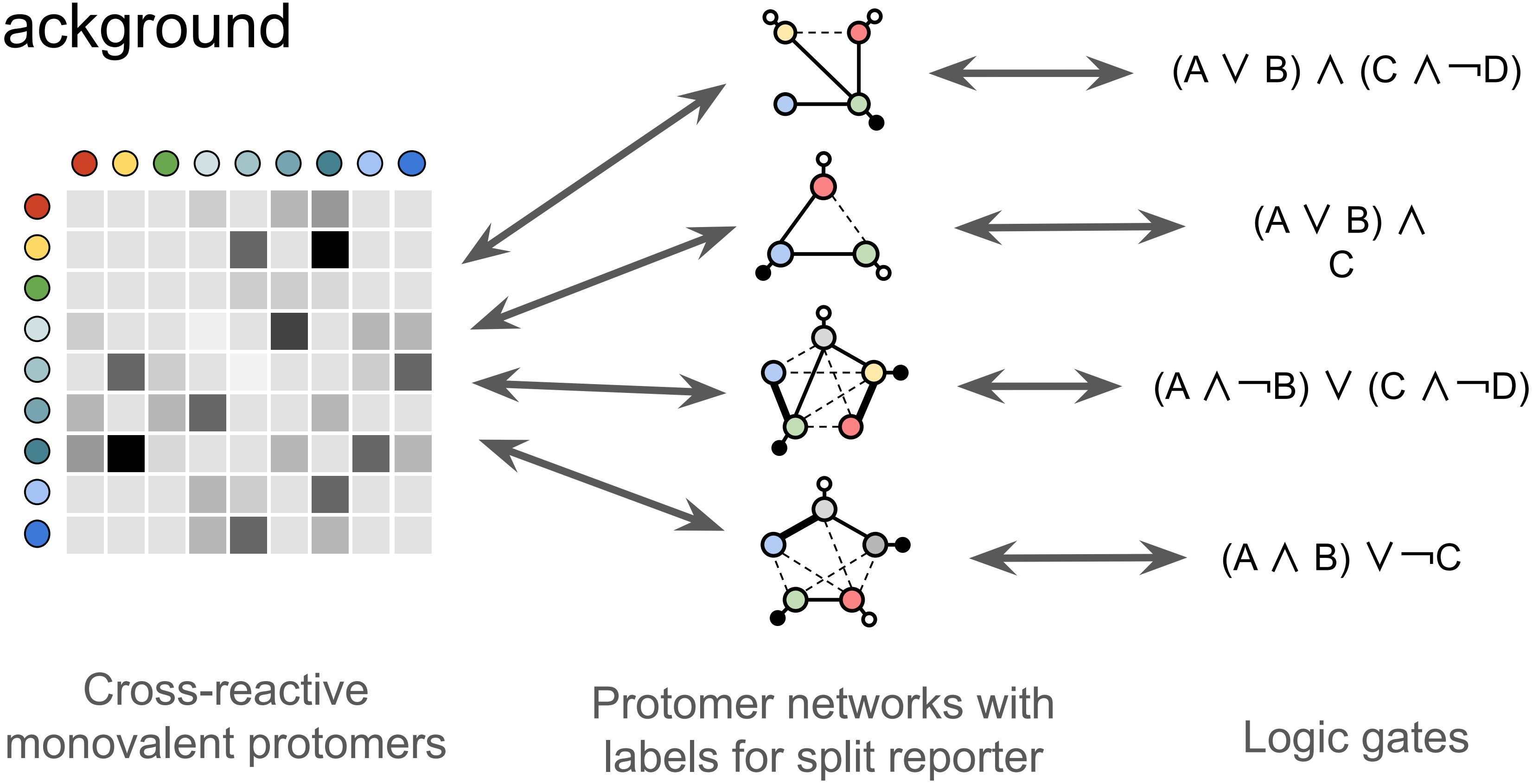
protomer-network-distributions

Isaac Lutz

Background

- Developed *de novo* dimeric proteins that interact at one interface
- Promiscuous interactions at shared interface (monovalent)
- Interactions are reversible and displace one another (competitive binding)
- Can combine sets of these protomers into networks, which will equilibrate to some distribution defined by their interaction affinities
- With a split reporter fused to two protomers, particular dimers will result in a signal upon binding
- Can create protomer network logic gates, where addition of different protomers as inputs will shift equilibrium distribution and change output signal

Background


(A ∨ B) ∧ (C ∧¬D)
(A ∨ B) ∧ C
(A ∧¬B) ∨ (C ∧¬D)
(A ∧ B) ∨¬C
Cross-reactive monovalent protomers
Protomer networks with labels for split reporter
Logic gates

Use Cases

- Interactions through a notebook
- Solve for an equilibrium distribution based on protomer identities, affinities, and concentrations in a protomer network
- Predict the binary output of this equilibrium distribution provided split reporter labels and a signal threshold
- Plot equilibrium distribution to visualize
- Turn off all outputs besides returned values to allow for incorporation of this tool into a larger systematic analysis of protomer networks

Demo

See https://github.com/idlutz/protomer-network-distributions/tree/main/examples

Design

- protomer_network class is initialized with all protomers that may be used and the affinities of all of their interactions, and is used for all functions
  - Provided as a python dictionary
  - Initializes with lists to store inputs and outputs of simulations
- SimulateEquilibriumSpeciesDistribution method runs a specific simulation
  - Provide subset of protomers, their labels, and optionally their concentrations as a dictionary input, with dictionaries of each property
  - Can optionally print model summary and plot time course of simulating to equilibrium
  - Returns a dictionary of every species (monomers and dimers) and equilibrium concentrations
  - Stores the inputs and outputs of the method within the class lists

Design

- PredictBinaryOutput method predicts the binary output of a simulation
    - Input index of simulation to use, default is most recent (-1)
    - Uses the lists of simulation inputs and outputs stored within the class
    - Returns a boolean for if the summed concentration of all signaling dimers is above a threshold
    - Can optionally return the raw signal value, in the form of a cumulative concentration
- VisualizeEquilibriumDistribution method plots a bar graph of the distribution
    - Input index of simulation to use, default is most recent (-1)
    - Uses the lists of simulation inputs and outputs stored within the class
    - Prints a bar graph of each species (monomers and dimers) and its concentration, colored by whether the species signals or not

Project Structure

See https://github.com/idlutz/protomer-network-distributions/tree/main/examples

Lessons Learned

- Clear project plan before coding
- Examples, documentation, unit tests
- Relatively unfamiliar with git/Github prior to this project
- Utility of separating finished work into python modules

Future Work

- Incorporate tool into systematic analysis of protomer network logic gates
- Distribute and use for research in designing and predicting protomer network logic gates