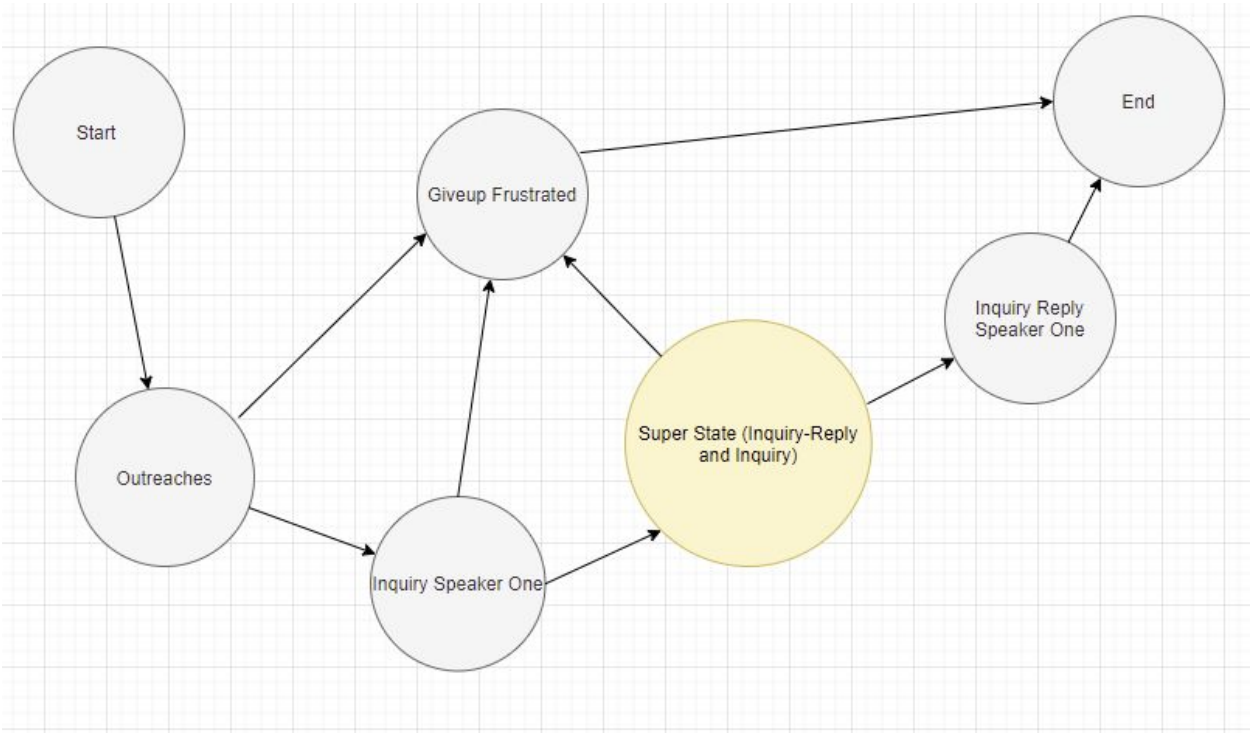
Clay Rosenthal, Archit Bose, Hari Madhu

Lab Report 4: Chatbot

Chatbot system:

The main loop of the chatbot is a state machine which incorporates 8 - 9 states that transition from a start state to an end state. The bot initially starts in a start state and either looks for a user's question for a span of 30 seconds, or a greeting from the user for the same span. If neither or these conditions are met, the bot will transition into an initial outreach state and will randomly choose a user to reach out to and introduce itself to them. If that user does not respond, we will try a second reach out after 30 seconds (secondary outreach), wait 30 final seconds and then give up on that user (give-up frustrated). It will then restart and do it all again. Alternatively the user could respond and the bot will then ask them how they are doing (inquiry state). The bot will finally wait for a response and question from the user before replying with its own response (inquiry_reply). We combined the state of the second speaker's response and inquiry (connected components/node compression) to more neatly encapsulate that both of those states are executed by the same speaker.



Phase 1 Strategy:

The strategy that was used to identify whether the bot was being spoken to was by using the python select function to poll and see if there was any information on the irc socket that was being used to connect to the server. If there was information available to be read, a recv call was created to get the byte response and was then decoded into a string. Afterwards the string was parsed to look for the bot's nickname, which had to include "-bot", and immediately followed by a colon; if the line of data did not have the specified format, the data was then ignored. If there was a valid response from the bot, the code was checked to see if it contained just the command "die" or "forget" in order to kill the chatbot or just forget all the data respectively. A class and various strings were used to store the data in the strings.

Phase 2 Strategy:

The strategy for the second phase was much more involved. We first built the framework that was much more rigid in what we could say, but followed the state machine transitions very closely as written in the spec. As the bot had to be able to act as both speakers, our state functions had two versions inside each, one for if we reach out, and one if we are being reached out to by a user. This caused a lot of complications because of the difficulty of ensuring we had the right information at the right time to be responding properly. Our decision to make a super state out of the second speaker's response and question helped us have a better flow and connect two nodes that had to input inferences between them. Once we were able to respond to a very structured conversation, we then moved on to actually understanding and normalizing the user input so that our bot was able to talk with more natural understanding of what was being asked.

Algorithms and Data Structures:

Our chatbot stored most of its data in various lists, that allowed us to reference their members to parse out common words used in questions and greetings to help us figure out what a user is saying. We normalized our text using keyword matching to see if the user was introducing themselves or asking some kind of small-talk question. To do this, words such as {What, How, And} were checked with words such as {everything, you, life} and slang phrases such as "wassup". The bot's response phrases were stored in a dictionary where it maps the keyword we normalized the text to a list of phrases. Then, if the user text corresponded to any of the response types, the bot would randomize a response from the list and display the text.