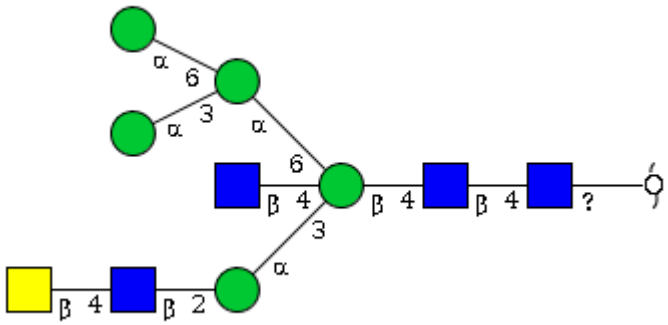
- **Prototype of interactive svg using javascript**
  - Some preliminary (prototyping) work has been done for the project using SVG to create interactive images of glycans.
  - A description of this project is shown after these bullet points. URLs for the prototype follow.
  - http://glycomics.ccrc.uga.edu/ggtest/files/svgGlycans-4.html?GOG123
    This is a "GET" request to retrieve html with javascript code that invokes a call to get the data of interest in json format - in this case information about the glycan with the id = "GOG123". Changing the argument of the request (e.g., changing "GOG123" to "GOG125") will change the data.
  - The json files containing the data used in the above example are in the following directory
    http://glycomics.ccrc.uga.edu/ggtest/files/json/
    These files are parsed and the svg representation is generated *on the fly* in this prototype.
  - However, **you will NOT use these json files and you will not generate svg representations on the fly**. You will use svg files that are being generated by the people at GlyTouCan. Here is a prototype example of javascript that manipulates one of these pre-generated svg files to make it interactive in new and different ways. (We are waiting for the full implementation of this protocol and the full set of svg files to be delivered.)
    http://glycomics.ccrc.uga.edu/ggtest/files/glycan-11.html
  - The web page encoded by url immediately above gets data from the following URL, which is pure SVG that already has some basic interactivity function. (Try it!) The html file immediately above overrides this basic functionality (by modifying the state of the <desc> tag) to do other things (like changing the image) when the objects in this svg file interact with the mouse
    http://glycomics.ccrc.uga.edu/ggtest/files/svg/glycan-11.svg
  - Remember, you are building a robust, generally usable javascript API that performs the kind of functions we described in our conversation. The code in the above URLs are just a prototype to show how it might work. You wll have to extend, modify, or completely rewrite this code to make your code usable in the broader context. For example, you will also have to write code to fetch a json response containing information about the individual nodes (indexed using a depth-first traversal of the structure tree) of the glycan being rendered. The http request and the information content of the json response is one of the things you will develop, along with more relevant actions to perform upon mouse-over, click, mouse out, etc. to represent the information in the json response. (Others will build the infrastructure to actually generate the json response with the appropriate information content.)
- **Advanced Graphical Query Interface**
  - Considerably less work at defining a prototype has been done for this project, which will utilize the d3 javascript library (https://d3js.org/).

- So, the aims of this project (outlined below) will have to be achieved in the order listed below so as to build up the required infrastructure.
- The first step (that you can do with minimal supervision) is to explore the d3 library and think about ways to parse object mapping information supplied in json format and convert that to an interactive image like the one shown in the project description below. Remember, we want code that is usable in a broad range of contexts, with specific mapping information for any context being provided as json.
- Subsequent steps will require considerable interaction with the other members of the team.
- You can explore all sorts of ways to describe collections of objects that are fectched and displayed using this approach. For example, you may want to show the populations of the elements of a hierarchical collection of objects as a set of concentric arcs, like this prototype example in svg http://glycomics.ccrc.uga.edu/ggtest/files/popArc3.html

Interactive SVG using Javascript

Introduction

Glycan molecules are commonly represented as a tree/graph using colored shapes as the node.



Each shape (circle, square, star etc.) represents a monosaccharide of the glycan. The edges of the graph represent the connections between the monosaccharides. One of our collaborators Prof. Aoki-Kinoshita is working on generating SVG representations of each glycans. We want to integrate these SVG in our webpage but allow interactive operations on the graphics.

Aim 1 – Interaction between SVN and webpage

Develop a JS library that allows interactive manipulation of the SVG and a webpage. Interactions between the SVG and a webpage, in which the SVG is embedded, are (in no particular order):

Task 1: Mouse over on elements in the SVG (e.g. nodes) results in tooltips showing additional information that is not embedded in the SVG but is retrieved by JS calls.

Task 2: Click events on the SVG (e.g. nodes) trigger change in the webpage (e.g. div).

Task 3: Events in the webpage trigger changes in the SVG (e.g. highlighting of certain notes and edges).
In this first aim we will develop the necessary basics that allow to interact between a glycan SVG and a webpage. This may require changes in the SVG generation to allow us to trigger the necessary functions. To this end we will collaborate with Prof. Aoki-Kinoshita to incorporate the necessary changes.

Aim 2 – Embedding in GlyGen

In this aim we will cooperate with partners in GWU and GT. They will provide the necessary data for the interactions in form of web services to our webpage.
Task 1: When mouse over a shape in the SVG a tooltip is shown with:

- Name of the monosaccharide
- Possible aliases of the monosaccharide
- Chemical image of the monosaccharide
- Mass
- Composition
- Link to PubChem

Task 2: When clicking on a shape in the SVG a in the webpage is updated with information of the monosaccharide:

- Name of the monosaccharide
- Mass
- Composition
- 3D image
- Chemical image
- Link to PubChem

Task 3: The webpage contains a list of motifs present in the glycan. When clicking on a motif the corresponding monosaccharides in the SVG are highlighted.

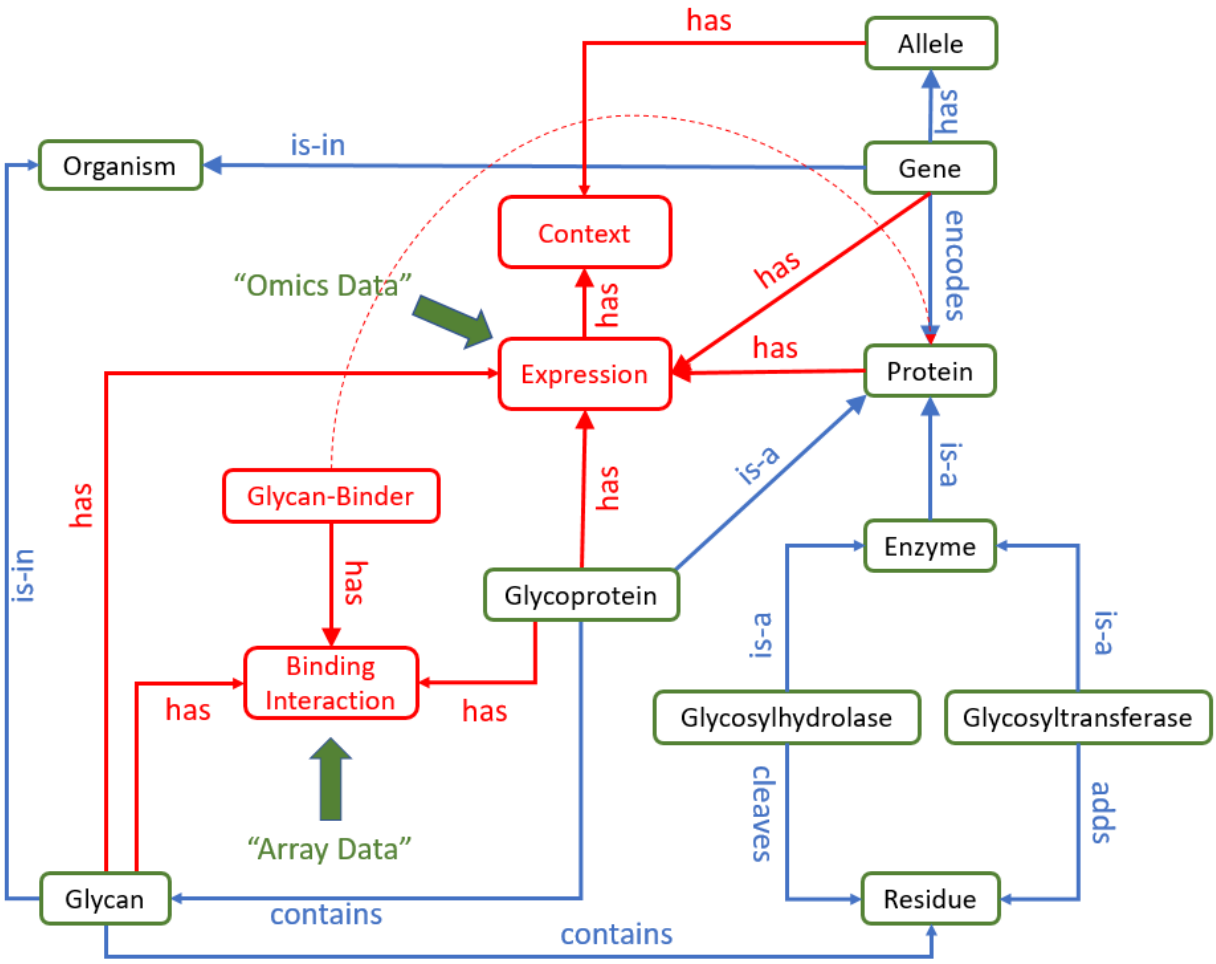
Material

- https://www.w3schools.com/graphics/svg_intro.asp
- https://css-tricks.com/using-svg/
- https://svgjs.com/docs/2.7/

Advanced graphical query interface

Introduction

As of now the query options in GlyGen (glygen.org) are limited to querying one type of molecule at the time by filling web forms. The limitation of this concept is that it is only possible to filter/query a type of objects/molecules by its properties. But it is not possible to query the entire dataset by properties of different molecule types. Although it would be possible to develop such a form based query interface it would be overwhelming and unusable. We want to create a graphical representation of the molecules/objects in our data model and their relationships and use this as the input interface for complex queries.



Aim 1 – Graph generation from JSON

In this first aim we will define a JSON format that contains the graph information needed to render a graph on a webpage using D3.js. At this stage D3 is a recommendation and alternative technologies (e.g. SVG) can be explored as needed.

Material:

- https://d3js.org/
- https://svgjs.com/svg.draw.js/demo/index.html

Aim 2 – Interactive queries

When clicking on one of the nodes or edges in the graph the user is represented with a query interface to filter based on the properties of this node or edge. This query interface can either be a dialog (JQuery dialog) or an adjacent in the webpage. Information should not be hardcoded but information about the query option should be retrieved from the JSON. This will also require using a query language to submit these queries to a web service. Since a user can sequentially select multiple nodes and edges in the graph and add options to them, this can be a complex query.

Aim 3 – Interactive update

For each type of object (green rectangle) a number is added that shows the number of objects of this type in our data model. When a new query is added to the graph (see Aim 2) these numbers get updated with the new query.

Aim 4 – Link out

A second type of click (e.g. double click, context menu option) will redirect the user to one of our list pages presenting the list of these objects. For this option it is also necessary to pass the query.