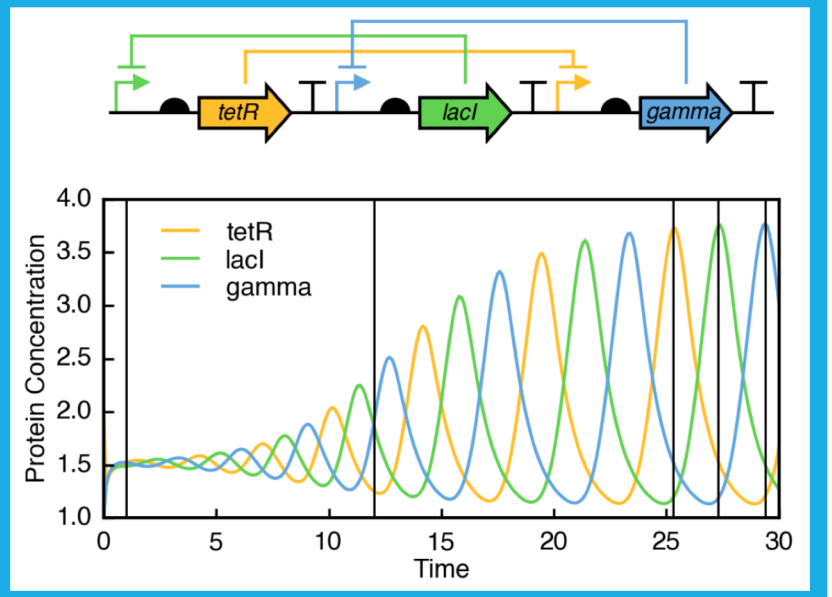
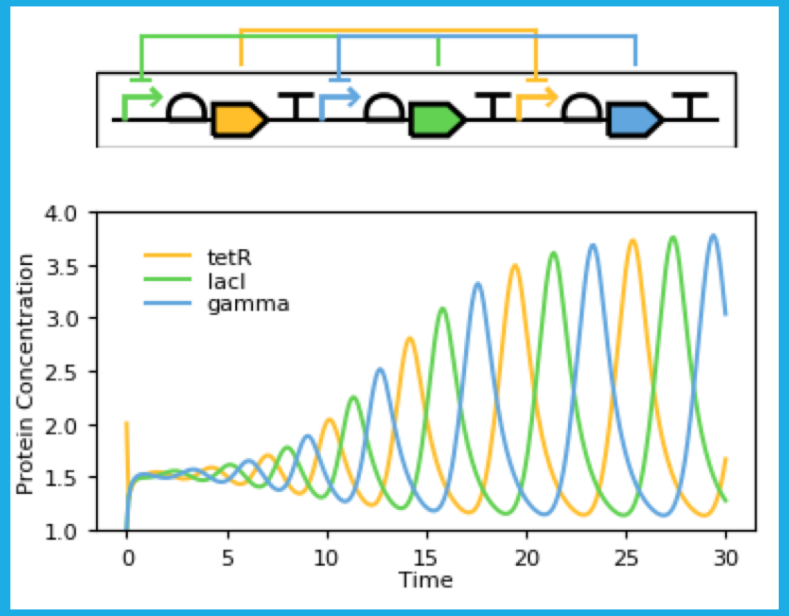




[fig 1]

Version 1

Version 2

RENDERING COMPLEX GENETIC DESIGNS WITH DNAPLOTLIB & PYSBOL

SUNWOO KANG
swkang73@stanford.edu
Department of Bioengineering, School of Engineering, Stanford University

OUTLINE

- Overview of DNAplotlib
  - Introduction of python rendering tool DNAplotlib
  - Focus of the updates
- Integrating pySBOL
  - Mapping sbol datatype into DNAplotlib datatype
  - Standard parametric svg files
  - Automated layout & annotations
- Demo
  - Example scripts
  - Instant design & rendering
- Conclusion

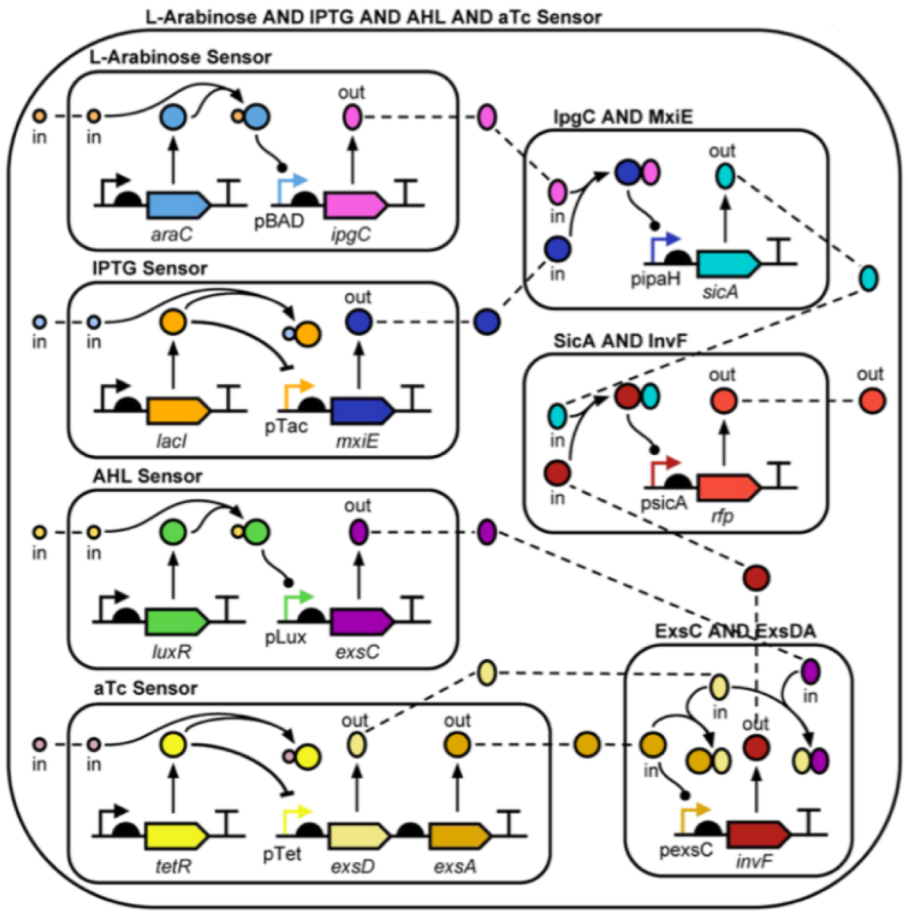


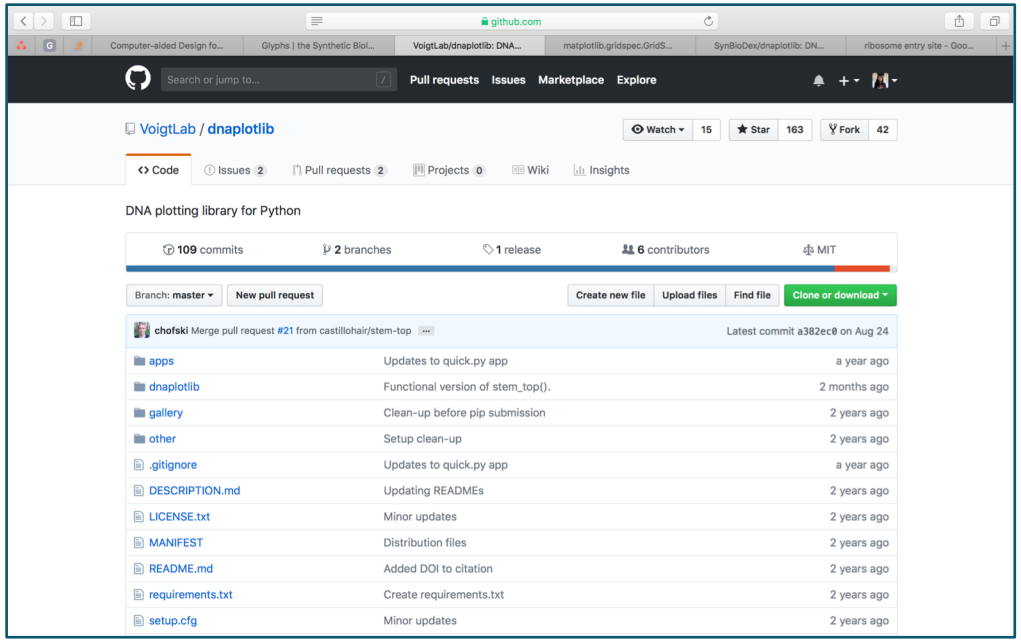
Image from *Roehner et. al,* 2016 (DOI: 10.1021/acssynbio.5b00215)

DNAPLOTLIB

INTRODUCTION OF DNAPLOTLIB

- Python rendering tool originally developed by Voigt Lab in Massachusetts Institute of Technology
- SBOL visualization onto python graph rendering tool **matplotlib**
- Advantage:
  - publication quality figures
  - complement circuit design with matplotlib graphs
  - simple, quick rendering toolkit for biocircuit visualization
  - Input: sbol files (rdf/xml)

    Output: jpg, pdf, xml, csv, txt

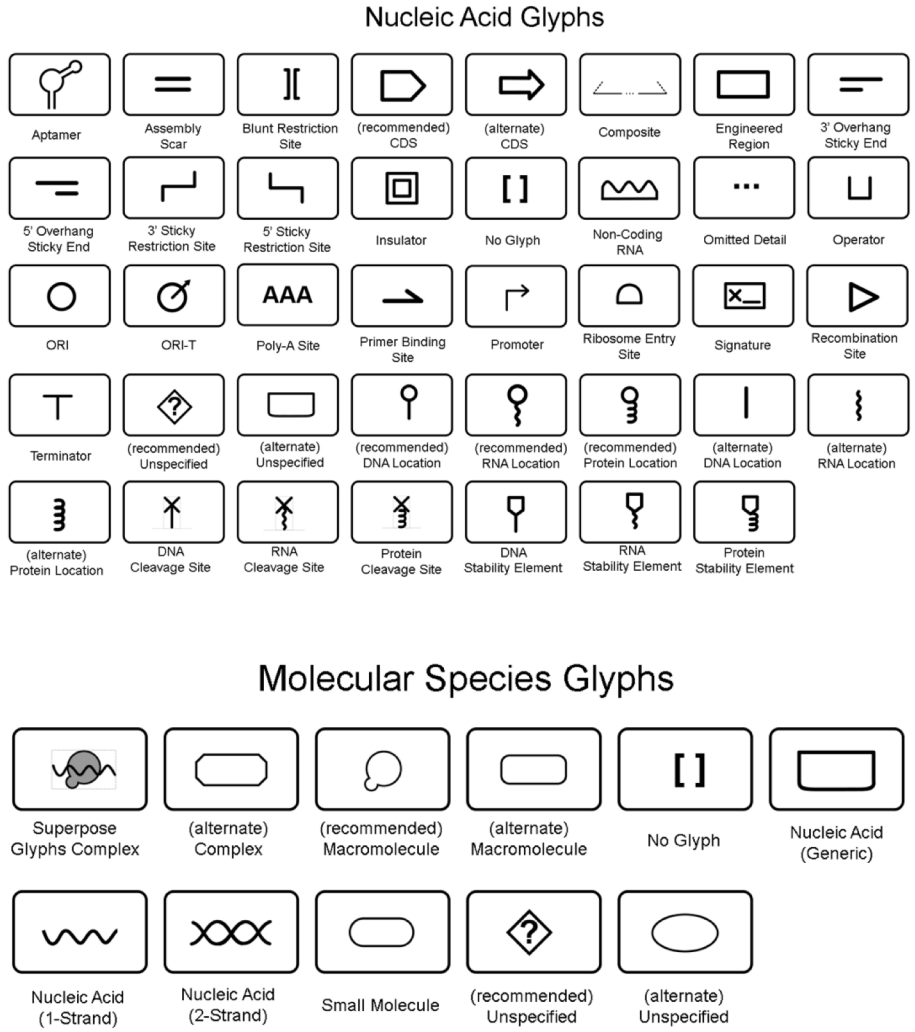


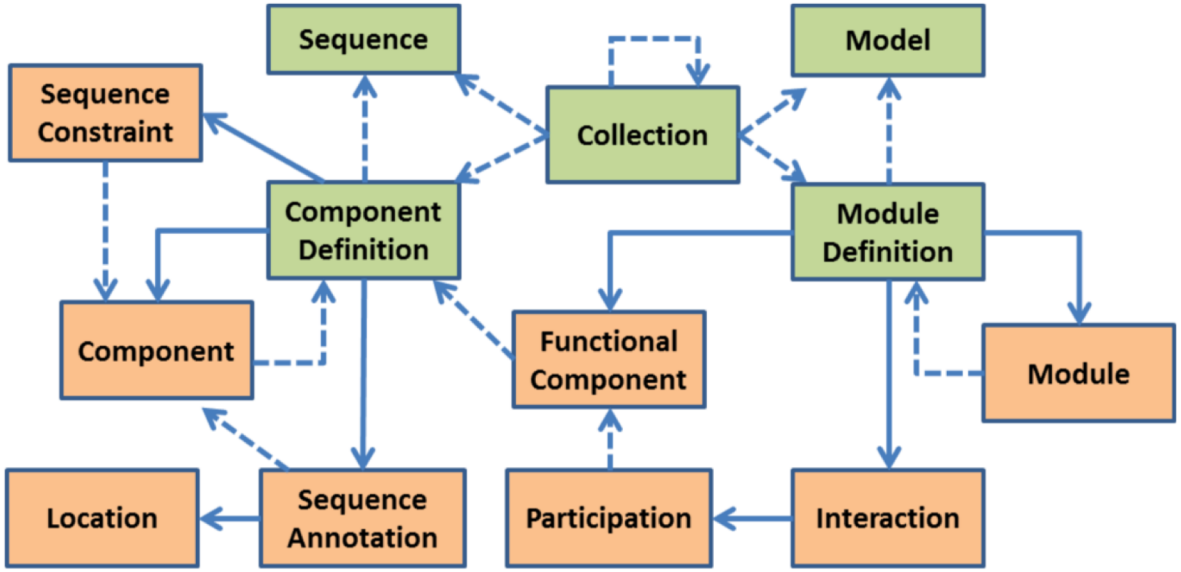
https://github.com/SynBioDex/dnaplotlib

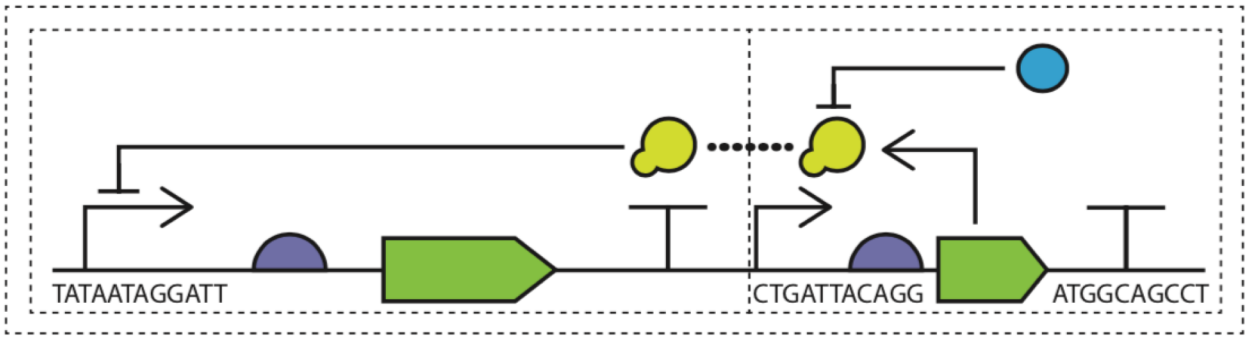
FOCUS OF THE UPDATES

- Compatibility with SBOL Visual version 2.2
- Advancements:
  1. Simple rendering of non-nucleic acid components
  2. Advanced visualization of five different interactions
  3. Recursive rendering of hierarchical module
  4. Automated export / import of design into xml

SBOL visual 2.2







Both figure from Synthetic Biology Open Language (SBOL) Version 2.2.0

INTEGRATING PYSBOL

MAPPING DATATYPE

- DNAplotlib: 5 new datatype classes

  : Part, PartList, Interaction, Module, Design

DNAplotlib	SBOL
Part	Components, ComponentDefinitions, FunctionalComponents, Participants (Interaction)
PartList	ComponentDefinitions
Interaction	Interaction
Module	Module, ModuleDefinition
Design	Document

MAPPING DATATYPE

- Sample test design from script datatype.py

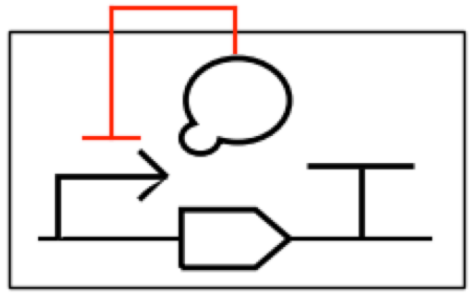
```
def create_test_design7():
    # create design
    design = Design('design7')

    # create modules & parts
    module1 = Module(design, 'module1')
    part_1_p = Part(module1, 'promoter1', 'Promoter')
    part_1_c = Part(module1, 'fnr_gene', 'CDS')
    part_1_t = Part(module1, 'terminator', 'Terminator')
    other_part_p = Part(module1, 'fnr', 'Macromolecule')

    module1.add_part([part_1_p, part_1_c, part_1_t])
    module1.add_other_part(other_part_p) # non-dna onto other parts!
    design.add_module(module1)

    # create interaction
    design.add_interaction(Interaction('inhibition', other_part_p, part_1_p))

    return design
```



Simple autoregulation circuit

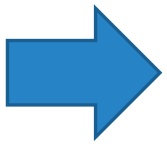
STANDARD PARAMETRIC SVG FILES

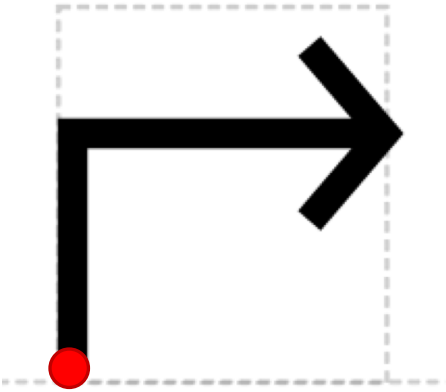
promoter.svg UNREGISTERED

promoter.svg

```
1  <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="680"
   height="190"  xmlns:parametric="https://parametric-svg.github.io/v0.2"
   glyphtype="Promoter" soterms="SO:0000167" parametric:defaults="
   totalWidth:0; height:45; box_size:33; offset:7.5; rectSize:30;
   othersize:16">
2
3  <rect class="bounding-box" id="bounding-box" x="4.854" y="2.862" width=
   "26.367" height="30.142" style="fill:none;stroke:rgb(153,153,153);strok
   e-opacity:0.5;stroke-width:0.4px;stroke-linecap:butt;stroke-linejoin:mi
   ter;stroke-dasharray:1,0.5;"/>
4
5  <path class="baseline" id="baseline" d="M-0.12,32.944C35.955,32.944
   36.015,32.944 36.015,32.944"  parametric:y="32.944" style="fill:none;st
   roke:rgb(153,153,153);stroke-opacity:0.5;stroke-width:0.4px;stroke-dash
   array:1,0.5;"/>
6
7  <path class="unfilled-path" id="promoter-body" d="M25,6 L31,13 L25,20"
   parametric:d="M25,6 L31,13 L25,20" style="
   fill:none;stroke:black;stroke-width:2.4px;"/>
8
9  <path class="unfilled-path" id="promoter-head" d="M6,31.797 L6,13
   L30,13" parametric:d="M6,31 L6,13 L31,13 z" style="
   fill:none;stroke:black;stroke-width:2.4px;"/>
10
11 </svg>
12
```

No duplicates

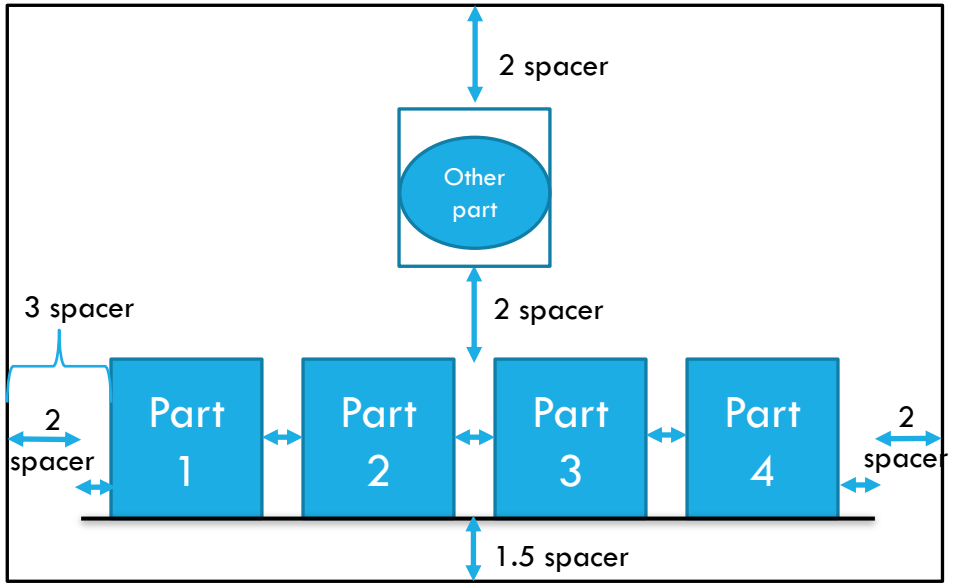




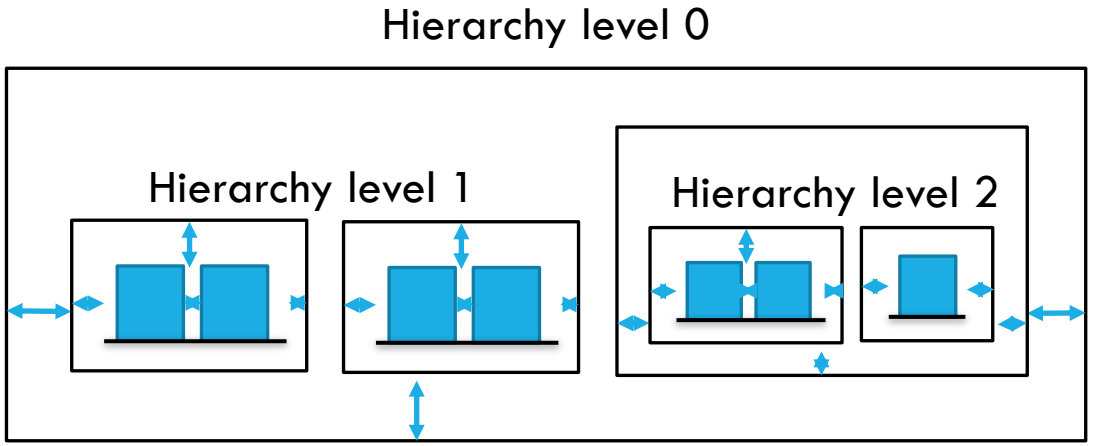
Promoter
Saved as Part

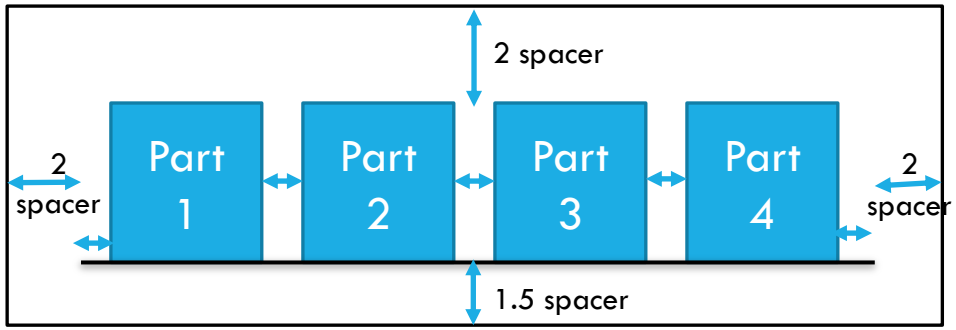
AUTOMATED LAYOUT & ANNOTATIONS

- Default module rendering layout
- Support recursive modules (up to 3 levels)



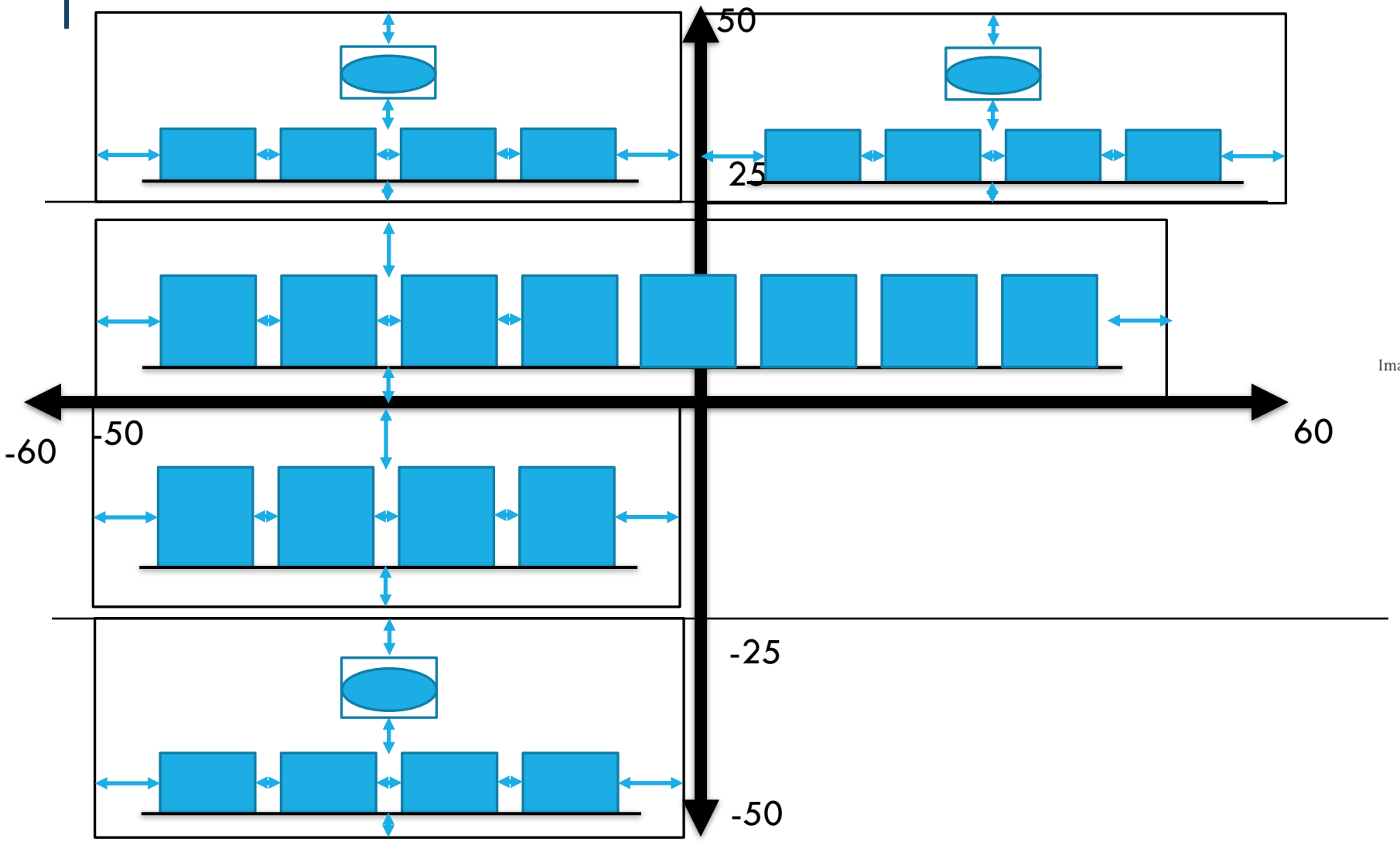
With non-nucleic part





Only nucleic part

AUTOMATED LAYOUT & ANNOTATIONS



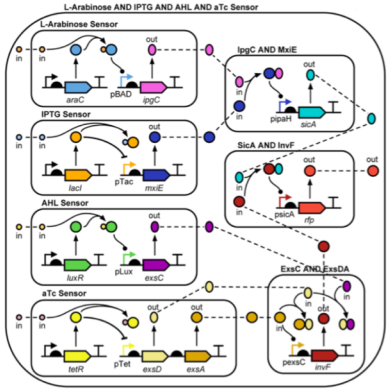


Image from *Roehner et. al,* 2016 (DOI: 10.1021/acssynbio.5b00215)

Order:

1 5

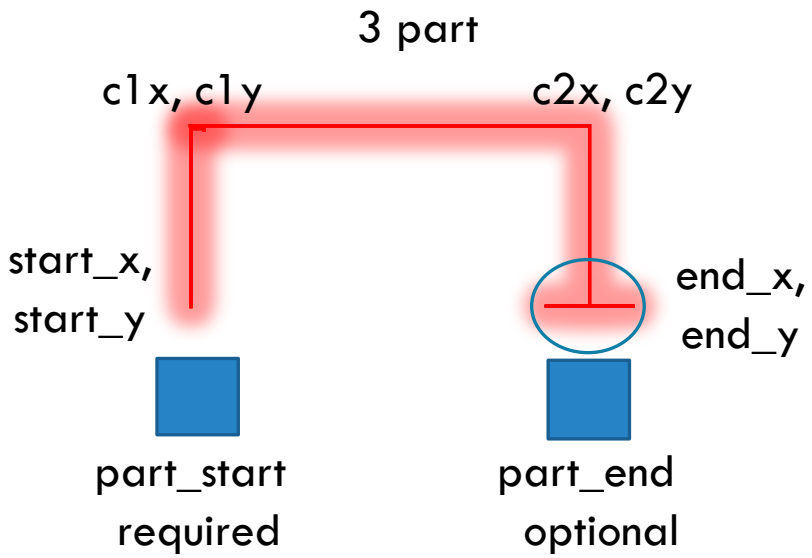
2 6

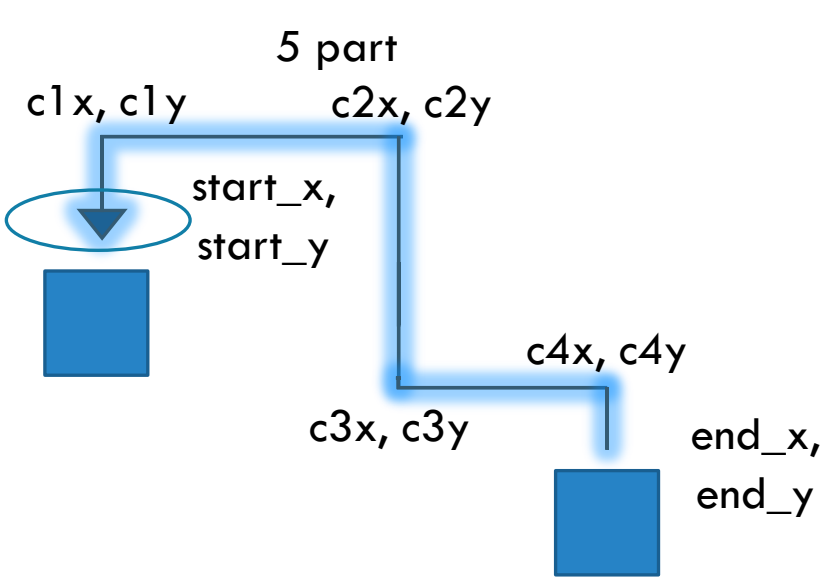
3 7

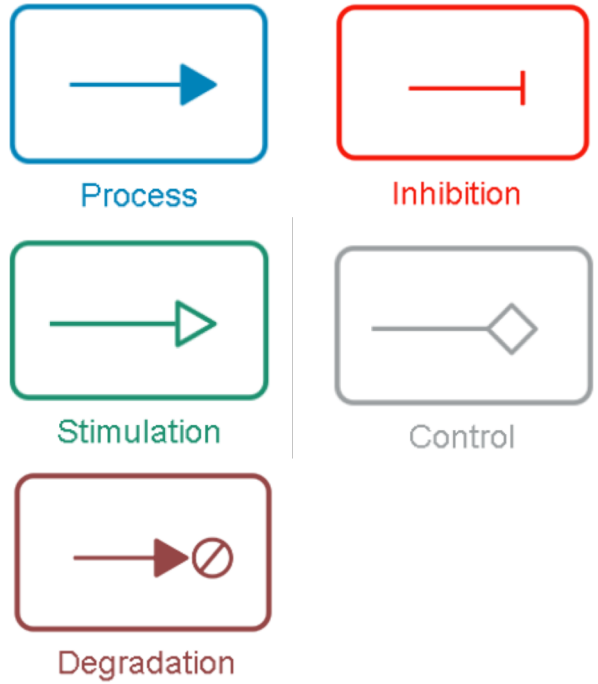
4 8

AUTOMATED LAYOUT & ANNOTATIONS

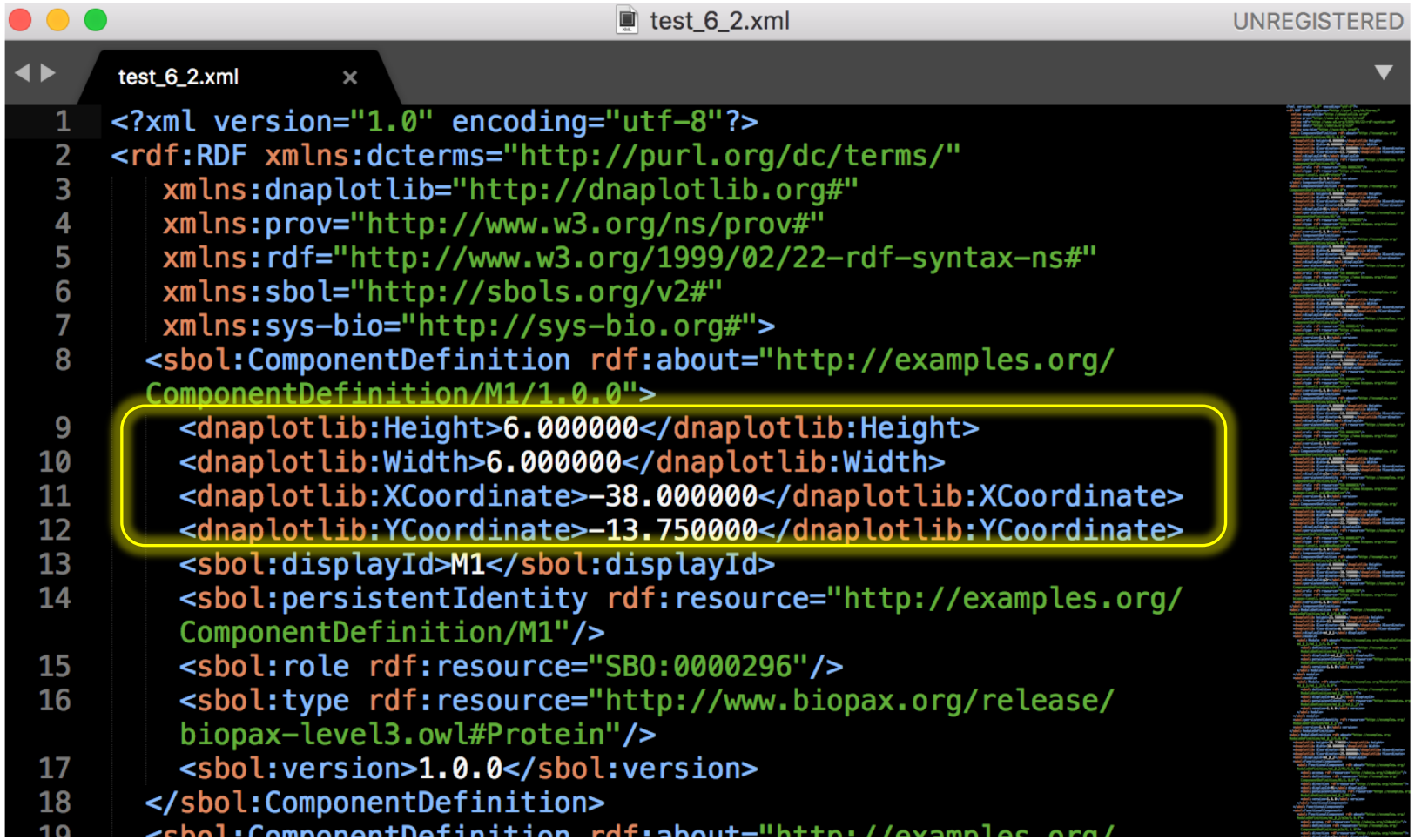
- Default rendering of interactions
- Random variation of y coordinate

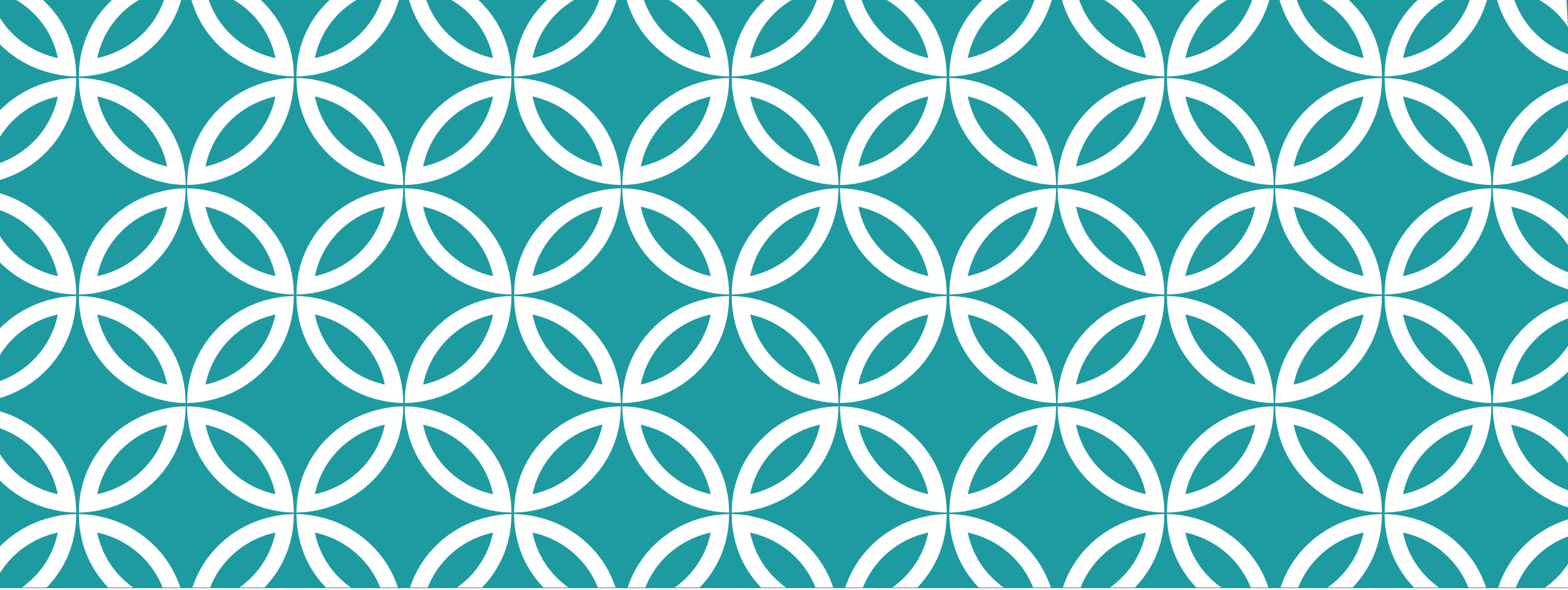






AUTOMATED LAYOUT & ANNOTATIONS





DEMONSTRATIONS

More at:

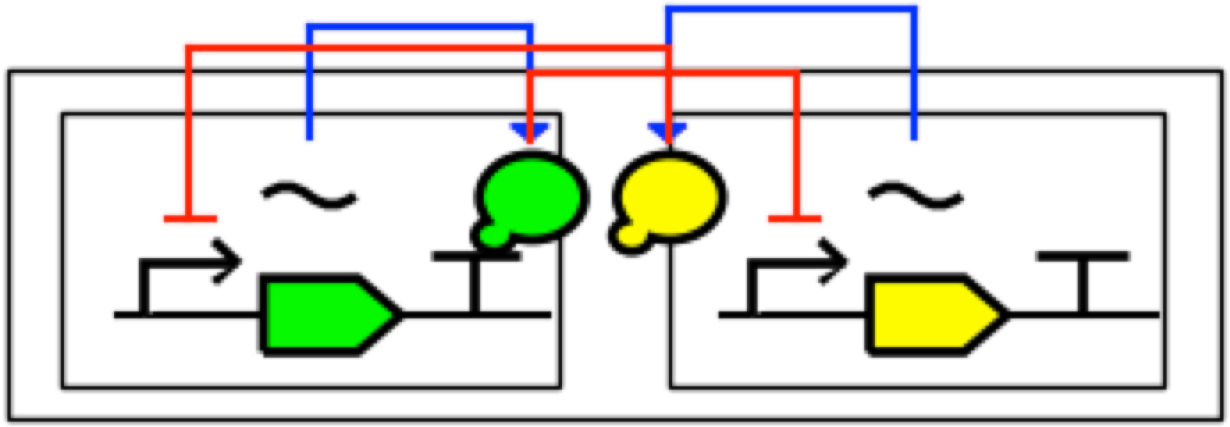
https://github.com/SynBioDex/dnaplotlib

EXAMPLE SCRIPTS

- https://github.com/SynBioDex/dnaplotlib/tree/master/dnaplotlib/examples

Script name	design
rotate.py	Three promoters rotated 0 / 90 / 180 degrees
backbone.py	Draw a promoter and a terminator on strand
oneModule.py	Draw a promoter, ORI, and terminator on strand with module frame box
eachModule.py	Draw promoter, ORI, and terminator on different strand With different module frame box
demo.py	'dynamics and evolution' rendered in DNAplotlib v2
testDesign.py	Basic script for rendering test design cases from datatype.py

INSTANT RENDERING



Toggle switch demo

CONCLUSION

- New tool for genetic circuit visualization available
  - Automated layout visualization
    + can be saved as annotations
  - Import / export with layout annotation
  - Development package available
- To Dos
  - Test import / export files
  - Need standardized svg glyph files
    (ex: alternate CDS, blunt site, etc)
  - Packaging via pip

Bryan Bartley
BBN Technologies, Cambridge,
Massachusetts, USA

Thomas Gorochowski
School of Biological Sciences,
University of Bristol, Bristol, UK

Google
Summer of Code

NSF

ACKNOWLEDGEMENT