DNA Assembly Grammar

Andrew James and Tanner Knabe

Project Summary (5-6 lines)

The purpose of ProjName is to simplify the DNA assembly process for biologists. There are myriad techniques for DNA assembly, each with its own capabilities, restrictions and reagents. With the grammar, a biologist can enter their DNA parts as well as any conditions they may have for assembly, and protocols will be given to them that best fit their requirements. The grammar is capable of handling both ordered and unordered joins as well as plasmid specification.

Major Software Components (1-3 lines each)

Syntax

```
<in> ::= [unordered_join] in v
<unordered_join> ::= ordered_join + unordered_join | ordered_join
<ordered_join> ::= seq * ordered_join | seq
<seq> ::= dnasequence | (unordered_join)
<v> ::= plasmidvector
```

Parsing

The tool takes an expression made in the grammar and breaks it into a set of expressions and subexpressions using placeholder variables to replace an expression in parenthesis.

Assembly Method Choice (Planned but not fully implemented)

The tool looks at the DNA part inputs to determine their compatibility with different assembly methods. Variables include length of DNA parts and number of DNA parts.

Graph Construction

To construct the Protocol Graph, we recursively resolve placeholder variables from the top-most expression and create nodes that correspond to each iteration of a potentially combinatorial design and draw edges from each subcomponent to each design they are contained in. We do this recursively until we have resolved every placeholder variable to its original parts. Due to the complexity of graph construction, designs are limited to series of pairwise combinations. However since all protocols support combining 2 or more DNA parts at a time, it would be possible to contract multiple nodes in the graph to require fewer total assemblies.

Protocol Compilation

The graph is traversed and each node beyond the original parts is given a protocol that includes the two parts that comprise the new part as well as the appropriate steps and reagents necessary to carry out the assembly. While full protocols can currently be generated for simple

expressions with no grouping (no parentheses), only partial protocols may be generated for some expressions with grouping.

Special compiling instructions(if necessary):

Requires Python3 and the packages numpy,itertools,networkx,copy,and pydot to be installed.

Requires environment that is able to execute IPython notebooks (like Jupyter Notebook)

If you would like to visualize your graph, you need to install graphviz and add it to your path (instructions on the graphviz website) and type the following command in CMD or Terminal: “dot -Tpng <yourgraphfile> > <yourgraphimage>”. The image will be available in the same directory you ran the command from. Or if you don’t need to save the image you can open your graph.dot file as a text file and copy and paste into webgraphviz.com and click “Generate Graph”.