IIT Documentation

Data Structures

Node Struct

Network Struct

`network.connect_mat`
: The NxN binary connectivity matrix

`network.options`
: An array of integers which represent the options (see Options section)

`network.nodes`
: A struct array of node structs (see Node Struct)

`network.num_nodes`
: The number of nodes, N, in the full system.

`network.tpm`
: A States X Node (2^N X N) transition probability matrix for the entire system. This matrix does not need to be State X State because of conditional independence.

`network.full_system`
: An array of the numbers 1:N.

`network.num_subsets`
: The number of subsets in the system, 2^N.

`network.current_state`
: An integer array of length N that is the current state of the system. This is only used when the software is set to only compute over one state as opposed to taking an average of all states.

`network.num_states`
: Total number of states of the system. This is equal to the product of the number of states of each node. The software is almost totally able to handle more than binary nodes, but there are still some changes that need to be made.

`network.noise`
: A number in the range [0,.5] that adds noise into the output of every node. That is, the deterministic output will be correct with probability 1 - noise.

Output

`output_data.results`
: This is the main struct which holds all of the information we want to know about the analysis of the network such as Big Phi values, small phi values, MIPs, concepts, etc, etc. The structure of this data follows.

```
results.state(state_index).subsystem(subsystem_index).
                    .Phi
                    .Phi_MIP
                    .MIP1
                    .MIP2
                    .concept(concept_index).
                          .phi.min
                          .phi.backwards
                          .phi.forwards
                          .distribution.backwards.whole
                          .distribution.backwards.MIP
                          .distribution.forwards.whole
                          .distribution.forwards.MIP
                          .denominator.backwards
                          .denominator.forwards
                          .MIP.forwards.numerator1
                          .MIP.forwards.numerator2
                          .MIP.forwards.denominator1
                          .MIP.forwards.denominator2
                          .MIP.backwards.numerator1
                          .MIP.backwards.numerator2
                          .MIP.backwards.denominator1
                          .MIP.backwards.denominator
                          .is_irreducible
```

What we see above is the structure of the results struct. First you index by state, then by subsystem. To get a state index you can use the function `state2index.m`, which takes as input a vector of length N where N is the number of nodes in the network and each entry is less than the number of states of that node and non-negative, as well as a vector which represents the number of states of each node. For now, that vector will always be a vector of all 2's of length N. The output of the function is an integer which is used to represent the state. There is also the reverse function `index2state.m` which works as expected - you input an integer between 1 and the number of states of a system along with the vector representing the number of states of each node. Next you choose the subsystem you are interested in, and again we have functions to turn a vector into an integer, and as expected it is called `subsystem2index.m` which takes a vector of integers that are a subset of the numbers 1 to N. Also, we have the inverse function as well: `index2subsystem.m.` These function don't need anything more than the vector or integer. You can also use `results.main_complex` which is an integer that corresponds to the index of the subsystem that is the main complex.

Each subsystem has it's own values for Phi (when we capitalize Phi we mean "Big Phi") and Phi_MIP. Each of these values is a non-negative real number. We also have MIP1 and MIP2 each of which represent on side of the MIP of that subsystem.

Each subsystem also has a struct array of concepts which are indexed by an integer. These indices represent the numerator of the concept, and once again we have a function for converting an

array of integers that are a subset of 1 to N to an index: `concept2index.m`, and also the inverse, `index2concept.m.` These functions take the vector or integer along with a vector that represents the subsystem (a vector of integers that is the subsystem, for example [1 3 4 6]). Each concept has a small phi value, the min, but we also include the individual forward and backwards phi values. There are also four distributions. Two for each direction, where each direction has the distribution over the full subsystem as well as over the subsystem cut according to the MIP for this concept. You can also access the denominator of the concept for both the forwards and backwards directions. The desciption of the MIP for the concept is split up into 8 pieces, first the MIPs for each of the forward and backward direction each of which is split up into numerator and denominator, which is finally split into two pieces which represents the MIP. Lastly, we have a logical value (0 or 1) named `is_irreducible` which tells us whether or not this concept is irreducible or not.