Primitive Feedback:

- Having to assign variables every time the state changes can lead to a cluttered scenario. One suggestion is to consider introducing primitives like start_assign() and end_assign(). For example: the start_assign() primitive could continue assignin the variables without manually doing so between each state until encountering the end_assign() primitive, then it stops assigning the variable.
- When working on states for creating a request packet or extracting the information from a packet and crafting a reply, multiple primitives are used. To use less primitives, it could be beneficial to combine certain primitives. For instance, combining create packet, set IP destination port, and set domain name primitives into one, such as create_DNS_packet (pkt) (IP_src name).
- In the future, including small state machines, such as the ones that I created, that can be called in the main scenario with the CALL primitive can be useful for future users.

Documentation Feedback:

- Consider including a section that provides guidance on executing the same scenario simultaneously on both endpoints.
- Include a dedicated section on IP tables, focusing on how they can be utilized to reject ICMP packets.
- Rearranging the listing of primitives could enhance usability, given the current challenge of locating specific primitives. One suggestion is to maintain an ongoing list and implement filtering options for more streamlined navigation. For instance, grouping primitives by category, such as DNS primitives, could prove beneficial.
- Following the suggestion of including small state machines that can be called in the main scenario, it would be an interesting idea to include a documentation listing the state machine, their description, and the list of inputs and outputs, similar to the documentation that I included.

State Machine Designer Website Feedback (http://127.0.0.1:4200/#):

I. Bugs

- When creating a state machine, if a user generates a new model without saving it, the components will not be submitted. I suggest implementing an error prompt as soon as the user attempts to add a component without saving the model.
- There appear to be hardcoded errors in the extracted JSON file.
- The created components do not save after a refresh. To address this, consider adding a user account feature that allows users to log in and save their work securely.
- Technical issues seem to arise when attempting to submit changes to a state or event; changes will not be submitted unless I click on the blank background and then resubmit the changes.
- I am unable to restore the default state of an event after deletion.
- I am unable to submit more than one change to a state or event simultaneously.
- Upon adding a new event, it does not automatically link to the preceding state.
- Upon connecting states to events, the state/event does not seem to respond except is I hover over it at the end.
- When connecting a state to an event, the "summit info" banner pops up, causing clutter while attempting to establish connections.

II. Feedback

- To improve efficiency, consider adding a user account feature that allows users to log in and save their work securely. A user can choose to make their projects either public or private, and a user can share the project to add collaborators that can edit the state machines in real time.
- Being able to view and restore version histories.
- When zooming in and out on the diagram and scrolling left and right, consider implementing an intuitive scrolling experience. Aligning the scrolling direction with the respective button functions could enhance the overall user experience and logic. To give an example, in xstate, the scrolling direction is inverted, requiring you to click the left arrow button to move the diagram to the right, which is not very intuitive.

- Another valuable feature would be the ability to download the state machine diagram as a PDF. This downloadable PDF should encompass all events and states. Furthermore, providing users with the choice to select between landscape and portrait orientations would enhance the versatility of this feature.
- implementing a feature that allows users to create multiple state machine diagrams within a single project. This feature should enable users to extract all of these diagrams simultaneously as JSON files, which can then be conveniently downloaded as a ZIP file directly to their computer. Furthermore, I recommend including an option that allows users to extract individual files from the project. This dual-option approach would provide users with enhanced flexibility and control over their workflow.
- Allowing users to copy JSON code directly from a state machine diagram and seamlessly importing code into the finite state machine diagram. Upon code import, the corresponding state machine diagram would pop on the editor screen.
- Having an undo/redo feature in the state machine editor.
- To improve efficiency, I recommend allowing the option to rearrange the order of submitted functions. I suggest having a feature for hovering over a specific primitive in a state and dragging over to its new position. This feature is also missing in xstate, and adding it will be a very interesting advantage. This would make modifying the order of primitives much more convenient, as currently the only way to so is to delete all the primitives after the primitive I want to add.

  • While there is a feature that displays function names when adding a primitive, it could be improved given the large number of approximately 50 primitives. Adding a filtering option to sort the list based on the first letters of a primitive that the user can input in a text box would enhance usability.
- Following on the previously suggested concept of incorporating callable state machines within a scenario, an additional user-friendly feature would be to introduce a banner that appears when using the function call. This banner should provide a list of all callable machines, along with their corresponding input and output fields. This would mirror the functionality of the list of primitives that appears when attempting to add a primitive, but tailored for state machines within the context of function calls.