# Supplement B

The algorithms to generate glycan structures from a finite state automaton and generate the virtual glycome are described below.

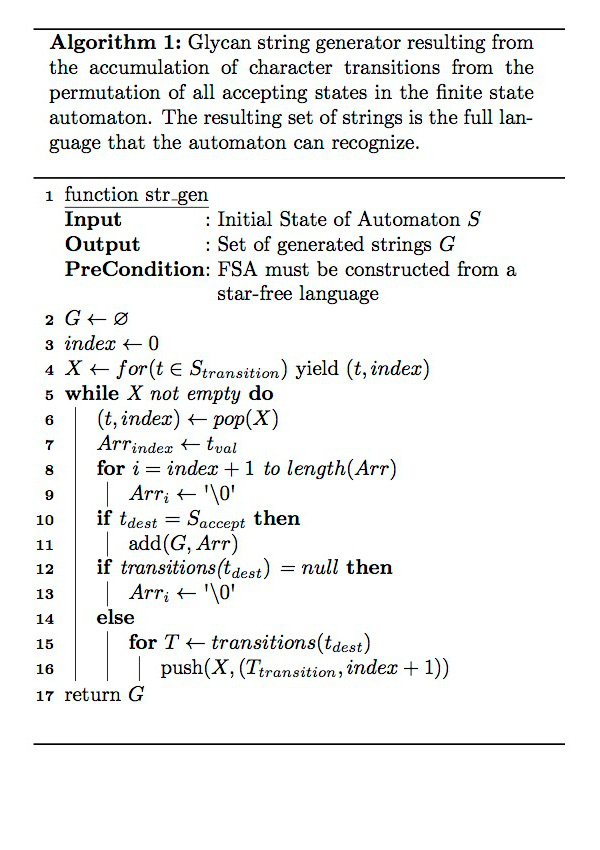


Figure . The algorithm for the str\_gen function has a pre-condition that the regular expression has the same expressiveness as a star-free regular language. Once the regular expression has been converted to an FSA, we simulate the Automaton through every possible input pattern that could reach an Accept state in the FSA using a non-recursive backtracking approach and add the generated symbols from the character array, *Arr*, to an output set, *G*. The *Arr* object contains the value associated with a transition at the location of the index variable. The index variable is also used to provide a mechanism to support backtracking of the transitions to incrementally add results to the output set, and restart the next result from the transition index on the transition stack, *X*.

Our Virtual Glycome Generator algorithm has the following inputs (we describe our inputs in parenthesis):

* The root glycan structure (lactose)
* The patterns that represent extensions and terminals structures (as defined in Supplement Figure 1)
* An extension token (Galb1-4 and Galb1-4 with a closing branch parenthesis)
* A glycan structure core size (we used a core size of 12 for the number of Hexoses and HexNAcs in the structure).

The Virtual Glycome Generator algorithm generates glycan structures by appending glycan extensions and glycan terminal modifications to the root glycan structure. We use the *str\_gen* algorithm to generate all the glycans from the regular expressions defined by the extension and terminal modification rules. We take the cross product of appending lactose and the terminal modifications and add them to the output set. We also add a resulting structure to an extension stack for further processing if the resulting string contains an extension token. The strings on the extension stack are replaced with the extension token concatenated with a terminal modification, repeated for each terminal modification. If there are multiple extension tokens in a string, then we perform the permutation of all possible replacements of the extension token. All resulting structures from each step of the extension and terminal modification process are added to the output set. The process of extending a structure or terminating a structure is repeated as long as the core size of the structure does not exceed the glycan structure core size.