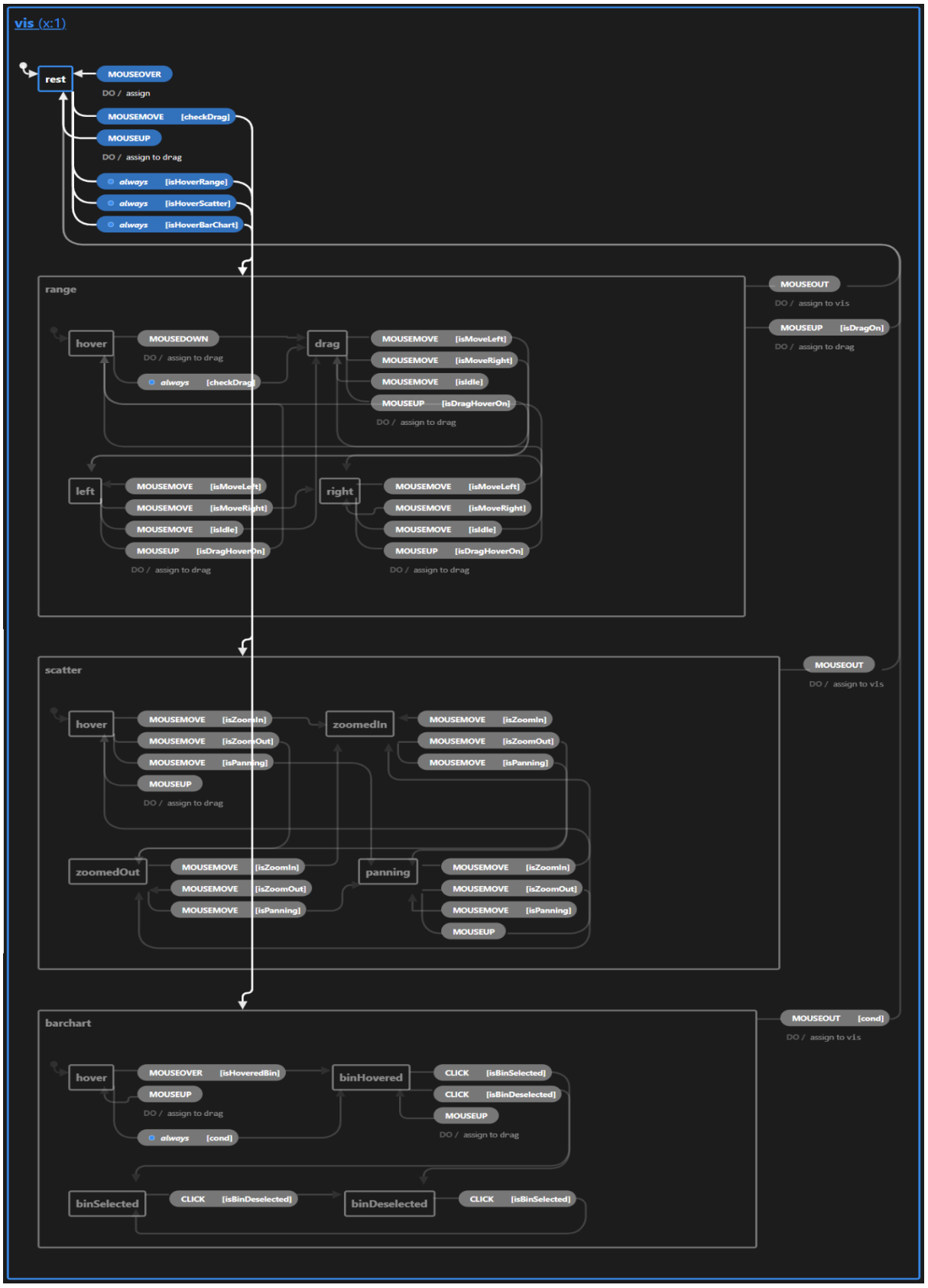

Figure 1: First implementation (*I1*) of the FSM

One issue raised with this implementation is when different widget interacts concurrently. This behaviour can be seen clearly during a mousemove event: in fact, the user can move the mouse freely around the visualization, even when some object is panned/dragged, and trigger a "*mouseup*" transition inside all the other widgets. For this reason, we've added a "*mouseup*" transition event inside all the widgets, which updates the context of the FSM.

Second Implementation

While the I1 captures correctly the behaviour of the user, it lacks in showing precisely what a widget should do. In fact, the "*mouseup*" transition is not an authorized transition for many states, such as "*hover*", but it was added to capture some events that may be raised by the users. These events, however, are strictly related to web browser implementations, thus they should not be captured by the *FSM* and ignored, i.e. a "mouseup" transition made by the range slider *HSFM* should be seen only within the range slider *HSFM* itself. Also, the drag flag used in the context of *I1* has been removed from context since the transitions now are filtered based on the current state reached by the user. (Fig.3)

For this reason, the *second implementation* (*I2*) adds a new section (Fig. 2) on top of the visualization which acts as an inspector of the *FSM* attached to the page by showing the following informations:

- the current state of the *FSM*;
- the last transition raised by the user;
- the set of current transition that can be done from the current state: if a transition is raised and is not in the available transitions, it won't be noticed by the *FSM*;
- the states that can be reached from all the available transitions

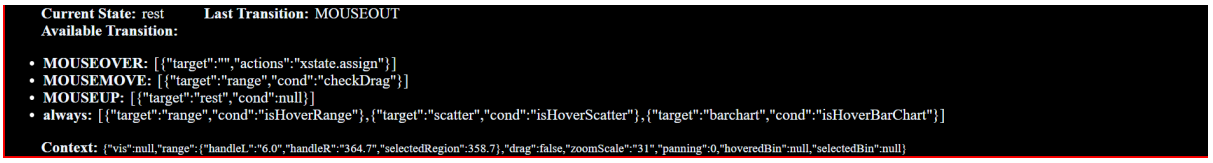

Figure 2 : I2 section showing FSM informations

Regarding the visualization, *I2* introduces a new version of the visualization written in *Vega*: using this new approach, it was possible to better define in the *FSM* some interactions realized by the user: in particular, inside the scatter plot, the *panning*/*zoom* states can be reached by *wheel*/*mousedown* events respectively.

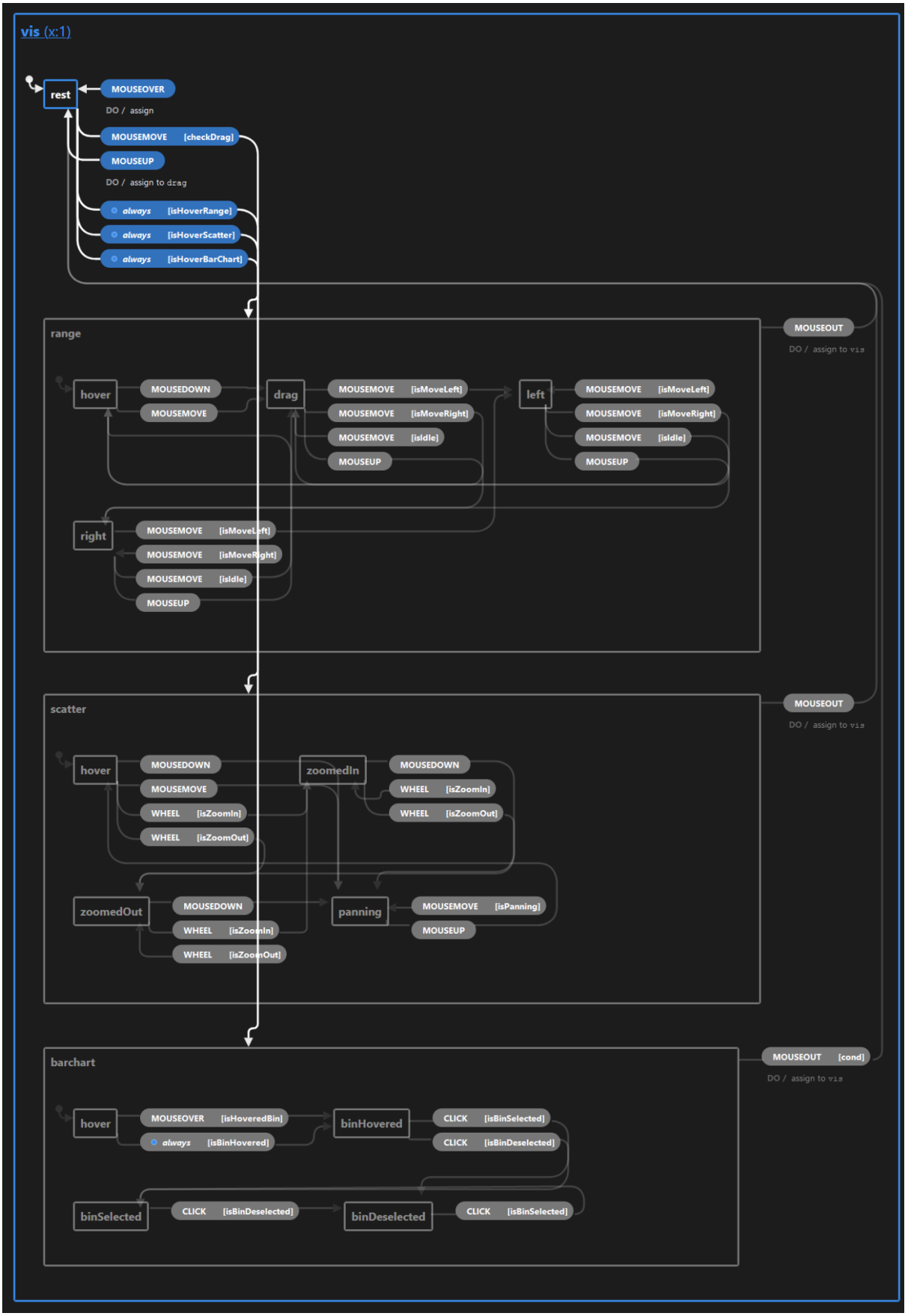

Figure 3: Second Implementation (*I2*) of the FSM