DyNAMiC Workbench Documentation

Casey Grun

December 7, 2014

Contents

Chapter 1

Getting Started

This document will get you started with Workbench as quickly as possible. For a more detailed introduction to what Workbench is and what it can do, see the Overview.

Note: This part assumes you've already installed Workbench Server, or that you're using a hosted installation. If you didn't configure this server, you're on a hosted installation; go on to the next section. See the server page if you're confused.

1.1 Designing a Nodal System

To design a system using the Nodal Abstraction described in Yin et al., 2008:

1. Create a new file
   - Select "New" from the main menu bar
   - Enter a filename, like "system.nodal" (you can omit the file extension)
   - From the menu under "Create," select "Nodal System." Your new file should appear in the tree on the left (labeled "Files")
   - Open the file by double-clicking on its name in the Files tres
2. Add some nodes to the workspace
   - Drag and drop motifs from the palatte labelled "Motifs" onto the main white area (the "Workspace"). Nodes can be renamed by clicking on the gray label above.
   - For a full reference of the different motifs, see nodal systems.
3. Connect nodes together

- To indicate complementarities, select the Complementarity tool (labeled “Connect”). Click and drag from output ports (circular) to input ports (triangular).
- You can delete complementarities by switching back to the pointer tool (arrow), selecting a link, and pressing the large red “X”

4. Build your project and send output to a sequence designer
    - Select the “Build” tab, then click “Compile”. A new tab should open showing a secondary structure representation of the compiled system

Read more about the nodal designer.

1.2 Secondary structure design with DIL

You should see a window with several panes, showing a secondary structure representation of your system. This is the DyNAMiC Intermediate Language (DIL) editor. In the middle, there is a grid showing the secondary structures of each complex in the system. Double-click a complex to edit its composition or structure. On the right, there is a list of each unique *segment* in the system; you can use this pane to impose sequence constraints on the system to be designed (these will be passed to the sequence designer). On the bottom, there is a list of each *strand* in the system; you can edit the composition of strands or view their sequences.

From the toolbar atop this window, you can export the system to one of several sequence designers (DD, NUPACK, Multisubjective), perform a reaction enumeration, or analyze system thermodynamics.

Read more about the DIL editor

1.3 Reaction enumeration

To enumerate possible reactions and get an idea about the possible reactions your system may undergo:

1. Click the “Enumerate” button inside the toolbar atop this pane
2. A new window should open showing a text-based representation of your system; within this new window, click “Run Enumerator”
3. One more window should open, showing the enumerated reactions between your initial complexes. Boxes represent intermediate complexes, connected by grey links to reactions (which are shown as circles). Drag the white background to pan, and use the mouse wheel to zoom. Click complex boxes to view their secondary structures.

Read more about the reaction enumerator

1.4 Sequence design with Web DD

To design some sequences for the system you described earlier:

1. Return to the secondary structure (DIL) tab
2. Click the "DD" button in the toolbar at the top of this window
3. Click "Mutate" to begin optimizting the design. You'll see the individual domains being designed in the center pane, a schematic of the various construct strands in the "Structure" pane to the right, and a real-time visualization of the candidate sequences in the "Strands" pane below. You can pause the mutation and save any part of this view by clicking the "Save" button and selecting a file name.
   - Mutate the design for a while
   - Once you're satisfied with the score (lower is better), pause mutations by clicking the "Mutate" button again.
   - Save your final strands by clicking "Save" in the "Strands" pane, and selecting a file (how about "system.seq").

Read more about Web DD, or Sequence design.

1.5 Thermodynamic analysis and simulation with NUPACK

To perform full physical model calculations using NUPACK on the sequences we've just designed:

1. In the "Strands" pane within Web DD, click "Compute"
2. Choose "Calculate Partition Function" to open a new window which will allow you to perform a thermodynamic analysis on the strands in the system
3. Click "Analyze" to begin a new analysis on the NUPACK web server.

Read more about using NUPACK in Workbench, or check out the NUPACK Website at Caltech

1.6 Next steps

- Read more about the Applications available in Workbench
- Learn more about the File system
- Discover how to customize Workbench by scripting and developing applications.

Chapter 2

Overview

DyNAMiC Workbench is an *Integrated Development Environment* for dynamic DNA systems. This document aims to survey the goals and architecture of Workbench and to acquaint you with important features. For a more rapid introduction, see the Getting Started guide.

2.1 Design Goals

Workbench has been designed with several goals in mind:

- **Integration** - Workbench allows access to the full set of tools necessary for designing assembly and computation systems based on DNA strand displacement, and it streamlines the workflow between these tools.
- **Usability** - Workbench presents an intuitive, cross-browser/cross-platform, web-based interface
- **Scalability** - Workbench has built-in task deployment system (see *Architecture* below) which can execute computationally-intensive tasks on a variety of targets, including the local web server, a cluster such as Orchestra, or a number of remote web services, such as the NUPACK or Mfold web servers.
- **Extensibility** - Workbench includes powerful, but simple tools for automating various tasks. The client-side interface is entirely scriptable, and tools are included for developing entirely new client-side applications. New server tools can be easily added to Workbench as well. Since the client and server are written in the same language (Javascript), code can be easily re-used between the client and server.

2.2 Architecture

Workbench includes two main components:

- **Workbench Server** – The server is responsible for storing and managing files, as well as deploying and execting computationally intensive tasks. If you're accessing a hosted version of Workbench, you likely won't interact directly with the Workbench server at all. If you're hosting Workbench yourself, it's helpful to understand a bit about the server. Read more.
  - **Server tools** - The server can run several types of computational tasks, each of which is encapsulated by a *server tool*. Server tools require a wrapper written in Javascript, but they can easily access and call tools written in any language. Most tools included with Workbench are written in C, while others are written in Python. Read more
  - **File management** - The server maintains a "home directory" for each user. Many types of files can be opened and edited within Workbench, including DNA-related files (.seq, .nupack, .nodal), and other relevant files not strictly related to DNA (.txt, .xml, .html, .tex), while other file types (.svg, .pdf) can be previewed from within Workbench. Read More
  - **Users** - To access the server, you need a user account. This is true even if you're hosting your own server instance. Workbench will redirect you to a login page from which you can make an account. Read More
- **Workbench Client** - The client is the main interface to Workbench. It can be run as a standalone application on Mac OS X, accessed within any modern web browser.
  - **Applications** - Workbench does lots of different things; *Workbench Applications* implement all of these different tasks on the client. Workbench includes applications for everything from behavioral design with the nodal abstraction to stochastic sequence design with Web DD. You can even write your own Workbench applications Read More
  - **Tasks** - Workbench allows you to launch and manage server tools from the client using the Task Manger. Read More

2.3 Contents of your Installation

If you downloaded Workbench as a disk image (.dmg) file for Mac OS X, several utilities have been provided to make your life a bit easier. These applications will be copied to your Applications directory as part of the installation process

- DyNAMiC Workbench.app - This contains a “site-specific broweser” – a mini web browser, automatically configured to connect to a locally hosted Workbench server. This is just for conveinience; you can just as easily navigate to the specified URL in a web browser.
- Workbench Server.app - This is a helper to allow you to launch and log in to the server. To start the server:
  1. Launch the ‘Workbench Server’ application from your Applications folder
  2. Click ‘Start’ from the server control window. You will see a Terminal window open with lots of output. You’ll also see a VirtualBox window open. Wait until you see `webserver-user@192.168.56.10's password:`; then enter the password for your server (the default is `" "`, a single space).
  3. Wait until you see `Server running from /media/sf_vmshare/infomachine2 at http://192.168.56.10:3000`. Congratulations, your server is running!

See the Getting Started guide for details on how to start on another operating system.

2.4 Installing Workbench

See Installation for details on how to install Workbench server.

2.5 Getting started

See the Getting Started guide for a quick introduction to the various things you can do with Workbench.

2.6 Further Reading

- Getting Started
- Applications

Chapter 3

File Management

Workbench has a flexible system for storing and managing files, in order to make the round-trip workflow between various applications a lot easier.

3.1 Creating files

You can easily create files within the Workbench interface:

1. Select "New" from the main menu bar
2. Enter a filename (you can choose to omit the file extension; Workbench will add it)
3. From the menu under "Create," select the type of file you would like to create. Your new file should appear in the tree on the left (labeled "Files"), and should open automatically. There are some file types which are not currently available in Workbench, but for which support is planned; these appear as greyed out menu items.
4. Later, you can re-open the file by double-clicking on its name in the Files tree, or right-clicking and selecting "Open" from the context menu.

3.2 Uploading files

To upload a file to Workbench from your desktop, just drag and drop the file from your normal file manager (Finder, Nautilus, Windows Explorer, etc.) onto the Files tree. Your new file will appear once it's been uploaded (you should see a notification).

3.3 Downloading files

To download a file, right click on the file name in the Files tree, and select "Download"; it will download as an attachment in a new window.

3.4 Renaming files

To rename a file, right click on the file name in the Files tree, and select the "Rename" field, where you can enter a new file name. As soon as you leave the field, your file will be renamed.

You can also drag and drop files in the tree in order to arrange them to different folders.

Note: Workbench creates some special files, which cannot be renamed. These files are used to store system information, such as preferences or the contents of `.package` files (see "Packages" below). You can edit these files if you wish, but you cannot rename them.

3.5 Deleting files

To delete a file, right click on the file name and select "Delete". Files deleted by Workbench cannot be recovered, unless they are independently backed up

3.6 Advanced information

3.6.1 Home directory

Each user gets a home directory on the Workbench server. This directory corresponds to that user's email address. On hosted versions of Workbench, that directory will generally be made available to users via SFTP or SSH. If you're hosting your own version of the Workbench Server, the "files" directory (where all users' files are stored) is available as a VirtualBox "shared folder" between the virtual machine where Workbench Server runs, and your host machine. During Installation you should be prompted for this location. Once you've used workbench a bit, you can access your files using your normal file manager (Finder, Nautilus, Windows Explorer), as well as managing them through the workbench user interface. Just navigate to your home directory within `(shared folder)/files/(your email address)`.

3.6.2 Packages

Packages are special folders which contain lots of files, but largely one useful set of data. They were introduced because some server tools, e.g. NUPACK, generate lots of files, which are sort of cryptic on their own, but much more useful when aggregated together into a single file. However, it may sometimes be useful to look or use the original file generated by the NUPACK executable; to address this, Workbench provides the following bargain:

- For certain server tools, the multi-file output is directed to a single folder.
- Contents of that folder are intelligenly aggregated into a single file.
- The folder is given a ".package" extension (but remains a folder, so you can open and browse it like any other)
- A "Package contents" file is written to the folder; this allows you to double-click the package itself (from the Files tree) to see the aggregated version, or to expand the package like any other folder, and view the individual output files generated by the server tool.

There is another kind of package file, called an Application Bundle, which can be used for creating custom Workbench applications. See customization for details.

Chapter 4

Workbench Applications

4.1 Behavioral Design

- Nodal Designer - Use a graphical language to specify relations between behavioral nodes; automatically convert behaviors into implementation systems.
  - Strand editor (DIL) - Graphical interface for viewing and editing a segment-level description of a nucleic acid system. Use to view the results of a nodal build, and to generate sequences with a sequence designer.
- Pepper Designer
  - Pepper intermediate language (PIL)
- CRN Designer (coming soon)

4.2 Sequence Design

- Web DD - A stochastic, domain-based sequence designer; can be used to design sequences for large systems very quickly by designing a set of noninteracting domains and then threading those domains together to form full strands.
- NUPACK Thermodynamic sequence designer - Uses Caltech's NUPACK web server to perform multi-objective thermodynamic sequence design. Enter a design using the NUPACK multi-objective sequence design script, select relevant parameters, and click "Design"; the task will be submitted to the Caltech server, and a popup window will be opened taking you to the results page.

- Multisubjective sequence designer

4.3 Simulation and Analysis

See Simulation and Analysis

- Reaction Enumerator – Enumerates the possible reactions and intermediate complexes between a set of initial complexes, at domain resolution
- NUPACK partition function calculation - Uses Caltech's NUPACK web server to compute the minimum free energy secondary structure of a strand or set of strands
- Mfold partition function calculation and MFE structure determination - Uses the University of Albany's DINAMelt web server to compute the minimum free energy secondary structure and base pair probabilities.
- TBI Vienna RNAfold partition function calculation and MFE structure determination - Uses TBI Vienna's RNAfold Websuite to compute the minimum free energy secondary structure of single DNA or RNA strands.

4.4 Utilities

- Sequence editor
- Segment Threader - Allows an arbitrary set of sequences to be threaded together into a strand. A set of named sequences can be entered using the pane on the left, and a set of strand specifications (e.g. `strand1 = a b c d b* a*`) can be entered on the right. Clicking the "Thread" button will produce a set of sequences on the bottom generated by applying the strand specification rules using the named sequences.

- Structure Editor - Accessed by "Tools" > Structure Editor from the main toolbar. Allows simple preview of a secondary structure entered in dot-paren notation.

Chapter 5

Behavioral Design

Workbench comes with two behavioral designers:

- Nodal - The nodal designer allows users to graphically design systems using the nodal formalism described by Yin et al. 2008, and focuses on behavior and geometry.
- Pepper - The Pepper compiler uses a powerful, text-based input formal for designing arbitrary strand-displacement systems

Chapter 6

Designing Nodal Systems

6.1 Overview

The nodal formalism allows you to express complicated computational or assembly processes in terms of simple behavioral units, called nodes. Many types of nodes exist—these types are called “motifs,” or “node types”; motifs are defined by mapping a structural unit of DNA or RNA (such as a hairpin) to a simple behavioral function. Nodes generally have several *ports* which correspond to *domains* of the underlying nucleic acid species. Nodal programs are written by adding instances of motifs (nodes) to a workspace, then connecting the ports together to indicate behavioral relationships. These behavioral relationships in turn imply sequence complementarities. The nodal compiler, called DyNAMiC (the Dynamic Nucleic Acid Mechanism Compiler) propagates those sequence complementarity requirements to generate a list of distinct sequences which must be designed by a sequence designer (such as DD or NUPACK).

The basic workflow is like this:

- Nodal systems are assembled using the nodal designer
- The nodal compiler runs in real-time, verifying the system as you design it.
- Once your system has been designed you can generate a DIL (DyNAMiC Intermediate Language) file, which represents your system as a scheme of complementarity relationships between segments of the strands in your ensemble (see “Compiling”, below)
- From the DIL file, DyNAMiC can generate input files for many different sequence designers.
- Sequences can be designed using sequence designers

DyNAMiC actually uses a JSON-based input format, called DyNAML. You can enter DyNAML directly using the DyNAML editor application, or you can design custom DyNAML motifs from within workbench. The DyNAML language, as well as the Nodal compiler, is described in this whitepaper.

6.2 How to design a nodal system

Here's the quick overview:

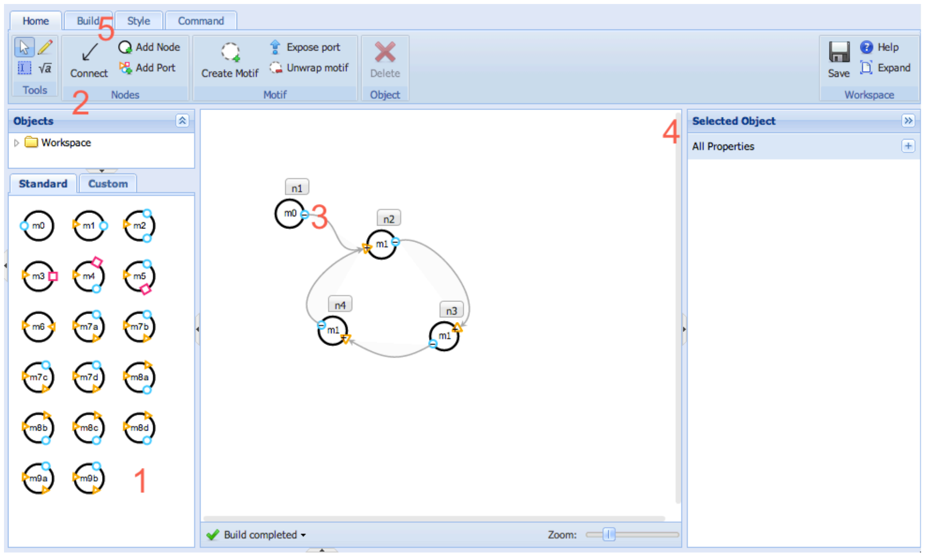


Figure 6.1: Nodal design—quick tutorial

1. Drag-and-drop pre-defined node types from the palette on the left
2. Select the "Connect" tool to connect ports together, indicating desired bindings (and thus complementarities) between nodes.
3. Click and drag from the output port (circle) of one node to the input port (triangle) of another.
4. (Optional) Select some node (by clicking on it), then expand the panel over here to examine the molecule that implements the node.
5. Click the "Build" tab, then click "Compile" to generate the system. A new window should pop up showing the segment-level view of your compiled system. Read about what to do next.

6.3 Nodal design interface

Here's a bit more detail about different parts of the interface:

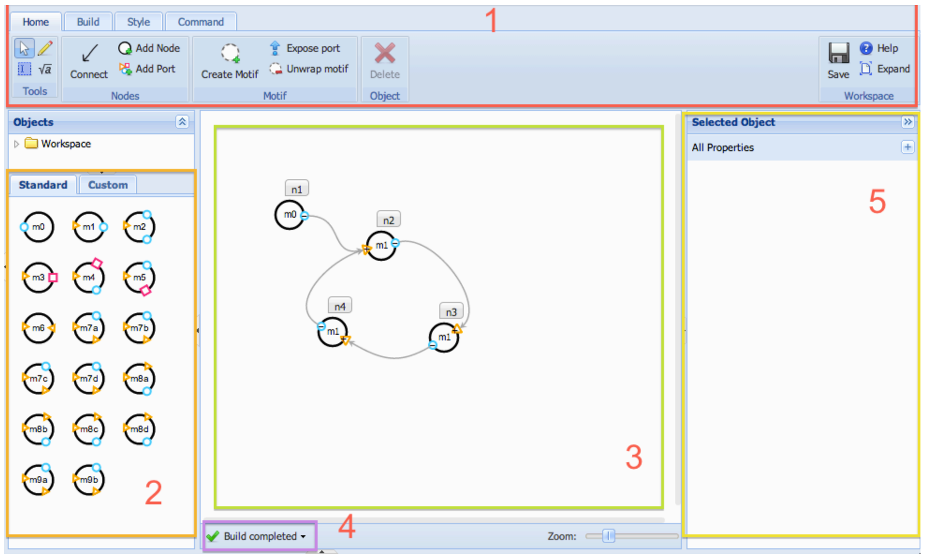


Figure 6.2: Nodal design interface

1. Ribbon – perform common actions from here; in particular, use the "Connect" tool to connect ports together, or use the pointer to go back to selecting things.

2. Palette – drag-and-drop pre-built node types from here to add them to the workspace (3). If this doesn't work, it might be because you're still using the "Connect" tool; switch back to the pointer (top-left) to add more node types. If you design a custom motif (node type), it will appear under the "Custom" tab.

3. Workspace – this is where you design systems.

4. Build status – the Nodal compiler automatically compiles your system in real-time as you edit it; this icon will show you if you have any errors. If there's an error, the affected node(s) will be highlighted in red, and you can mouse over this button to see the message.

5. Inspector – select something, then go here to see its properties. For instance, you can select a node and look at its structural implementation.

Read on for more details about different things you might want to do.

6.4 Adding Nodes

You can add nodes to the system by draging and droping existing motifs (node types) from the “Standard” panel in the lower-left, onto the workspace. You can also define new motifs (see below), which will appear in the “Custom” tab of the same panel.

6.5 Defining Motifs

You can create new motifs in two ways, each using the “Create Motif” tool in the ribbon. First, click the “Create Motif” button, then either:

- Click on the workspace to generate an empty motif. Then select this motif and use the inspector on the right to describe its implementation. You can either:
  - Click the “Edit Motif” button, and use a graphical interface to define the structure of the motif, or
  - Expand the “DyNAML Code” box and enter a custom DyNAML description. If you want to use one of the built-in motifs as template, click the “Copy from Built-in” button and select a built-in motif from the dropdown menu.
- Alternatively, click and drag on the workspace to select and existing system of nodes, and wrap that system in a new motif.
  - This will effectively remove those nodes from the system, and transform them into part of the motif.
  - All outgoing connections will be removed.
  - To “expose” ports within the motif to external complementarities, use the “Add Port” or “Expose” tool:
    * “Add Port” : Select the “Add Port” tool, then click on the motif to add a port (hold ‘alt’ for an input, or ‘shift’ for a bridge). Then use the “Expose” tool to drag from the internal port to the external (motif-level) port
    * “Expose”: Select the “Expose” tool; drag from the internal port to the motif itself; a new port will be created and exposed
  - You’ll need to instantiate the new motif by dragging and dropping it from the “Custom” panel in the lower-left.

If you decide you want to restore the nodes inside a motif to the workspace as normal nodes, select a motif and click the “Unwrap motif” button.

6.5.1 Using the Motif Editor to edit or define motifs. {#motif-editor}

The Motif Editor is a small graphical tool for defining custom motifs. It can be launched by creating a new motif (see above), selecting the motif in the workspace, and clicking "Edit Motif" in the inspector on the right side. To create a new motif:

- Start in the "Segments" pane in the top-left. Click the "Add" button to add new segments. You can click the arrow to the right to add segments of different lengths, or to add many specific sequences.
- Then use the "Strands" pane to thread segments together into strands. Click "Add" to add a new strand, and then type a strand specification in the DyNAML compact format. You can add as many strands as you'd like.
- In the "Structure" pane, click the "Strand Order" field and enter the names of your strands, separated by + signs, to describe the order in which the strands should appear in the starting complex. Once your cursor leaves the Strands field, the field will be populated with a simple graphic depicting the segments that comprise the strands you've selected. Note: If you create a strand using the "Strands" pane but don't add the strand name to the "Strand Order" field, it will be omitted from the final motif.
- Then, in the "Structure" field, enter the structure for your strand in dot-parenthesis notation. Once a valid structure has been entered, an image of your complex and a depiction of the nodal motif will appear to the right.

6.6 Complementarities

To declare complementarities between ports:

- Select the "Complementarity" tool
- Drag from one port to another to declare complementarity. You may drag from Input to output, or Bridge to bridge; other connections will be disallowed. You may also add connections between ports within a motif, but not between ports inside and outside the motif.

Complementarities may only be drawn from output ports to input ports, or between bridge (square) ports. Some connections are not possible, depending on the shape, or "footprint" of the underlying domain. For instance, a connection between a domain with 3 segments: `a b c`, each of length 8, and a domain with

4 segments `a b c d` each of length 4 would be invalid, since the total number of nucleotides (24 and 16) do not match up. Likewise, a connection between a domain with 2 segments `a b` each of length 8 (total length 16) and a domain of one segment `a` of length 16 (total length 16) would *also* be invalid, since the number of segments in each domain is different. The nodal editor doesn't prohibit these connections from being drawn, but it will highlight the relevant nodes in red and report an error. See "Errors" below for how DyNAMiC and the Nodal editor handle these errors.

You can inspect the footprint of a port by moving your mouse over the port. A small tooltip will appear and will display the length of each segment in the underlying domain

6.7 Errors

Some connections may be drawn which are invalid. In these cases, DyNAMiC will report an error, and the Nodal interface will indicate this by changing the Build status indicator (in the lower right) to read "Error". You can mouse over this field to view the error message. You can also click the field and select "Check for errors" to recompile the system, or "Show full results" to view the compiled library before sending to a sequence designer.

6.8 Compiling

Once your system is assembled, you can generate a picture of the underlying species, and generate input to sequence designers, using the "Build" tool. Select the "Build" tab in the ribbon, then click "Compile". This will generate a new file with the extension `.dil`; this is a DyNAMiC Intermediate Language (DIL) file, and it contains your compiled system. The DIL file will open automatically, and you should see the species used to generate your system. From there, you can send results to one of several sequence designers. See DIL system editor for details.

Additionally, you can bypass the DIL step and select "Build all targets" from the dropdown next to the "Compile" button. This generates several files, which will appear as siblings to your system in the files tree:

- (system)`.dynaml` - A formal, textual representation of your system (in the Dynamic Nucleic Acid Markup Language–DyNAML)
- (system)`.svg` - A graphical representation of the underlying species in your system
- (system)`.nupack` - A textual input file for the NUPACK multi-objective thermodynamic sequence designer

- (system)`.domains` - A textual input file for WebDD
- (system)`.pil` - A version of the compiled system in the Pepper Intermediate Language (PIL)

You can select several other options from the drop-down next to the “Compile” button:

- Clean – Deletes any of the above files in the same directory as your system.
- Compile with Compiler v2b – Compile with an old version of the compiler
- Compile locally – Compile in your browser (not on the server); this will not generate any output, but can be used for debugging.

Chapter 7

Pepper Systems

7.1 Overview

Pepper is a DNA circuit compiler developed by Shawn Ligocki in Erik Winfree's group at Caltech. It assists DNA programmers in building DNA computers by providing a templating language for specifying generic components and interfacing with state of the art designers and kinetic simulators to create and test sequences. Much like DyNAMiC, it can be used to design generic components and to

Pepper systems are comprised of *components* and *systems*. Components are re-usable "building blocks," which can be composed together to form systems. The rest of this documentation was written by Shawn Ligocki, and describes the syntax for Pepper component and system files.

7.2 Pepper File syntax

7.2.1 Pepper Components

A component file (name.comp) contains the specification for a single component in a DNA system (like an AND gate or a threshold, etc.). It specifies the secondary structure, kinetic and sequence constraints for that component. System files are used to connect together components.

The component syntax is based on Joe Zadeh's design specification syntax.

Comments

The pound sign (`#`) denotes that the rest of the line is a comment (python/sh style comments):

```
# This is a comment
```

Comments may appear on their own lines or after a command, like so:

```
declare component And22: x + y -> s + c  # This is the function declaration
```

Component Declaration

In the spirit of Matlab's first line declarations, each component needs to have a declaration statement at the first line of code. Syntax:

```
declare component <name> : <inputs> -> <outputs>
```

For example:

```
declare component And22: x + y  ->  s + c
```

The `<inputs>` and `<outputs>` are `+` separated lists of sequences that will be defined and used below in the specification. These sequences will be constrained on the system level and so usually represent the recognition regions of the inputs and outputs. Waste byproduct will generally not be represented here, because it will not be constrained on the system level.

`<name>` should be the same as the base name of the component file (i.e. this file should be HalfAdder.comp).

Sequences

In order to constrain some regions/subdomains of some strands to be complementary to regions on other strands we define sequence regions:

```
sequence <name> = <constraints> : <length>
```

For example:

```
sequence x = "6N S 13N S" : 21
```

The total length constraint is optional; if you don't want it, omit it and the colon.

We include the (redundant) length input for quick error checking as complex constraints can be easily misnumbered. In addition one wildcard (?) may be used instead of a number, to imply (make this as large as possible to fit the specified total length. For example:

```
sequence dx = "S ?N S" : 15
```

is the same as:

```
sequence dx = "S 13N S" : 15
```

Sequence region definitions may include previously defined regions, for example:

```
# The toehold, clamp and data region of the carry-bit output
sequence tc =  "6N" :  6
sequence cc =  "1S" :  1
sequence dc = "29N" : 29
# We combine them into a single label
sequence  c = tc1 cc1 dc1 : 36
```

So now, for example:

```
c* = dc* cc* tc*
```

We use quotes for nucleotide constraints so that we can distinguish them from other sequence names and group them (each pair of quotes denotes one domain).

Strands

Strands represent individual strand of DNA in the system. The syntax is:

```
strand <name> = <list of sequences and explicit constraints> : <length>
```

For example:

```
strand C = "?N" c : 44
```

The `"?N"` is an explicit constraint. First of all, it uses the wildcard, so it fills the remaining space on the left. Furthermore, since it is never declared as a sequence it will be constrained only by what it is forced to bind to. The idea is that you have a data region of the strand and some other regions that you're required to have, but don't want to name.

Structures

Now we glue the strands together to make multi-stranded (or single-stranded) structures:

```
structure <name> = <list of strands> : <secondary structure>
```

So continuing the above example:

```
structure Gate = X + C + S + Y : U6 H15(+ H15(U29 + U14 H15(+)))
```

using Joe Zadeh's (H)elix (U)npaired notation or, say

```
structure Gate = X + C + S + Y : 6. 15( + 15( 29. + 14. 15( + 45)
```

in a sort of shorthand dot-paren notation (where each number is a multiplier on the symbol following it).

By default it is assumed that you want to optimize the thermodynamics of the structure to be as close to the specified secondary structure as possible. If not, you can tell the compiler not to optimize a specific structure. For example:

```
structure [no-opt] Gate = X + C + S + Y : U6 H15(+ H15(U29 + U14 H15(+)))
```

This can be useful for two reasons. First, you may with to impose base-pairing constraints but don't want thermodynamic optimization, for example, because you know this structure is not the MFE structure. Second, you may want the compiler to know that these strands should be grouped together as a complex, but the intended complex has psuedoknots that cannot (yet) be properly expressed in Pepper.

Alternatively for limited optimization:

```
structure [10nt] Gate = X + C + S + Y : U6 H15(+ H15(U29 + U14 H15(+)))
```

This aims to be within 10 nucleotides of the specified structure, on average. Note that how these parameters are interpreted is up to the back-end sequence designer. E.g. Zadeh's designer will try to minimize the average number of incorrect nucleotides; another designer might want the MFE structure to be within 10 nucleotides; the SpuriousC designer could use the parameter for some kind of weighting, but currently doesn't.

We can also specify the secondary structure on the domain level:

```
structure Gate = X + C + S + Y : domain .(+(.+.(+)))
```

assuming, for example, that the strands had been defined as

```
strand X = toe_x data_x

strand C = carry "29N"

strand S = "14N" sum

strand Y = sum* carry* data_x*
```

Here each dot-paren represents an entire domain.

Kinetics

Now all we have left is to explain the desired kinetics, what structures will interact and what will they produce:

```
kinetic <input structures> -> <output structures>
```

So if we're working with the half adder, we might have:

```
kinetic inX + Gate        ->  waste_X + inter_G

kinetic inY + inter_Gate  ->  waste_Y + outS + outC
```

I might allow some optional parameters to fine tune these, maybe specifying fuzzy states or desired speed of reactions.

Examples

- DNA compiler/And22.comp

7.2.2 Pepper Systems

A system file (name.sys) contains a specification for the connectivity of components in a DNA system. It specifies each component and and ties their signal sequences together. A system may be used as a component in a larger system.

Comments

The pound sign (#) denotes that the rest of the line is a comment (python/sh style comments): # This is a comment Comments may appear on their own lines or after a command, like so:

```
import And22  # Half adder is used in the first layer only
```

System Declaration

In the spirit of Matlab's first line declarations, each system needs to have a declaration statement at the first line of code. Syntax:

```
declare system <name>: <inputs> -> <outputs>
```

For example:

```
declare system HalfAdder: x0 + y0  ->  s0 + c1
```

The `<inputs>` and `<outputs>` are '+' separated lists of sequences that will be constrained to components below.

These are used if this is a system being used as a component. However, for top-level systems, we will not have inputs and outputs and so

`<name>` should be the same as the base name of the component file (i.e. this file should be HalfAdder.comp).

Imports

In order to use a component file you must import it first (python-style). Syntax:

```
import <component name> For example:
```

```
import And22
```

You may import from a different directory/name than the name you use in the file. For example:

```
import Georg0711/Parallel_And22 as And22
```

which would import from the component file 'Parallel_And22.comp' from the directory 'Georg0711/', but still use the name And22 in the specification.

Components

This is the meat of the circuit file. You must make one statement for each component in the DNA system specifying the input and output sequences. The syntax is:

```
component <name> = <component name>: <list of input sequences> -> <list of output sequenc
```

for example

```
component G0_01 = And22: nx0 + y0  ->  s0 + nc1
```

Note: Components **may** have 0 inputs or 0 outputs (or even both, if you can find that useful). Example:

```
component DNS0 = Detector: ns0 ->
```

Is a ‘ns0’ detector. It will activate if ns0 is input but doesn’t produce any outputs for downstream gates.

Examples

- DNA compiler/Half Adder
- DNA compiler/Two-Bit Adder

Chapter 8

Segment-level/Domain-level Systems

Segments, known elsewhere in the literature as *domains*, are contiguous regions of basewise complementarity. For instance, the following strand:

```
AAAAA GGGG TTTT
```

Might be considered to consist of three segments. Segment-level designs are often more convenient than behavioral or sequence-level designs.

Workbench has several facilities for handling systems at a segment level:

- The DyNAMiC Intermediate Language is the result of compiling Nodal systems using DyNAMiC. DIL features a sophisticated graphical editor, and can be exported to various sequence designers, as well as the reaction enumerator.
- Pepper Intermediate Language is the result of compiling a Pepper system with the Pepper compiler.

Chapter 9

DyNAMiC Intermediate Language (DIL)

The DyNAMiC Intermediate Language (DIL) is the intermediate representation generated by the Nodal compiler, DyNAMiC. DyNAMiC converts behavioral representations into molecular implementations using a set of pre-defined translation rules; these rules map "node types" to "molecule types." The result of nodal compilation is therefore a segment-level representation of a set of complexes with a number of explicit Watson-Crick complementarity and orthogonality relationships, but no real nucleotide sequences.

The DIL Editor lets you visualize the primary and secondary structure of these complexes, specify sequence constraints for the segments which compose them, and export these structures to various sequence designers.

9.1 Viewing and Editing systems

To design a system from scratch, follow this procedure, guided by the figure:

1. In the *Segments* pane, click "Add" to add several segments (you can edit their names, sequences, or colors if you'd like)
2. In the *Strands* pane, click "Add" to add several strands.
3. Click the blank space under the "Segments" column (still in the Strands pane), and type a description of the strand. You can either use DyNAML short notation to group segments into domains, or you can just list the segments you wish to use, separated by spaces (e.g. `1 2 3* a 4 x`, etc.); the "Sequence" column should be filled in automatically with the sequence you've just described

4. In the main *Complexes* pane, click “Add” to add a complex; select the new complex and click “Edit” or double-click the complex name to view a pop-up editor for the complex

5. Under “Strand Order”, click the box (which says “Click to add”), and type the names of each of the strands you want in the complex, *separated by + signs.*

6. Under “Structure”, enter a segment-wise structure for the complex, in dot-parenthesis notation. Once you have entered a valid structure, you should see a preview in the right-hand pane. You can close the complex editor popup once you’re done.

The following sections describe the interface in more detail.

9.1.1 Ribbon

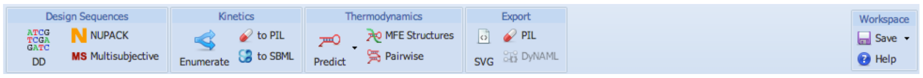


Figure 9.1: Ribbon toolbar

Use the ribbon toolbar to design sequences, explore system kinetics, make thermodynamic calculations, or export the system to a Scalable Vector Graphics (SVG) file.

9.1.2 Complexes pane

The main interface pane displays a grid of complexes. Complexes are displayed as a secondary structure “planar graph.” A “View” menu in the lower-right corner of this panel may be used to modify the visualization; bubbles representing the individual nucleotides may be colored according to segment identity, domain identity, or base identity. Alternatively, the bubbles may be hidden and text labels may be used instead.

9.1.3 Segments pane

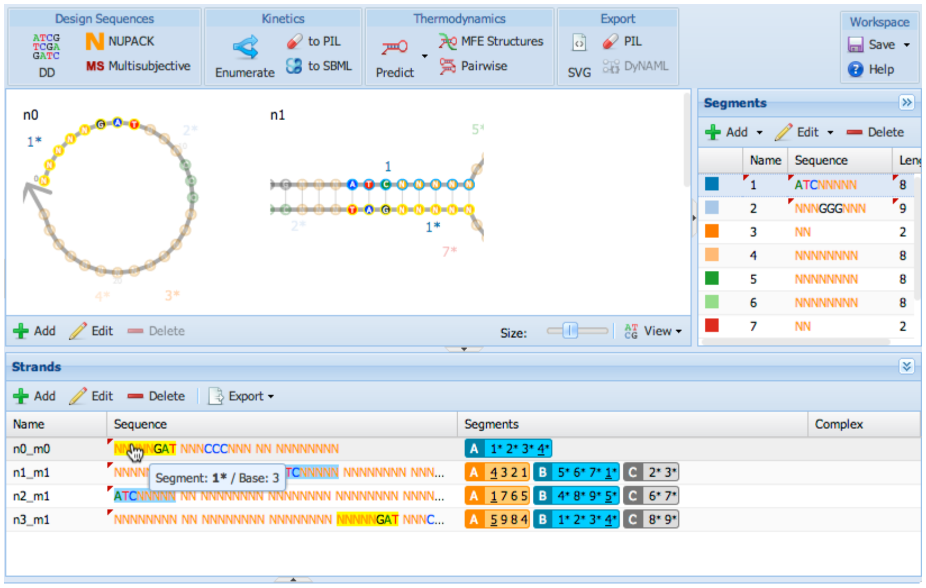
Shows the sequences of the segments which compose the system. Segments are discrete regions of sequence complementarity. A segment’s sequence can be edited by double clicking on the sequence and typing a new one; the complexes and strands panes will be updated automatically.

9.1.4 Strands pane

Shows the strands which comprise the system. Strands may not currently be edited directly, but their composite sequence and component segments/domains may be viewed.

9.1.5 Interactivity

You can mouse over segments in the Segments or Strands pane in order to highlight them elsewhere in the system. This can be useful for tracing where a particular segment has propagated within the system. The segment will be highlighted in yellow, and its complement will be highlighted in blue.



Segments

	Name	Sequence	Len
🟦	1	ATCNNNNN	8
🟦	2	NNNGGGNNN	9
🟧	3	NN	2
🟧	4	NNNNNNNN	8
🟩	5	NNNNNNNN	8
🟩	6	NNNNNNNN	8
🟥	7	NN	2

Strands

Name	Sequence	Segments	Complex
n0_m0	NNNNNGAT NNNCCCNNN NN NNNNNNNN	A 1* 2* 3* 4*	
n1_m1	NNNN Segment: **1*** / Base: 3 TCNNNNN NNNNNNNN NNN...	A 4 3 2 1 B 5* 6* 7* 1* C 2* 3*	
n2_m1	ATCNNNNN NN NNNNNNNN NNNNNNNN NNNNNNNN NNNN...	A 1 7 6 5 B 4* 8* 9* 5* C 6* 7*	
n3_m1	NNNNNNNN NN NNNNNNNN NNNNNNNN NNNNNGAT NNNC...	A 5 9 8 4 B 1* 2* 3* 4* C 8* 9*	

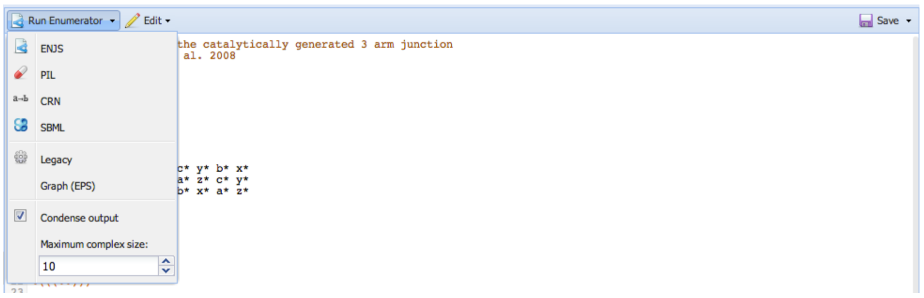
Figure 9.2: Interactively highlight segments

9.2 Designing sequences

From the ribbon, DIL systems may be exported to various sequence designers, including Web DD, NUPACK, and Multisubjective. Simply click the appropriate button Clicking these buttons will generate a file and open the requisite sequence designer.

9.3 Enumerating reactions

From the ribbon, DIL systems may be exported to a file readable by the reaction enumerator. Simply click the “Enumerate” button, and an enumerator tab will open. Then click “Run enumerator” to run the reaction enumerator:



☑ Condense output

Maximum complex size: 10

```
the catalytically generated 3 arm junction
 al. 2008

c* y* b* x*
a* z* c* y*
b* x* a* z*
```

Figure 9.3: Run enumerator button and menu

See the Enumerator page for details.

9.4 Evaluating thermodynamics

From the ribbon, you can invoke various thermodynamic calculations. Click “Predict” to open a window for analysis using NUPACK, or click the arrow to analyze using Mfold or RNAfold. You can use the “MFE Structures” button to open a NUPACK analysis that will examine only single-stranded complexes (the “complex size” parameter is set to 1); the “Pairwise” button will open a NUPACK analysis with the “complex size” parameter set to 2.

Chapter 10

Pepper Intermediate Language (PIL)

The Pepper Intermediate Language (PIL) allows concise description of segment-level systems. PIL files can be generated by the Pepper compiler, or by the Nodal compiler. PIL was developed by Shawn Ligocki, Chris Berlind, Joseph Schaeffer, and Erik Winfree. See their whitepaper for full details.

PIL follows the basic constructions of the Pepper language, but disallows some features (sequence constraints, components, etc.)

Each PIL line should consist of a directive of one of the following forms:

- `sequence <name> = <sequence>` – declare a new domain
- `strand <name> = <list of sequences and explicit constraints>` – declare a strand comprised of domains
- `structure <name> = <list of strands> : <secondary structure>` – declare a complex comprised of several strands
- `kinetic <input structures> -> <output structures>` – declare a reaction between several structures

Additionally, PIL lines beginning with a `#` sign will be treated as whitespace.

Please see the original Working with Sequences =======================

Workbench has many powerful tools for working with sequences of bases.

10.1 Sequence Editor

The sequence editor provides lots of tools for working with sequences as standard text files: The editor color-codes bases in the sequence, allows various metrics to be easily calculated, and exposes functions for formatting, transforming, and manipulating sequences. To view details about each of the menu items in the sequence editor, just hover your mouse over it; help will appear in a tooltip.

Read more about the sequence editor

10.2 Sequence Designers

- Web DD - A stochastic, domain-based sequence designer; can be used to design sequences for large systems very quickly by designing a set of noninteracting domains and then threading those domains together to form full strands.
- NUPACK Thermodynamic sequence designer - Uses Caltech's NUPACK web server to perform multi-objective thermodynamic sequence design. NUPACK produces thermodynamically optimized sequences, but can be slow for some systems. To use: Enter a design using the NUPACK multi-objective sequence design script, select relevant parameters, and click "Design"; the task will be submitted to the Caltech server, and a popup window will be opened taking you to the results page.
- Multisubjective sequence designer (coming soon)

10.3 Analysis and Simulation

NOTE: The below features are deprecated. Please see Simulation and Analysis instead.

Workbench provides an interface to the powerful nucleic acid computation package NUPACK. To perform NUPACK calculations of sequences, from the sequence editor, select a set of strands, then select one of the available calculations from the "Compute" menu:

- MFE Complexes - Computes the structure, free energy, and relative concentrations of the various complexes expected at equilibrium. To speed up computation, you can select a maximum complex size to prevent NUPACK from comparing complexes containing more than the indicated number of species.
- Pairwise MFE Complexes - Same as MFE, except the max complex size defaults to 2.

- Subsets MFE - not implemented
- Brute force - not implemented

Chapter 11

Sequence Editor

Workbench features a powerful text editor for working with DNA sequences. The sequence editor can be accessed from within sequence designers like DD and Multisubjective, by creating a `.seq` file, or by opening *Tools > Sequence Editor* from the main toolbar.

Most operations in the sequence editor operate on a set of strands. Each separate line is assumed to be on a separate strand; if a single line is selected, sequences separated by spaces or tabs will be treated as different strands. Most operations will replace the selection with the indicated transformation (e.g. selecting "Transform", then "Reverse" will replace the selected sequence with its bases in reverse order). For operations in the "Transform" menu, this can be disabled by unchecking "Replace Selection"; this will cause the transformation to be inserted below each strand.

Within the sequence editor, all operations can be accessed by the menus in the toolbar:

11.1 Sequence Editor Tools

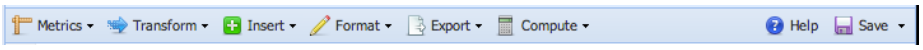


Figure 11.1: Sequence Editor toolbar

- Metrics – calculate various properties between sequences. Select one or two sequences (separated by newlines, tabs, or spaces), then choose one of these options:
  - Levenshtein distance – Measures the difference between two strings, allowing for substitutions, insertions, and deletions

- Hamming distance – Measures the difference between two strings, allowing only for substitutions
- Shannon entropy – Measures the information entropy of a sequence, giving a metric of base diversity

- Transform – modifies the selected sequence. If "Replace selection" is checked, the selection will be replaced with the result of the transformation; else, the result will be inserted on the line below the selection. If multiple lines, or sequences separated by spaces or tabs are selected, each will be handled individually.
  - Reverse – reverses the order of a sequence
  - Complement – gives the Watson-Crick complement of a sequence
  - Reverse Complement – reverses and complements the sequence
  - Duplicate – duplicates the sequence
  - Truncate – removes bases from the beginning or end of a sequence. Use a positive number to truncate from the left (5' end), use a negative number to truncate from the right (3' end)
  - To DNA – replaces uracil (U) bases with thymine (T)
  - To RNA – replaces thymine (T) bases with uracil (U)
  - Pairwise Align – attempts to align two sequences to minimize the Levenshtein distance
- Insert
  - Insert Poly-X Sequence – inserts the provided sequence, repeated several times. For instance, specify `N` and `6` to insert `NNNNNN`, or `AT` and `4` to insert `ATATATAT`
  - Generate Poly-X Sequence – generates a random sequence according to a pattern of bases or degenerate bases specified (e.g. `NNNN` -> `ATCG`; `YYYY` -> `CTCT`, etc. Specify which base(s) (e.g. A, T, C, G, N, ...) you would like to use; degenerate bases following the IUPAC nucleotide codes are allowed and will be replaced by randomly selected bases matching the specification (e.g N becomes any base, R becomes A or G, Y becomes C or T, etc.). Then specify how many copies of the pattern you would like generated.
- Format
  - Strip Extra Characters – Removes characters other than A, T, C, G, U, and whitespace from the selection
  - Strip Whitespace – Removes whitespace characters (spaces, tabs, line returns) from the selection
  - Strip Newlines – Removes line return characters from the selection

  - Insert Line Breaks – Replaces spaces in the selection with line returns
  - Uppercase – Converts the selection to uppercase
  - Lowercase – Converts the selection to lowercase
  - Comment/Uncomment – Adds/removes comment characters (`#` or `%`); that is, if the selected line(s) begin with a `#` or `%`, removes them; otherwise, inserts a `#` sign before each line.

- Export
  - To FASTA – converts strands named in any format to the FASTA format used by the NCBI.
  - To NUPACK – converts strands named in any format to the NUPACK script format (e.g. `domain NAME = SEQUENCE`)
  - To Excel/TSV – converts strands named in any format to a tab-separated values format, suitable for copy-and-paste into Excel
  - To CSV – converts strands named in any format to a comma-separated values format, suitable for reading by many spreadsheet programs
  - To Plain Text – removes strand naming
  - Make DD Input File – generates an input file for classic DD
  - Thread Sequences to Strand – Opens the sequence threader
  - Name Strands – automatically prepends a name, consisting of a prefix and a number, to each sequence on a line. For instance, if the prefix were `n`, lines would be prefixed with `n1 :`, `n2 :`, `n3 :`, etc.
  - Prepend strands – prepends each line with some prefix text

- Calculate – Allows the partition function and/or MFE secondary structure(s) to be calculated using one of several thermodynamic analysis web servers. See Analysis and Simulation for details.

11.2 IUPAC Degenerate Bases

The following table lists the IUPAC degenerate base codes recognized by DyNAMiC:

Code	A	C	G	T	Mneumonic
A	A				adenosine
C		C			cytidine
G			G		guanine
T				T	thymidine
U				U	uridine
K			G	T	keto
S		C	G		strong
Y		C		T	pyrimidine
M	A	C			amino
W	A			T	weak
R	A		G		purine
B		C	G	T	not A
D	A		G	T	not C
H	A	C		T	not G
V	A	C	G		not T
N	A	C	G	T	A/G/C/T (any)

11.3 Sequence Editor interactivity

In addition to the utilities, the sequence editor will interactively highlight other instances within the design of the selected sequence. The status bar in the lower-right hand corner will show the number of bases and strands (lines) selected.

See also Working with sequences.

Chapter 12

Thermodynamic analysis and sequence design with NUPACK

Workbench provides facilities for easily starting analysis and sequence design tasks on the NUPACK web server, operated by the Niles Pierce Lab at Caltech.

NUPACK dialogues can be opened from a few places in workbench:

- Select *Tools* in the main toolbar atop Workbnech, then *NUPACK*, then either *Analyze*, *Design*, or *Utilities*
- From within a sequence editor, select *Compute*, then *Calculate Partition Function*

12.1 Thermodynamic Analysis

Enter sequences in the main window, separated by newlines. If desired, modify the advanced options, and click "Analyze"

12.2 Sequence Design

Enter a NUPACK multi-objective design script, then click "Design"

Chapter 13

Sequence Design with Multisubjective

Multisubjective is a multi-paradigm sequence designer that combines the rapid iteration of heuristic designers such as Web DD with the thermodynamic analysis capabilities of NUPACK.

Multisubjective (`.ms`) files can be generated using DIL. See DIL for details.

Chapter 14

Simulation and Analysis

14.1 Thermodynamic analysis packages

14.1.1 NUPACK partition function calculation

Uses Caltech's NUPACK web server to compute the minimum free energy secondary structure of a strand or set of strands. See help from Caltech here. Read more about NUPACK in DyNAMiC

14.1.2 Mfold partition function calculation and MFE structure determination

Uses the University of Albany's DINAMelt web server to compute the minimum free energy secondary structure and base pair probabilities. See help from University of Albany here. Read more about Mfold in DyNAMiC

14.1.3 Vienna RNAfold partition function calculation and MFE structure determination

Uses TBI Vienna's RNAfold Websuite to compute the minimum free energy secondary structure of single DNA or RNA strands. See help from TBI here. Read more about RNAfold in DyNAMiC

Chapter 15

Domain-level reaction enumeration

Reaction enumeration is the process of calculating all possible reactions between, and intermediate complexes formed, from an initial set of complexes. The process proceeds essentially as follows: we beginning with a pool of starting complexes, then calculate all possible "fast" (unimolecular) reactions within this pool. Any complexes (or cycles of complexes) without any outgoing "fast" reactions is deemed a "resting complex;" all other species are designated "transient" Subsequently, "slow" (bimolecular or higher-arity) reactions are then computed between these resting complexes. In DyNAMiC, this process is performed between *domains* (known elsewhere within DyNAMiC as *segments*)—continuous regions of complementarity.[1] That is, rather than considering each distinct base pairing as a different secondary structure, reactions are only considered between longer regions of nucleotides. Pseudoknotted intermediates are not considered.

Once reactions have been enumerated, the enumerator can also generate a "condensed" network of reactions; this network groups all "resting states" (individual resting complexes or cycles of resting complexes connected by fast reactions), and shows only reactions between these resting states. Such a network helps visualize more complex reaction networks by occluding some detail.

The reaction enumerator was originally developed by Karthik Sarma, Brian Wolfe, and Erik Winfree at Caltech, and was also based on algorithms developed by Seung Woo Shin; it has been debugged and extended by Casey Grun at Harvard.

[1] In the Nodal compiler and formalism, a distinction is drawn between *segments* (continuous regions of complementarity), and *domains* (groups of segments with a particular behavioral function, such as input or output). In this sense, the reaction enumerator operates on *segments*; here we use the terms *domain* and *segment* interchangeably to refer to a continuous region of complementary bases.

15.1 Preparing input for the enumerator

The enumerator accepts several text-based input formats:

- *Standard input format* (`.enum`) – This is a simple format that is specific to the enumerator. A simple example of the format is included below. The format has three types of statements:
  - `domain` statements declare individual domains, as follows: `domain name : specification`, where:
    - `name` is the name of the domain (e.g. `a`, `1`, `th`, etc.)
    - `specification` is either the length of the domain (e.g. a number of bases, or just `long` or `short`) or a sequence (e.g. `NNNNNNN` or `ATTACG` or even a mixture of specific and degenerate bases `AANATCY`)
  - `strand` statements group domains into strands, as follows: `strand name : domains`, where:
    - `name` is the name of the strand
    - `domains` is a space-separated list of domains
  - `complex` statements group strands into complexes and assign them a secondary structure, as follows:

    ```
    complex name :
    strands
    structure
    ```

    where:
    - `name` is the name of the complex
    - `strands` is a space-separated list of strands
    - `structure` is a domain-wise description of the structure in dot-parenthesis notation
- Pepper Intermediate Language (`.pil`) – PIL is a general-purpose format for describing domain-level secondary structures. See the PIL page for details.

Input files for the enumerator can also be generated automatically by the graphical DIL editor.

Here is a simple example of the standard input format:

```
# This file describes the catalytically generated 3 arm junction
# described in Yin et. al. 2008
```

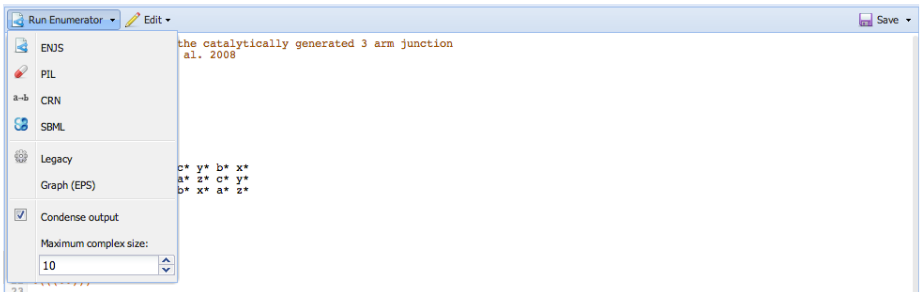
```
domain a : 6
domain b : 6
domain c : 6
domain x : 6
domain y : 6
domain z : 6

strand I : y* b* x* a*
strand A : a x b y z* c* y* b* x*
strand B : b y c z x* a* z* c* y*
strand C : c z a x y* b* x* a* z*

complex I :
I
....

complex A :
A
.(((..)))

complex B :
B
.(((..)))

complex C :
C
.(((..)))

complex ABC :
A B C
((((((((. + ))))((((. + )))))))).
```

15.2 Running the enumerator

Once an input file has been created, you can use the “Run enumerator” button to invoke the enumerator. Click the arrow next to the button to select a specific input format or set advanced options:

There are 6 output formats available; you may click any of these menu items to generate output in that format, which will open in a new window:

- Graphical results (ENJS) – produces a file which can be rendered into a graphical, interactive network by DyNAMiC and exported to SVG. This file is also a valid JSON file and may be suitable for consumption by other tools.



```
the catalytically generated 3 arm junction
 al. 2008

c* y* b* x*
a* z* c* y*
b* x* a* z*
```

Figure 15.1: Run enumerator button and menu

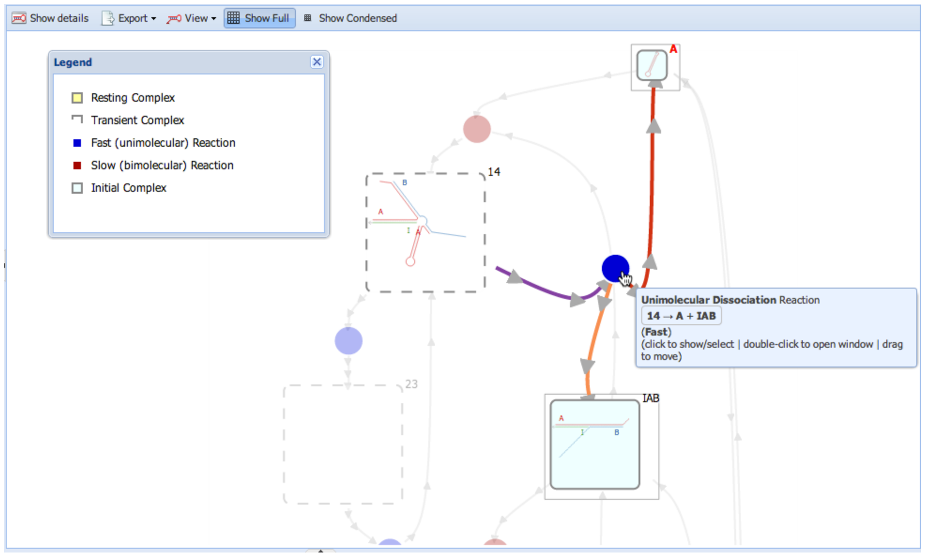
- Pepper Intermediate Language (PIL) – produces a representation of the network, including reactions, in the Pepper Intermediate Language
- Chemical Reaction Network (CRN) – produces a list of simple reactions between chemical species
- Systems Biology Markup Language (SBML) – produces a representation using the Systems Biology Markup Language, an industry standard format for modeling biological and chemical networks. SBML can be consumed by a reaction simulator, such as COPASI
- Legacy – produces output in the format of Brian Wolfe's old enumerator
- Graph (EPS) – produces an EPS file showing the reaction network, laid out using Graphviz

In addition, the following options may be set:

- Condense output – Check this box to generate the condensed version of the reaction graph. For the ENJS output format, both the condensed network and the full network of reactions will be viewable; for all other output formats, only the condensed network will be written.
- Maximum complex size – Select a maximum number of complexes (beyond which the enumeration should be truncated); this allows systems e.g. with potentially infinite polymers to be enumerate in finite time.

15.3 Viewing enumeration results

Once enumeration is completed, results should open in a new tab automatically. For the text-based output formats (PIL, CRN, SBML, etc.), this tab will show a text editor. For the graphical output format (ENJS), you should see an interactive, graphical representation of the network:



Unimolecular Dissociation Reaction
14 → A + IAB
(Fast)
(click to show/select | double-click to open window | drag to move)

Figure 15.2: Graphical enumeration results

- Complexes are represented by rectangles, while reactions are represented by colored circles (*nodes*). Complexes and reactions are connected by grey lines (*links*). Complexes are colored and outlined colored to indicate whether they are initial, resting or transient.
- You can grab and drag the white background to navigate, and use the mouse wheel to zoom in and out.
- Mouse-over a complex or reaction to view a description in a tooltip, and to highlight incoming and outgoing links; cool colors show incoming links, while warm colors show outgoing links.
- Click a complex to show its secondary structure; double-click a complex name or a reaction node to view details of the complex or reaction (see below)
- Complexes, reactions, and lines may be dragged to re-arrange the graph for readability.
- Resting states (groups of resting complexes) are outlined in grey

15.3.1 Enumeration results toolbar

- *Show Details* – if you click to select a complex or reaction, it will be outlined in red; you can then click “Show Details” to view the complex or reaction details (see below)

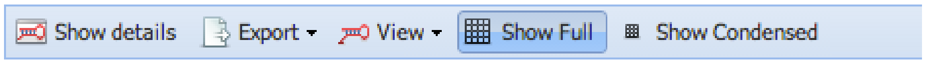


Figure 15.3: Enumeration results toolbar

- *Export* – you can export the entire reaction graph, in its current view, as a Scalable Vector Graphics (SVG) file which can be viewed or edited in Adobe Illustrator, Inkscape, or similar vector graphics programs.
- *View* – you can choose to view all secondary structures in the network according to various coloring schemes (e.g. by domain, by strand, or by base identity) and view options; for instance, you can choose to view or hide labels for all domains and strands, number bases on the backbone, and view bases as lines, text, or circles ("bubbles").
- *Show Full* – you can toggle whether the full network of uncondensed reactions is displayed
- *Show Condensed* – you can toggle whether the condensed network of reactions between resting states is displayed

15.3.2 Viewing complex and reaction details

If you double-click on a complex name or a reaction node, a small window will open showing details of the object; this is useful if you would like to compare objects on different parts of the network.

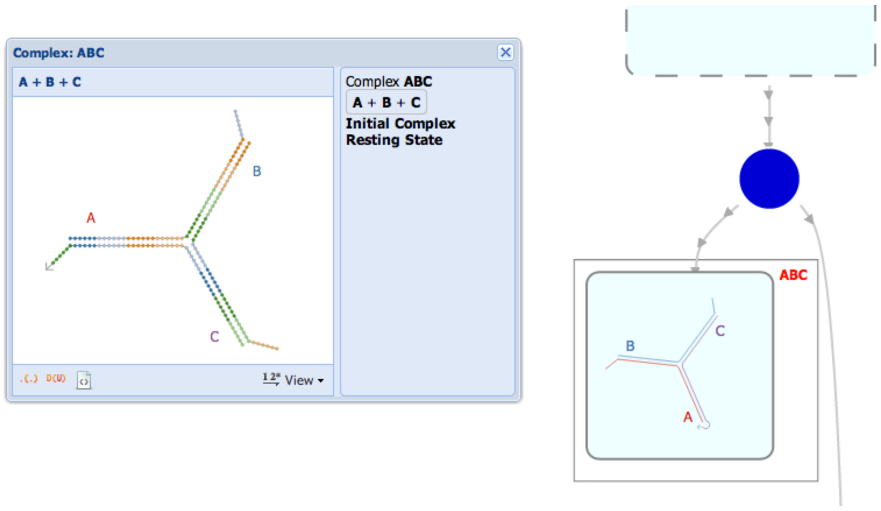


Figure 15.4: Complex details

In complex detail view, you can view the secondary structure of the complex and the constituent strands. You can copy the structure in dot-parenthesis or DU+ notation, or download it as an SVG file. You can apply several view options to the complex as well.

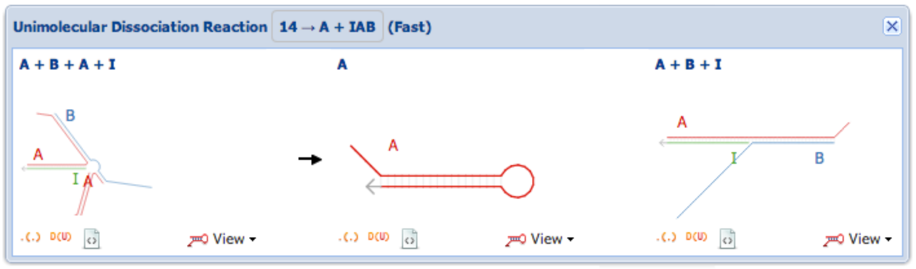


Figure 15.5: Reaction details

In reaction detail view, you can see a complex preview for each complex in a reaction.

15.3.3 Viewing full and condensed reactions

Using the *Show Full* and *Show Condensed* buttons in the toolbar, you can choose to view the full reaction graph, the condensed reaction graph, or both overlaid. For the condensed reaction graph, reaction nodes are larger and shown in a lighter color, while links are darker.

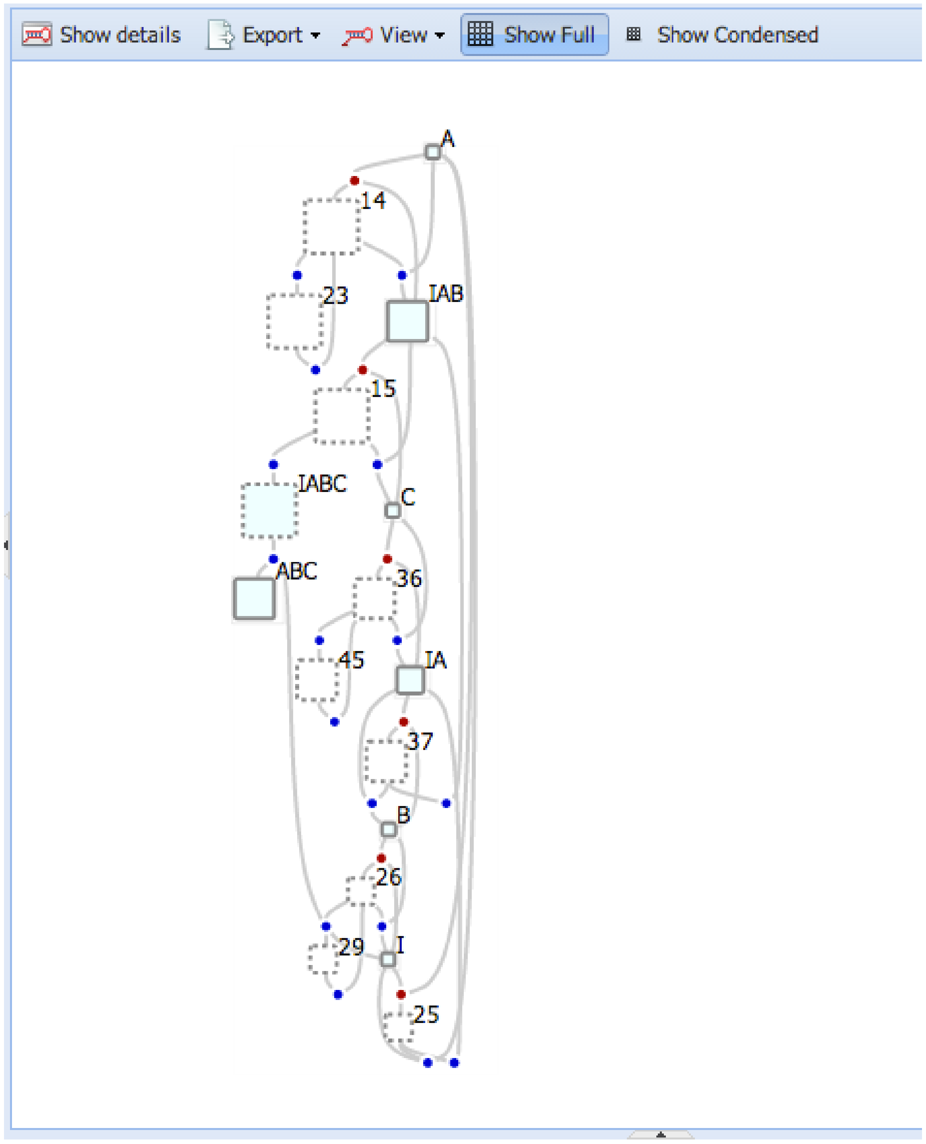


Figure 15.6: Full reaction graph

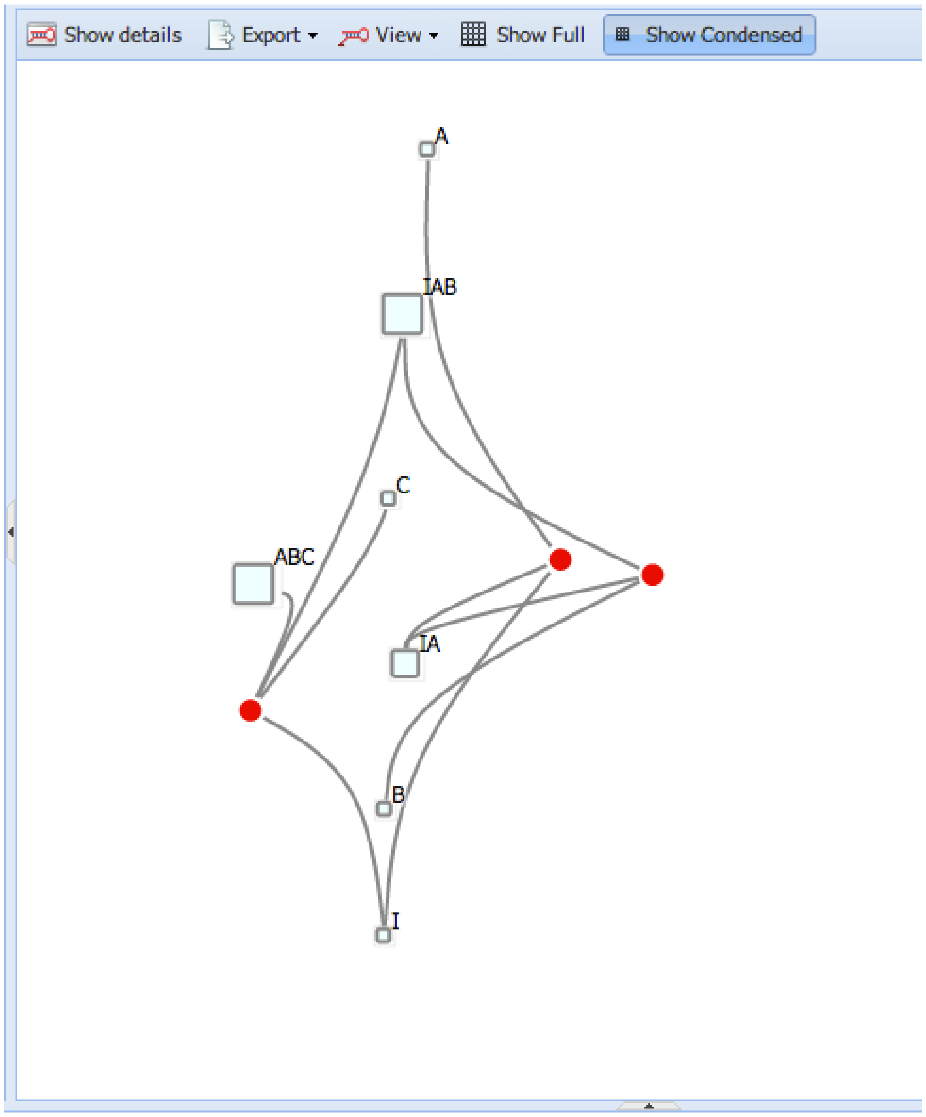


Figure 15.7: Condensed reaction graph

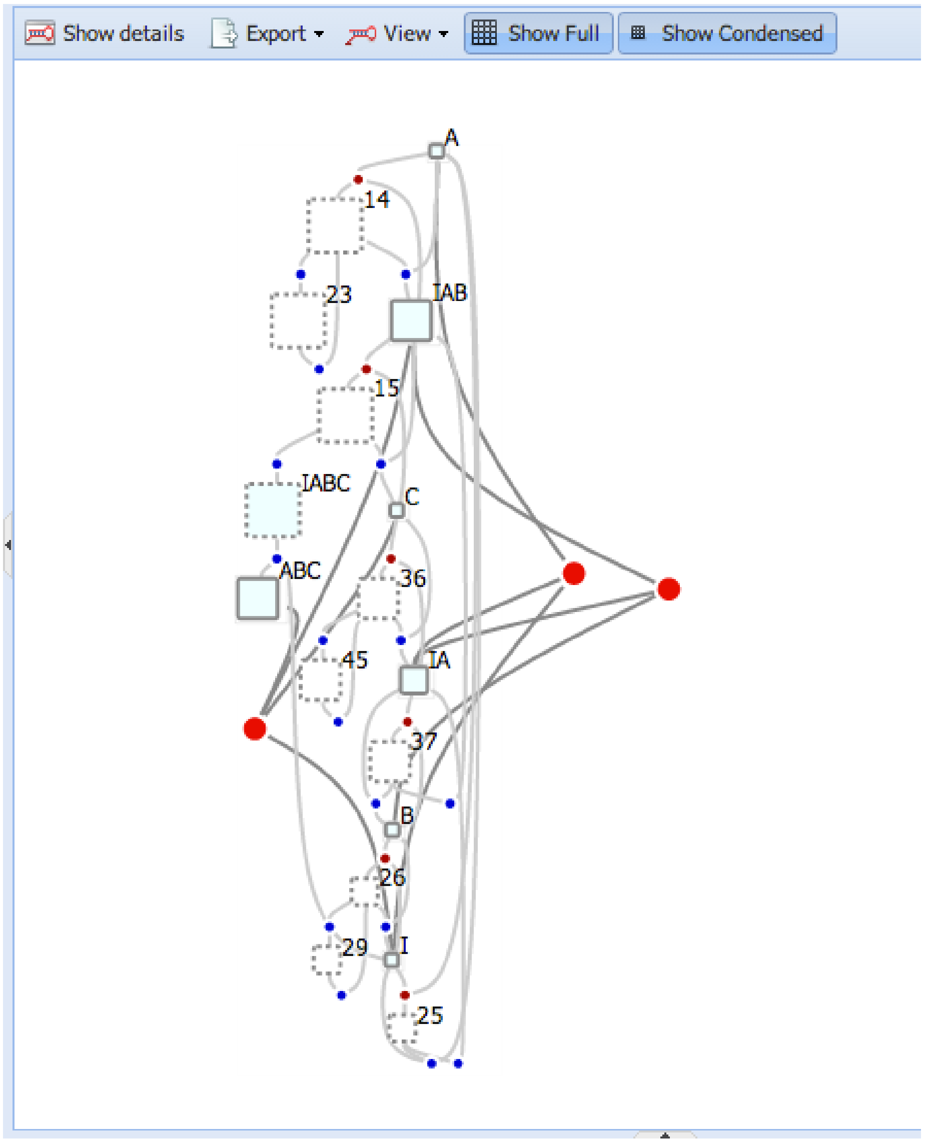


Figure 15.8: Both reaction graphs overlaid

Chapter 16

Thermodynamic analysis and sequence design with NUPACK

Workbench provides facilities for easily starting analysis and sequence design tasks on the NUPACK web server, operated by the Niles Pierce Lab at Caltech.

NUPACK dialogues can be opened from a few places in workbench:

- Select *Tools* in the main toolbar atop Workbnech, then *NUPACK*, then either *Analyze*, *Design*, or *Utilities*
- From within a sequence editor, select *Compute*, then *Calculate Partition Function*

16.1 Thermodynamic Analysis

Enter sequences in the main window, separated by newlines. If desired, modify the advanced options, and click "Analyze"

16.2 Sequence Design

Enter a NUPACK multi-objective design script, then click "Design"

Chapter 17

Thermodynamic analysis with RNAfold

Workbench provides facilities for easily starting analysis tasks on the RNAfold web server, operated by the Theoretical Biochemistry Group (TBI) at the University of Vienna.

RNAfold dialogues can be opened from a few places in workbench:

- Select *Tools* in the main toolbar atop Workbnech, then *Vienna RNA*, then *RNAfold Server*
- From within a sequence editor, select *Compute*, then *Calculate Partition Function*

See additional documentation from TBI

Chapter 18

Thermodynamic analysis with Mfold

Workbench provides facilities for easily starting analysis tasks on the Mfold/UnaFold web server, operated by Michael Zuker & Nick Markham of the RNA Institute at the State University of New York at Albany.

RNAfold dialogues can be opened from a few places in workbench:

- Select *Tools* in the main toolbar atop Workbnech, then *Mfold*, then *Quik-Fold*
- From within a sequence editor, select *Compute*, then *Calculate MFE*

See additional documentation from SUNY Albany

Chapter 19

Utilities

Workbench includes several utilities for handling common tasks in DNA system design. You can access these utilities from the "Tools" menu in

19.1 Structure Editor

Enter a base-wise secondary structure in dot-parenthesis notation. Click "Preview" to update the graphical preview. Use the "View" menu to select various view options. Use the buttons in the lower-left to convert the structure to:

- Dot-parenthesis notation
- DU+ notation
- Scalable Vector Graphics (SVG) – Exports the current view as an SVG file suitable for editing in programs like Adobe Illustrator or Inkscape.

19.1.1 Secondary Structure View Menu

Choose one of four coloring modes, and several additional options from the "View" menu.

There are four coloring modes:

- Segments – Color the base bubbles, letters, or backbones based on the segment to which they belong
- Domains – Color the base bubbles, letters, or backbones based on the domain to which they belong

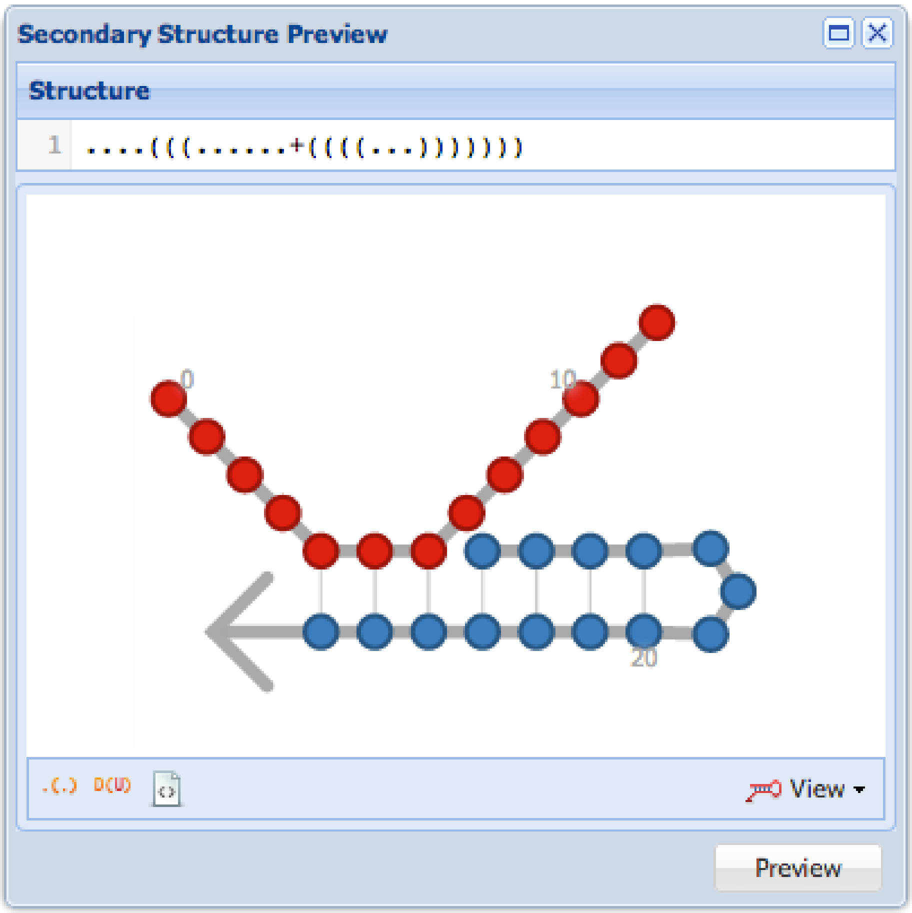


Figure 19.1: Secondary Structure Tool

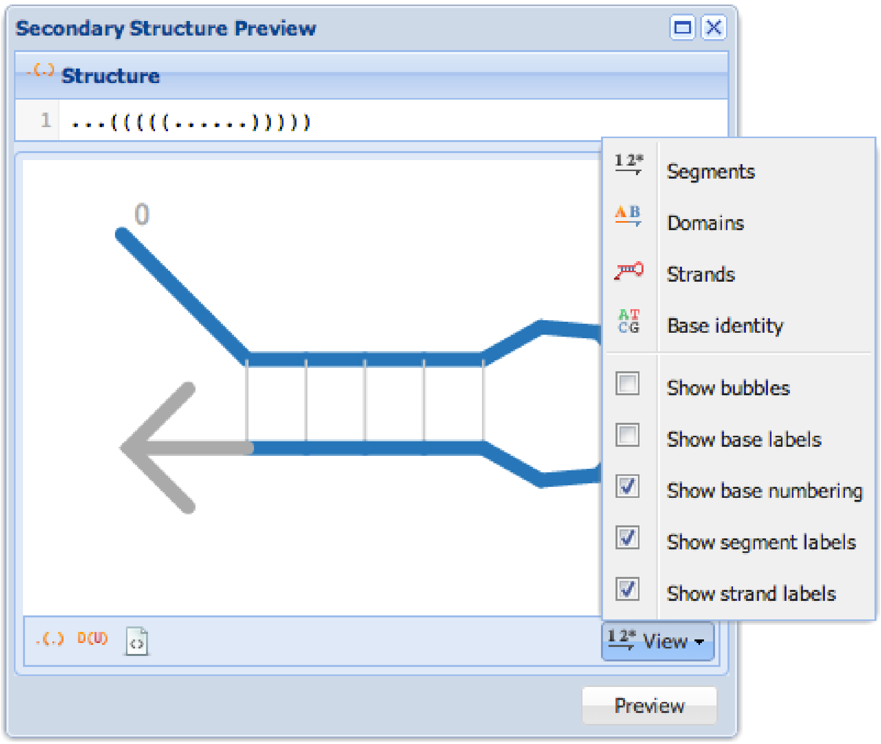


Figure 19.2: Secondary Structure “View” menu

- Strands – Color the base bubbles, letters, or backbones based on the strand to which they belong
- Base identity – Color the base bubbles, letters, or backbones based on the identity of each nucleotide base

Depending on the additional options (below), different features will be shown and colored:

- Show bubbles – tick to show each base as a colored circle, or "bubble" (with coloring determined by the mode)
- Show base labels – tick to show a letter indicating the base identity of each base.
- Show base numbering – tick to show numerical indices every 10 bases
- Show segment labels – tick to show colored labels indicating the location of each segment; these can help you see where segments are, even when the coloring mode is not set to "segments"
- Show strand labels – tick to show labels for each strand

If "Show bubbles" is checked, the base bubbles will be colored. If "Show bubbles" is unchecked, but "Show base labels" is checked, the base letters will be colored. If both are unchecked, the backbone will be colored.

19.2 Sequence Threading Tool

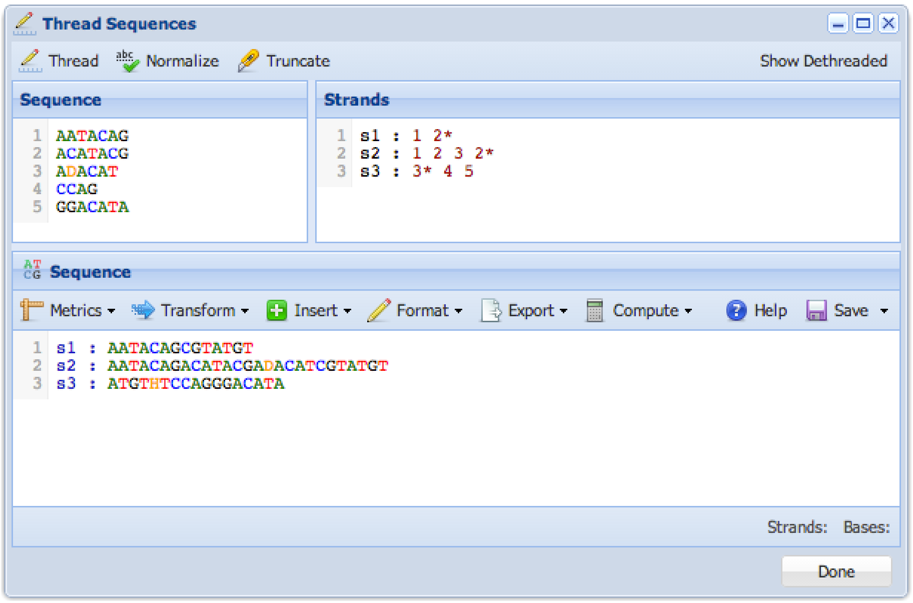
The sequence threading tool can be used to map a set of sequences on to a design. For instance, if you have the following sequences:

```
1 : AATACAG
2 : CCATAG
3 : GGAATAC
```

And you wish to arrange these segments into strands as follows:

```
s1 : 1 2 3
s2 : 3* 2* 1*
s3 : 1 2 3 2*
```

You could enter the sequences into the left pane, and the strand specifications (in the above format) into the right pane, then click "Thread" to view the full sequences below. Click "Show Dethreaded" to indicate the boundaries between segments in the final sequences shown below.



```
AATACAG
ACATACG
ADACAT
CCAG
GGACATA
```

```
s1 : 1 2*
s2 : 1 2 3 2*
s3 : 3* 4 5
```

```
s1 : AATACAGCGTATGT
s2 : AATACAGACATACGADACATCGTATGT
s3 : ATGTHTCCAGGGACATA
```

Figure 19.3: Seqeunce Threading Tool

19.3 Sequence Editor

Read more about the sequence editor.

Chapter 20

Customization

Workbench includes a documented Javascript API which you can use to extend the built-in capabilities, perform custom processing, or modify the default behavior of applications. There are two main ways to do this:

- Console - You can execute arbitrary Javascript within the context of the active Workbench Application using the console.
- Custom scripts - You can write and execute complete Javascript programs from within Workbench. This is most useful for complicated workflows which need to be automated.

20.1 Console

You can also run small snippets of code within a particular application, using the Workbench console.

To open the console:

- From the main menu, select 'Tools', then 'Console'.
- Enter your code in the console window which appears at the lower right.

20.2 Custom scripts

You can use a custom script to write and execute a Javascript program within Workbench; this allows you access to all of the programmatic tools that Workbench provides, while letting you automate complicated or repetitive behavior.

To build a custom script:

- Create a new Javascript file.
- Open the file and enter your script.
- Click ‘run’ to execute your script.

Chapter 21

Installation

Note: These instructions assume you're hosting your own server. If you're using a hosted installation, you don't need to do anything to install! If you're confused, see the server page.

1. Install VirtualBox. VirtualBox is a free "hypervisor"—a program that lets you run virtual machines—from Oracle.
2. Install Vagrant. Vagrant is a tool that lets you easily configure and run virtual machines
3. Download Workbench to a folder on your computer.
4. Open a terminal (such as Terminal on Mac OS X or `cmd.exe` on Windows), and change to the directory where you've downloaded Workbench
5. Run `vagrant up` to setup Workbench.

Re-building the server

If the server gets messed up somehow, you can rebuild it from scratch by running:

```
vagrant destroy
vagrant up
```

This will preserve all user files, but will *not* preserve any user account information.

Chapter 22

DyNAMiC Workbench Server

22.1 Overview

Workbench server is the part of the Workbench suite which is responsible for managing computationally intensive tasks, and storing files for Workbench users. There are two ways you might access a Workbench server:

- Hosted installation - Workbench server is intended to be installed on a cluster computing platform and made available via the web. In this case, you need only interact with the web-based Workbench client interface. You'll just need an invitation code from whoever runs the server, and you'll be able to create an account and begin using Workbench. The only hosted installation currently available is at provided by the Molecular Systems lab at Harvard.

- Local installation (hosting your own server) - For testing purposes, you may wish to host your own Workbench server. This requires a bit more effort, but you have full control over the entire system. See below for details about how to set up your own server.

Note: You only need to read this chapter if you're interested in setting up your own local installation. If you're accessing Workbench from a hosted installation (such as from [http://www.molecular-systems.net/workbench]), you can skip this section.

22.2 Server Tools

Workbench ships with several server tools installed. For details, see Server tools.

22.3 Hosting your own server

If you are reading this documentation, you've likely already obtained a copy of Workbench from the Molecular Systems Lab. Because of the number of external dependencies that the Workbench server has, and the relative difficulty in setting them up, Workbench server is deployed as a VirtualBox appliance, managed with Vagrant. VirtualBox is a free virtualization platform provided by Oracle, and Vagrant is a tool for easily configuring virtual machines. This means Workbench will run as a Virtual Machine, with its own isolated operating system, file system, process management, etc. You just need to install the virtual machine, and you'll have access to all of the relevant server tools (such as NUPACK, SpuriousDesign, the Nodal and Pepper compilers, etc.) without needing to configure them individually. This setup has the added benefit that if Workbench or one of its server tools crashes, it won't affect your host machine.

It's important to understand how this setup works: The Workbench server virtual machine will run (using VirtualBox) on your computer (which is called the "host" in this circumstance); it contains a separate operating system (the "guest" operating system, which in this case is a version of Ubuntu linux), and a lot of software, including a web server and the server tools. All of this software which will run within the virtual machine, sharing your processor and memory, but essentially isolated from your computer. There are two special communication channels between the virtual machine and the host:

- Shared folders: this VirtualBox feature allows folders on the host to be mirrored in the guest, and vice-versa. This lets you to access your files stored on Workbench from within your normal operating system file manager (e.g. Finder, Nautilus, Windows Explorer).
- Host-to-guest network: this creates a special network only between the host and guest. This means that the virtual machine will not be visible to the internet at large, but it will be able to connect only to the host (for instance to expose the Workbench web server).

22.3.1 Running the server

As part of the installation process, you'll install a copy of the pre-packaged Workbench Server virtual machine on your computer. You'll be able to launch the virtual machine (VM) directly from the command line; simply change to the directory where Workbench is installed, then run

```
vagrant up
```

This will install and setup the server (if not done already), then boot the server. If you've previously halted the server, you can also restart it with `vagrant up`.

The actual server component is configured to launch automatically when the appliance starts. That means if you just use the Workbench interface, all you need to do is launch the appliance and point your browser

However, if you want to tweak the server beyond what's described in the customization page, or to use any of the installed server tools directly (from the command line), you'll need to log in.

22.3.2 Logging in to the server

To log in to the server, you use a serparate set of user credentials (different from the username and password that you use to log into your host machine, or that you use to log in to the Workbench client interface on the web). These credentials are preset when you download Workbench, although you're encouraged to change them.

There are two pre-defined user accounts:

- `vagrant` (password: '`vagrant`') – this is a privileged account which can execute commands with `sudo`
- `webserver-user` (password: " '; a single space) – this is an unprivileged account which is used to run the server process.

(single quotes are not part of the username or password; the password is a singe space: `' '`).

The recommended method for logging in to the virtual machine is via SSH. This will allow you command-line access to the server.

To connect to the server via SSH:

- On Mac OS X or Linux, open a Terminal, and enter the following command:
  - For `webserver-user`: `ssh webserver-user@192.168.56.10`. You will be prompted to enter `webserver-user@192.168.56.10's password:`; enter the password.
  - For `vagrant`: change to the directory where Workbench is installed, and enter `vagrant ssh`. You won't need to enter a password

- On Windows: you'll need to download an SSH client called `ssh.exe`, which must be in your `%PATH`; `git` comes with one, which you just need to add it to the `%PATH%` environment variable. You can also use a graphical SSH client such as PuTTY, but you'll need to configure it for use with Vagrant. Open your SSH client, and login using credentials like this:
  - For `webserver-user`:
    host: 192.168.56.10 port: 22 user: (see above) password: (see above)
  - For `vagrant`: use `vagrant ssh`.

22.3.3 Using the web interface

The server will start automatically after the VM has finished booting (your server should be running by the time you see a login prompt). To view the web interface, point your web browser to: [http://192.168.56.10:3000/].

See documentation for the web interface.

22.3.4 Interacting with the server via SSH

Once you've logged in to the server with SSH, if you're comfortable, you can play around with shell accesss to the server.

The actual server process is described in a shell script: `/home/webserver-user/startup`, which you can look at if you curious. This script starts the Node JS web server process, which does the heavy lifting of running the server.

`startup` is in turn controlled by an Upstart script, located in `/etc/init/workbench.conf`. The upstart script makes sure that the server gets launched on startup, killed on shutdown, and restarted if it crashes. You can control the server using Upstart commands:

- `sudo start workbench` – starts the server
- `sudo stop workbench` - stops the server
- `sudo status workbench` - tells you if the server is running or not

One other shell script is provided for your convenientce: ~: Occasionally, the database server doesn't shut down properly (this happens when the virtual machine is powered off without killing the server process). If when you launch the server normally and attempt to log in via the web interface, the login progress bar just keeps resetting, your database needs to be repaired; in that case, run:

```
sudo stop workbench
sh ~/repair
sudo start workbench
```

Note: `sudo` is required because administering Upstart processes requires administrator privileges. However, `startup` is run as `webserver-user` when launched via Upstart. `webserver-user` is *not* on the `sudoers` list.

22.3.5 Shutting down the server

To shut down the server and avoid damaging the database, simply shut down the virtual machine by:

- Entering `vagrant suspend` from the command line in your host machine (in the same directory as Workbench is installed); rather than shutting down the virtual machine, this will simply pause its execution.
- Entering `vagrant halt` from the command line in your host machine (in the same directory as Workbench is installed)

22.3.6 Managing users

You can manage users by visiting the /admin page from within Workbench; if your current account is an administrator, you will see a list of all users in the database. You can edit users' names, affiliations, and email addresses, you can activate or deactivate accounts, and you can make users administrators.

When you first install the Workbench server, however, the first account you make will not have administrator privileges and so you will have no other way of activating/managing user accounts. To remedy this, Workbench includes a simple command-line tool that you can use to manage user accounts. To access it:

- Sign into the server via `ssh` (e.g. using `vagrant ssh`)
- Navigate to `/home/webserver-user/app` (you must be in this folder)
- Run `meta/utils/users --help` to see a list of options and usage information.

For example, if you've made a user with the email address `example@example.com` and you'd like to make that user an administrator, you can run:

```
meta/utils/users edit example@example.com --admin
```

You can also do things like list all registered users, export users to a JSON file, import user data from a JSON file, and edit other properties of user (including resetting their passwords). Run `meta/utils/users --help` for a full list.

22.3.7 Troubleshooting

Can't access login page

When you navigate to Workbench in a browser and the login page does not load, this likely suggests the server is not running. Use the following steps:

1. Check that the VM is running by running `vagrant status`; if the result indicates the the `default` VM is `saved` or `halted`, then resume the VM using `vagrant up` and try again.

2. Check that the Workbench server is running; `ssh` into the VM (run `vagrant ssh`), then run `sudo status workbench`. If you see `workbench stop/waiting`, then the server is not running; try re-starting it using `sudo start workbench`.

3. If you instead see something like `workbench start/running, process 15445` (the number will be different), then the server *is* running. However, the server is configured to restart itself if its process crashes; this is generally rare, but an incorrect configuration may cause it to happen repeatedly, making the server unresponsive. You can check that this might be happening by running `sudo status workbench` twice in a row (e.g. `sudo status workbench; sudo status workbench`), and checking whether the process numbers are different. For instance, if you see

   ```
   sudo status workbench; sudo status workbench
   workbench start/running, process 15445
   workbench start/running, process 15447
   ```

   this suggests that Workbench is restarting repeatedly. You'll need to check the log files to see why this might be.

4. To access the log files, you can look in `home/webserver-user/logs` (from within the VM), or on your host machine in the `logs` folder (which will be a sister to the main Workbench install directory). There are two main log files:

   - `startup.log` – This file contains everything written to `stdout` and `stderr` by the Workbench startup script; look here if Workbench has crashed or is crashing repeatedly.

- `full.log` – This file contains application errors written by the Workbench application code; check here if Workbench is refusing to run a particular computational tool, load a file, etc.

5. If it's not clear how to resolve the issue from here, post an issue on GitHub.

Login progress bar goes forever

If you can get to the login page and enter credentials, but when you login the progress bar keeps resetting itself, this suggests the user database is not running properly. To troubleshoot:

1. Check if the database server is running; `ssh` into the VM (run `vagrant ssh`), then run `sudo status mongod`. You should see `mongod start/running, process 854` (the number may be different).
2. Try repairing the database; run `sudo /home/webserver-user/repair`, then restart Workbench (`sudo stop workbench; sudo start workbench`) and try logging in again.
3. Check the database log file in `/var/log/mongodb/mongod.log` (e.g. `less /var/log/mongodb/mongod.log`), and look for clues as to what may be going wrong.
4. If it's still not clear how to resolve the issue from here, post an issue on GitHub.

Chapter 23

Scripting

Coming soon.

23.1 Overview

23.2 Console

23.3 Scripting Applications

Chapter 24

Workbench Application Development

Coming soon.

24.1 Overview

24.2 Bundling

24.3 API Documentation