Merging and Searching within a FOON network

Joseph Gleason

***Abstract*—This paper will cover the concept of a Functional Object-Oriented Network. The main topics covered will be what is needed to create this Network, how to merge a sub-graph into an existing object-oriented network, and how to search a network to obtain a task tree for a given goal node.**

I. Introduction

THE motivation for this project was to understand and construct a Functional Object-Oriented Network, otherwise known as a FOON.[1] The entire purpose of a FOON is to create a knowledge representation for robot learning and manipulation. With such a network, robots can use the data stored in the network to perform certain actions and learn from these actions.[2]

A FOON is constructed of a few components. The nodes of this network are composed of objects and motions. The edges connecting the nodes are directed, bipartite and will always connect a motion to an object or an object to a motion. Throughout the network, several object nodes will be directed to a motion meaning the objects are interacting with the motion. Also, a single motion will be directed to several different object nodes, denoting the objects are the outcome of the motion.[3] Below is a visualization of this concept.

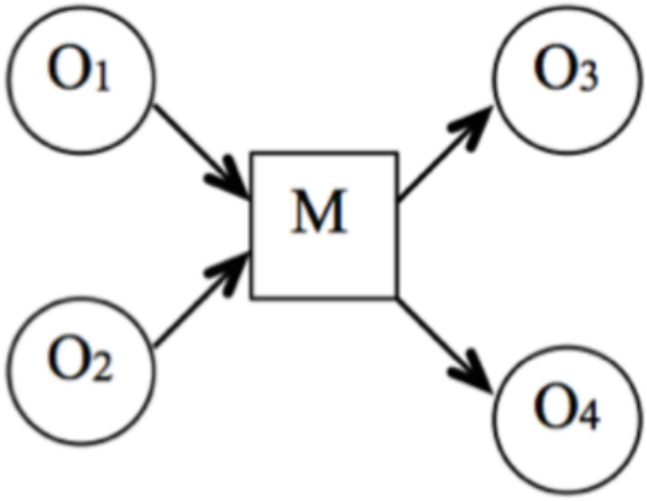

Fig. 1. Visualization of objects and motions connected by directed edges.

The data structure used for this given network is an adjacency matrix. Each node is represented by a row, and its relation to other nodes is given by the columns of the matrix. An edge is denoted by a 1 from node Ni to Nj, preserving directionality of edges. If two nodes are not connected, then the index at those two edges is a 0.

To represent all of these nodes in a reasonable fashion, a node list will be used with specific rules to follow. A node list is a text file that keeps track of all object and motion nodes found in a graph. The following rules are implemented when writing these graph text files: 1) The input objects are listed BEFORE the motion line and output objects AFTER the motion line. 2) Every object node is followed by a state node. 3) Everything in each line is tab separated.

The object line, state line, and motion line is formatted as follows:

```
O<ID>\t<NAME OF OBJECT>\t<0 or 1, describing if this object is moving or not>
```

Fig. 2. Object line.

```
S<ID>\t<OBJECT'S STATE>\t{LIST OF INGREDIENTS, COMMA SEPARATED}
```

Fig. 3. State line.

```
M<ID>\t<NAME OF MOTION>\t<STARTING TIME>\t<ENDING TIME>
```

Fig. 4. Motion line.

If robots can effectively learn tasks from a functional network like the one discussed, then these robots can be used for daily service tasks that we will no longer need to worry about such as cooking. With robots being integrated into day to day life, an effective way to provide information to these robots and to train them is necessary for their success. This is what a FOON is able to provide.

Although the scope of this project just covers the concept of cooking, robots have already infiltrated many markets of common life. The most recent being self-driving cars. In any field, we need to provide an efficient way of feeding these robots data. Without valid data to learn from, the robots do not know how to properly behave. Autonomous vehicles are a great example of this because with every mile driven, the cars are learning the roads and the driving habits of other vehicles.

II. Video Annotation

The first step in building a universal FOON is collecting the data from videos. When watching these videos, we are looking for the objects, states, and motion carried out in each step so we can know how to perform these tasks by just observing the data. [4] Throughout the project, I collected data for two different videos provided by the TA David Paulius. An example of one complete functional unit is shown in figure 5.

Course Name: Intro to Robotics. The work is supervised by Professor Yu Sun and TA David Paulius.

```
O13     pan     0
S193    ingredients heated inside       {broccoli}
O173    kettle  1
S193    ingredients heated inside       {water}
O46     water   1
S400    in kettle
M4      pour    1:13    1:26
O13     pan     0
S193    ingredients heated inside       {broccoli,water}
O46     water   0
S176    in pan
```

Fig. 5. Example of a complete functional unit.

III. Merging the annotation results into FOON

After the data was collected, we were provided the univeral FOON that contained upwards of 60 different annotated files that were already merged as the univeral FOON. With multiple subgraphs being added at any rate, we need a way to merge these newly created graphs into the existing universal network. In doing so, this allows us to have multiple representations of producing something without the need of overlapping nodes that are common in many different subgraphs.

The main idea behind the algorithm for merging is for each functional unit in the subgraph, check to see if it exists in the universal FOON. A functional unit exists within FOON if it has the same number of input and output nodes, equal object and state types, equal list of ingredients (regardless of order), and equal motion node type. If the functional unit does exist in FOON, we can move on to the next functional unit but if it does not exist in the universal FOON, we must add that functional unit to the network. During the process we want to make sure we are not adding any object nodes that already exist in the FOON as well. To test the merging algorithm before using it on the universal graph, I merged the two annotated files I had created in the first part of the project to see if it was functioning properly. Figure 6 is the visualization of the merged file.

Since the two data sets were small, I was able to check the result by hand and through the initial testing I was obtaining the correct results. One thing to mention about the algorithm I chose to implement is the fact that I did not consider the list of ingredients when merging the two files. This will inevitably produce different results and will generate less functional units in the universal graph. When you do not consider the list of ingredients, more objects are similar which means less objects are needed to be added to the universal graph. This is just something to consider when looking at the data and further discussion.

IV. Searching a network for a task tree, given a specific kitchen

The ability to create a universal FOON is quite useful because now we have all of the data in one location. With all the data in one location, a robot can use this data to solve specific problems that are defined in the graph. A problem can be solved by breaking down the steps needed to solve it. We can now look at the data and perform certain actions on it and

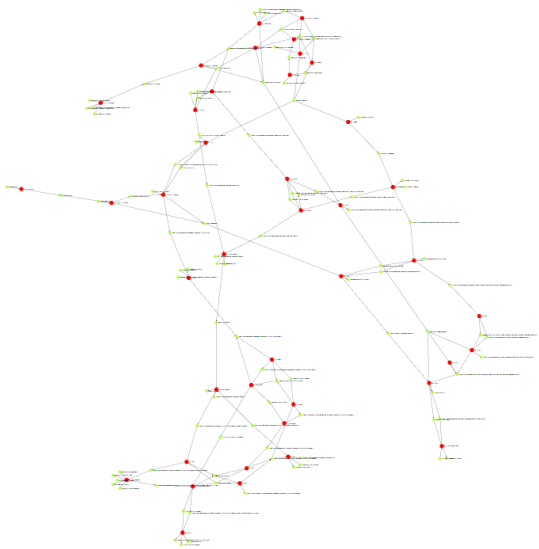

Fig. 6. Visualization of the two annotated files merged into one network.

the main action we are concerned with is the action of finding how to make a specific goal node. The process of finding the steps necessary to produce a certain goal node is known as task tree retrieval.

The main idea behind the task tree retrieval is as follows. We first define a goal node that must exist in the network. We also define what we have available in our ”kitchen” (this is a file of objects and states that we have readily available). We will create a queue that holds the items that we do not know how to make, which first contains just the goal node. For each node in the queue, we will search for the functional units that can create the node we are observing. When we look at each of these functional units, we need to make sure the input objects are defined in the kitchen file. If not, we do not have the necessary tools to make the goal node. While the goal node is not in the kitchen file, we keep on searching for the path to create the goal node. One thing to be careful of is the fact that the search can not progress forever so we do need some stopping condition of when it has performed enough iterations. In my testing, I found that 10000 iterations is sufficient to stop the search if we have not found a solution already. Figure 7 is a representation of searching the univeral FOON for a given goal node.

V. Discussion

The results of this project are fairly sufficient depending on what files you use. Through the initial testing of the merging of two files, small files worked as expected. As mentioned above, I was able to test the merging algorithm by hand to make sure the correct output was given. For the actual implementation of the merging algorithm, I believe it worked as expected. The only reason why the results may vary a bit is due to the fact that I did not consider the list of ingredients. For some files, this will not have too much of an impact on what the merged file looks like but on some, the results may seem unorganized and random. This is due to the fact that the same object can

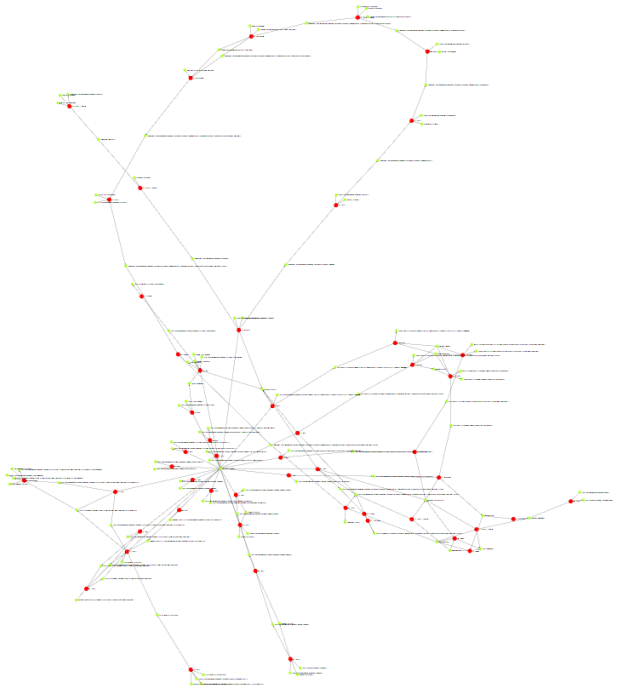

Fig. 7. A visualization of searching the universal FOON for a given goal node.

be tied to vastly different states that should be separate based on the ingredients list. Without the ingredients list, these very different states are seen as the same object so there may be a loss of useful data when combining larger files.

For the task tree retrieval, again initial testing of searching on small datasets worked as expected. Through the initial testing, I was using files I had created myself so I would know the expected outcome. When performing the same search on the files provided, there were several times where there was a read error that stopped the code from running. I believe this is due to the written format of the files given. If the files were written on a mac computer and tested on a windows, the end of line character is different and may have messed up the data parsing when reading in the file. I believe this is one of the main struggles of the algorithm. Since the FOON file was over 10000 lines long, there was no convenient way to rewrite the file on windows. The other complication of the task tree retrieval was checking to see if we have the necessary items in our kitchen. This was a complication in the algorithm and was only functioning properly for some instances of the search.

Although I did have the complications with the task tree retrieval, I was able to manage to get an output for some of the goal objects that were given to me. That being said, there are many useful implementations for a universal FOON. With such a network, robots can learn tasks to be performed. As the FOON continues to grow, robots can continue to learn and gain knowledge on how to perform the tasks seen in the graph.

REFERENCES

[1] D. Paulius, A. B. Jelodar, and Y. Sun, "Functional object-oriented network: Construction expansion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5935–5941.

[2] D. Paulius and Y. Sun, "A survey of knowledge representation and retrieval for learning in service robotics," *arXiv preprint arXiv:1807.02192*, 2018.

[3] Y. Sun and Y. Lin, "Modeling paired objects and their interaction," in *New Development in Robot Vision*. Springer, 2015, pp. 73–87.

[4] A. B. Jelodar, D. Paulius, and Y. Sun, "Long activity video understanding using functional object-oriented network," *arXiv preprint arXiv:1807.00983*, 2018.