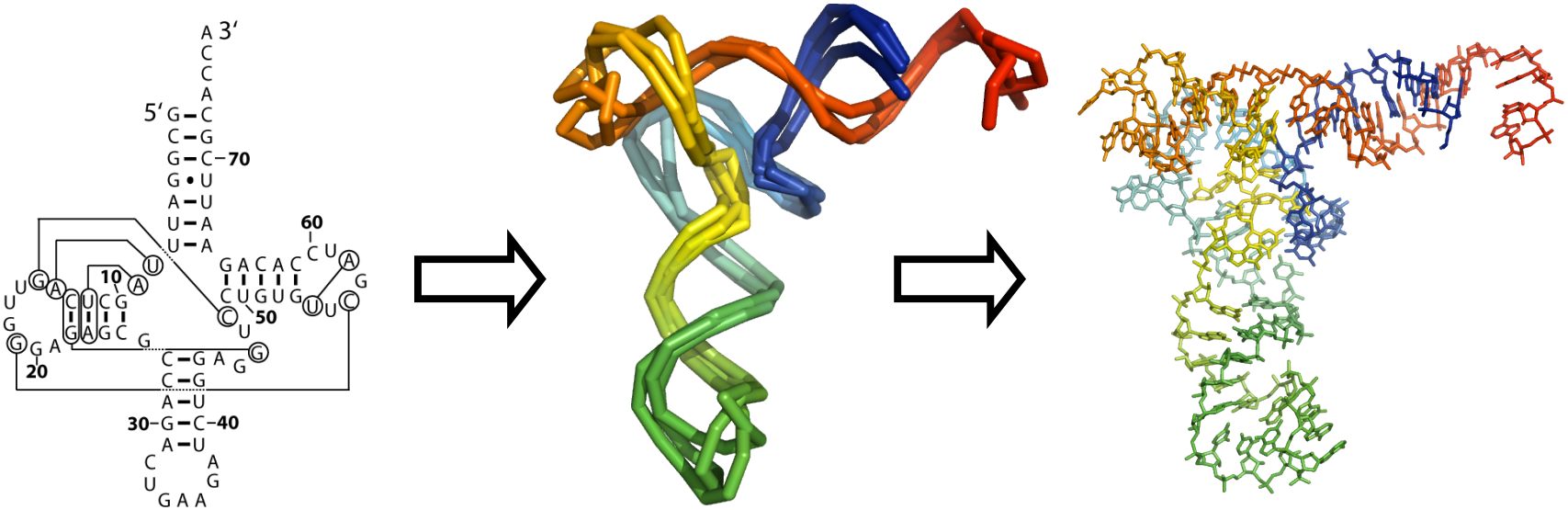
Using NAST and C2A to Model RNA 3D Structure

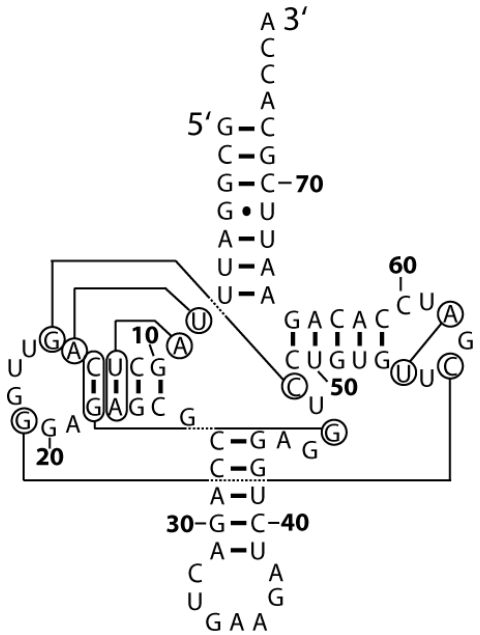


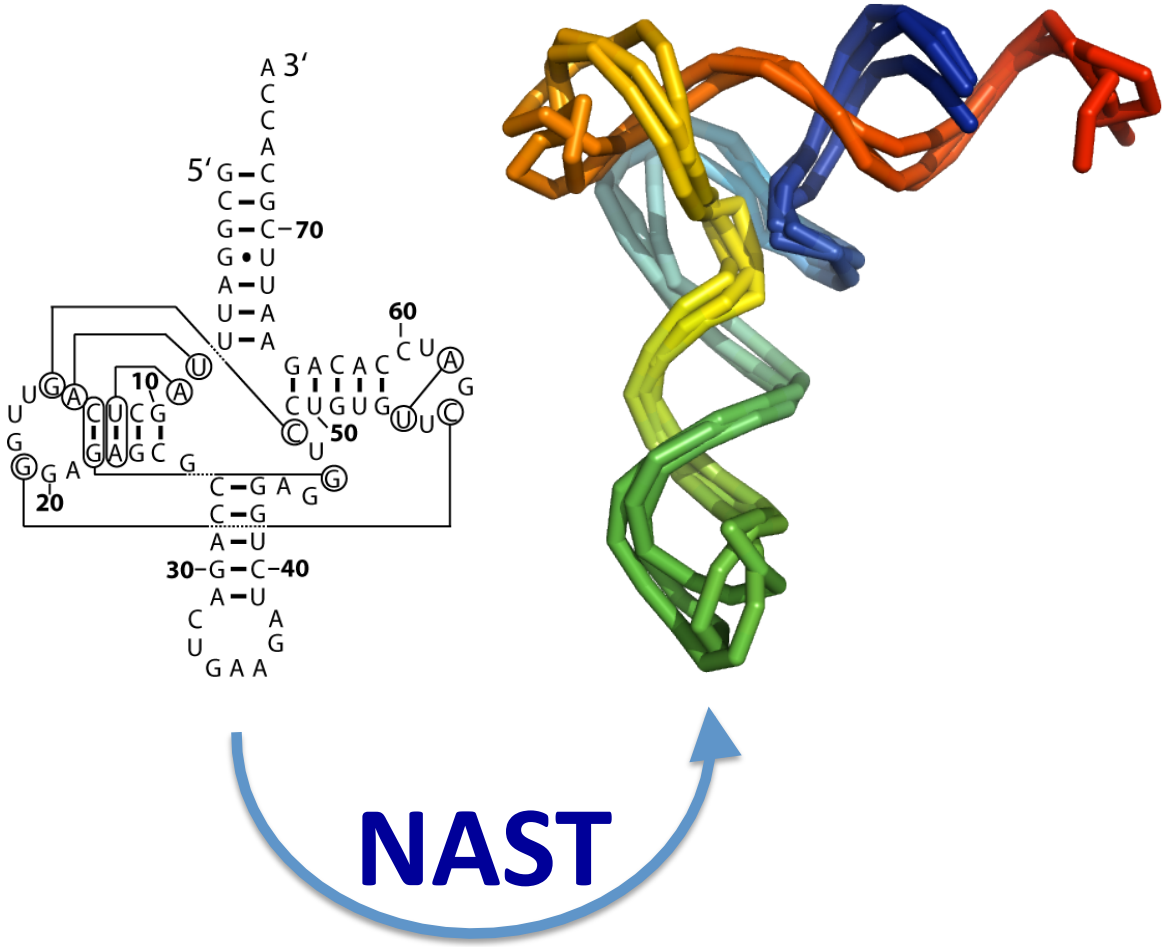
Magdalena A. Jonikas

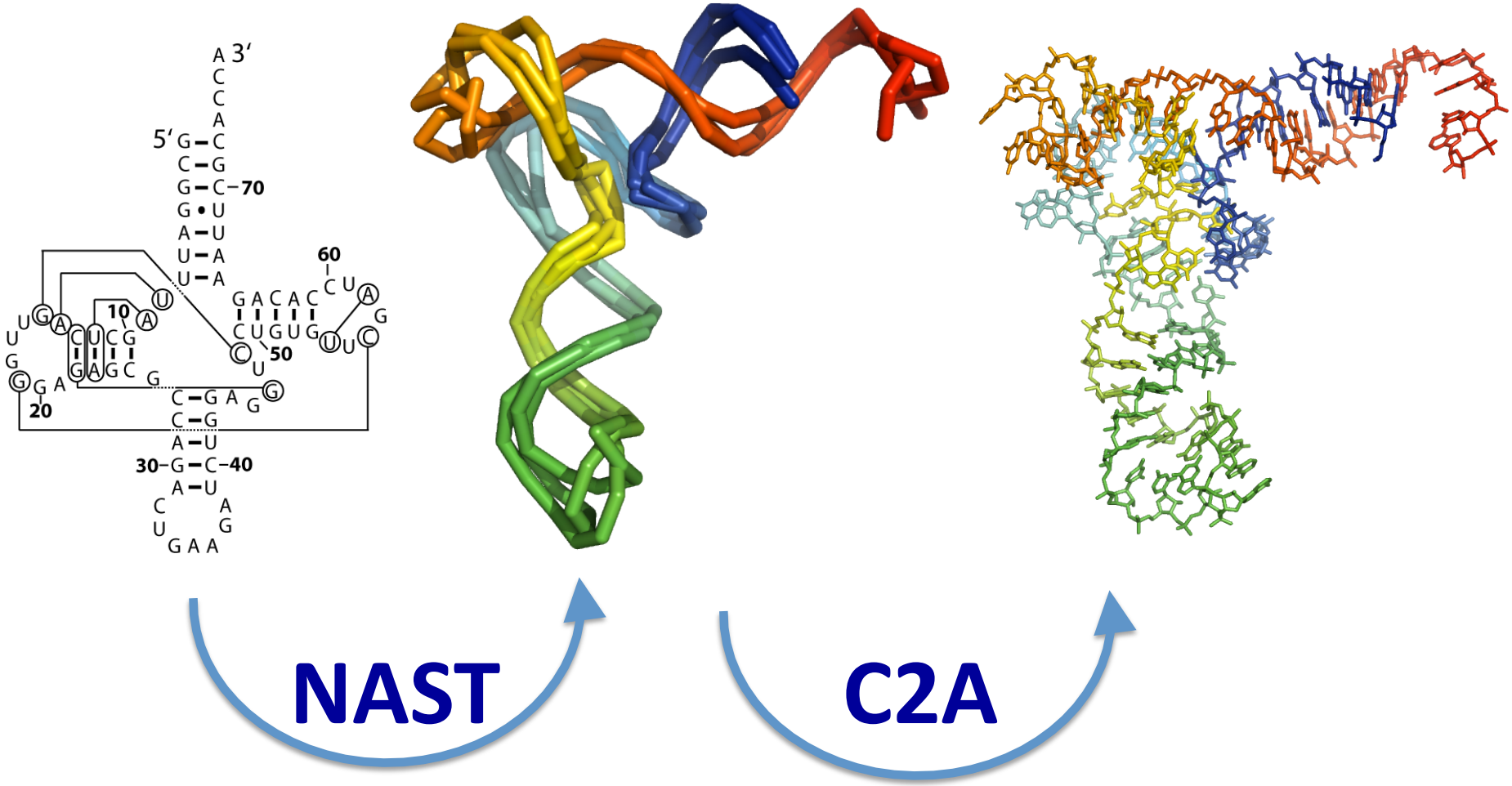
Stanford University

Department of Bioengineering

March 29th, 2010

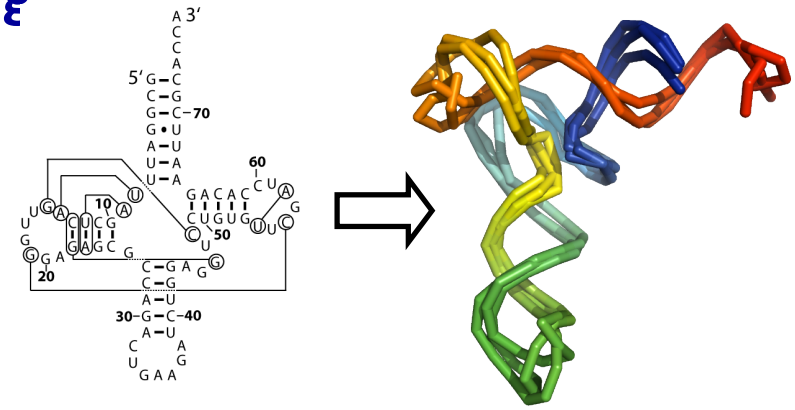

A 3'
C
C
A
5' G C
C G
G C 70
G • U
A U
U A
U A
60
C U
A
G
G A C A C
C U G U G
U U C
C
50
U
10
U
A
U
G
G
C U C G
G A G C G
G G A
20
C G A G
G
C G
A U
30 G C 40
A U
C A
U G
G A A


A 3′
C
C
A
5′ G–C
C–G
G–C–70
G • U
A–U
U–A
U–A
60
C U
G A C A C
C U G U G
50
U
10
C–G A G
20
C–G
C–G
A–U
30–G–C–40
A–U
C A
U G
G A A
NAST

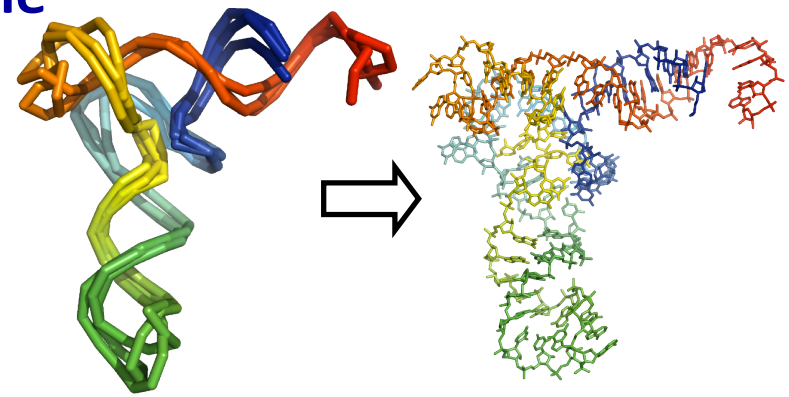

NAST
C2A

Outline

1. **Building coarse grain models using the Nucleic Acid Simulation Tool (NAST)**
   a) Theory
   b) Tutorial



2. **Adding full atomic detail using the Coarse to Atomic (C2A) tool**
   a) Theory
   b) Tutorial

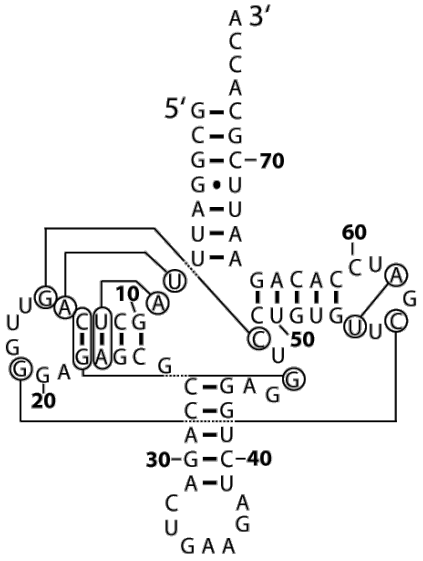


3. **Bringing it together: from primary sequence to full atomic structure**

Knowledge-Based Coarse-Grain Modeling of RNA Structure
NAST Overview

Input

Primary sequence
Secondary structure
Tertiary contacts



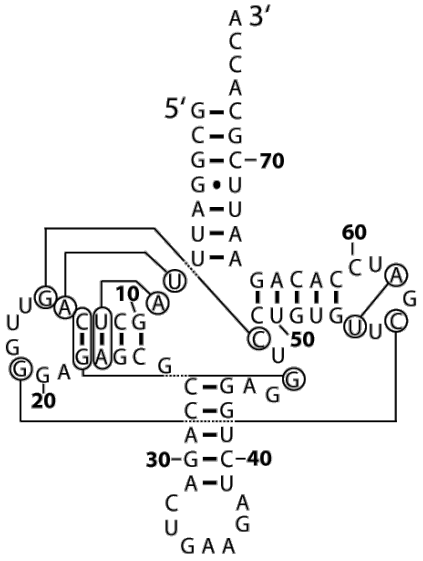
The Nucleic Acid Simulation Tool (NAST)

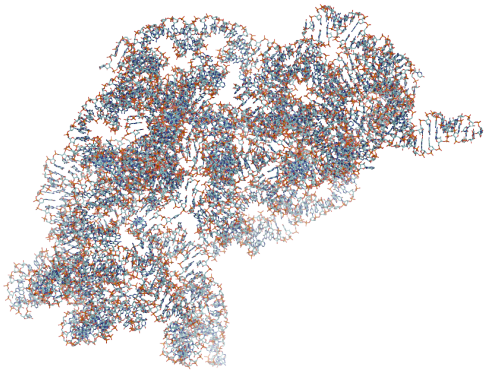
Jonikas et al., RNA 2009

Knowledge-Based Coarse-Grain Modeling of RNA Structure
NAST Overview

Input

Primary sequence
Secondary structure
Tertiary contacts





Geometry statistics

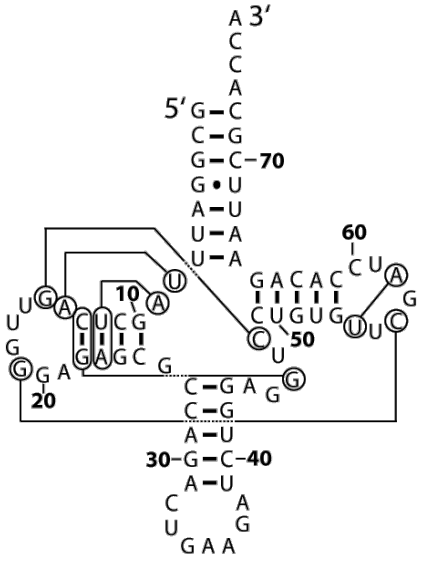
The Nucleic Acid Simulation Tool (NAST)

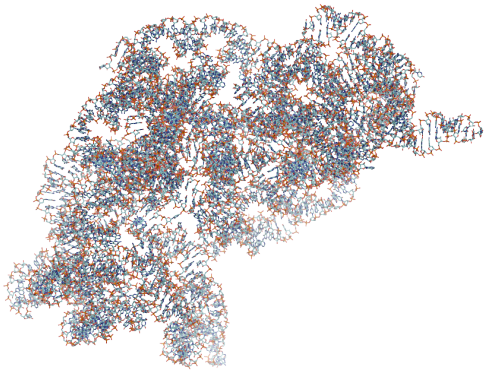
Jonikas et al., RNA 2009

Knowledge-Based Coarse-Grain Modeling of RNA Structure
NAST Overview

Input

Primary sequence
Secondary structure
Tertiary contacts



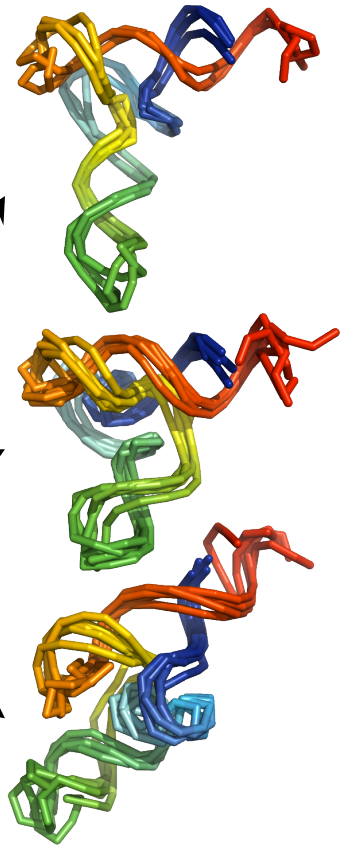


Geometry statistics

The Nucleic Acid Simulation Tool (NAST)

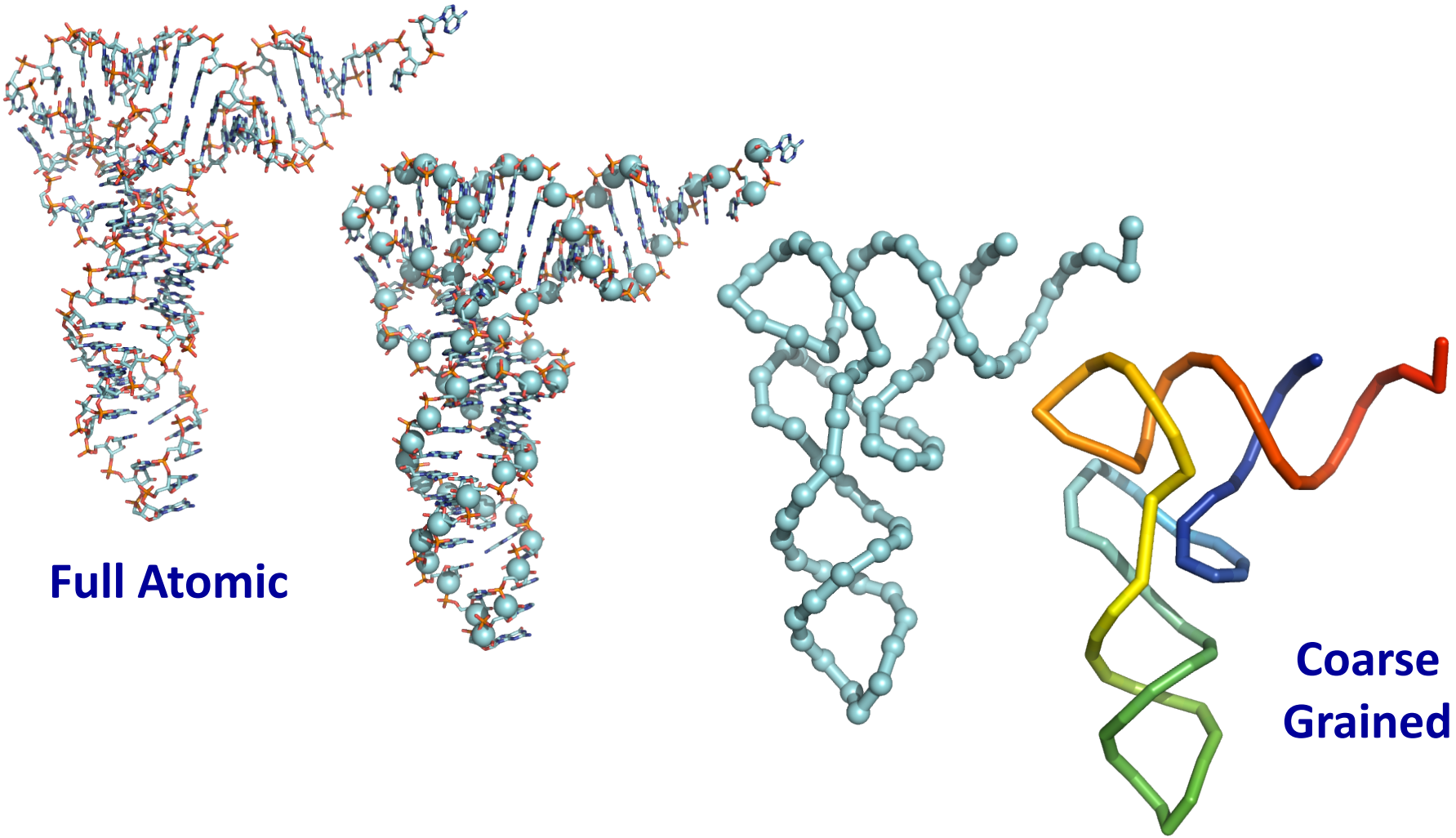
Output

Coarse grain models

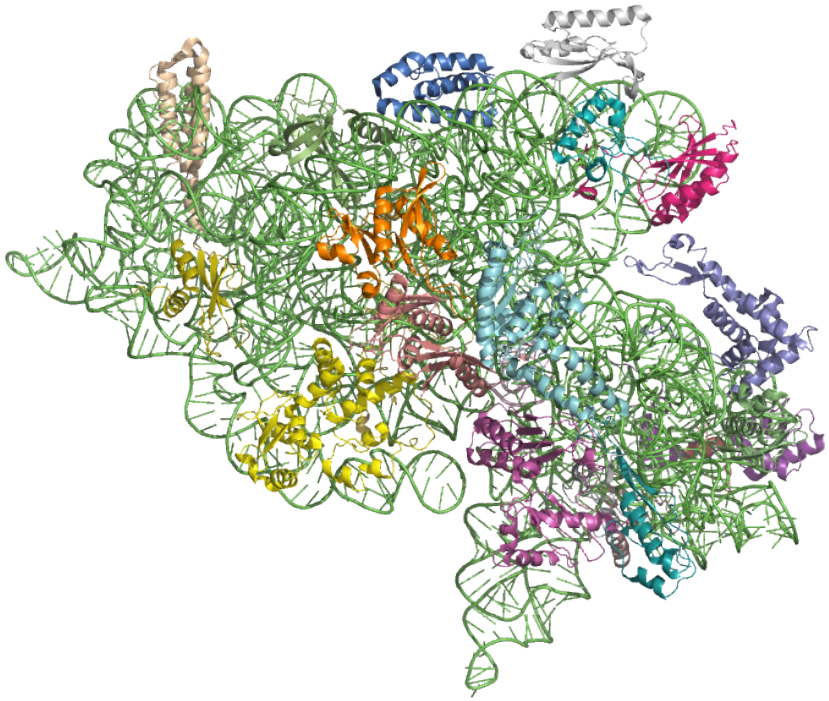


Jonikas et al., RNA 2009

Coarse-grained = one point per residue



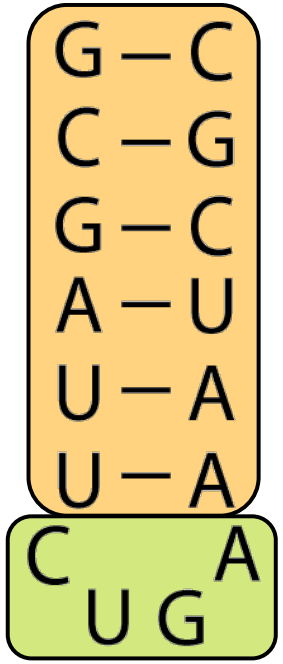
Observed geometry



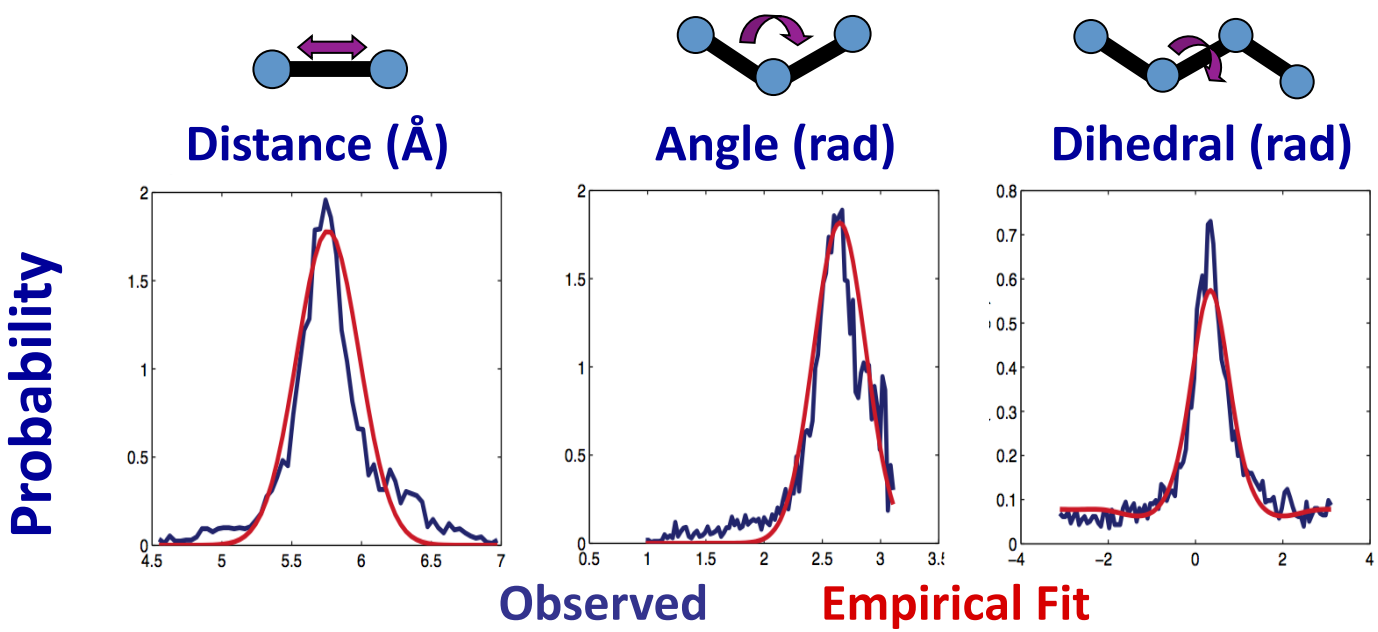
Thermus thermophilus
30S ribosomal subunit
(PDB ID1N32)
Ogle et al., 2002 Cell

Helical regions

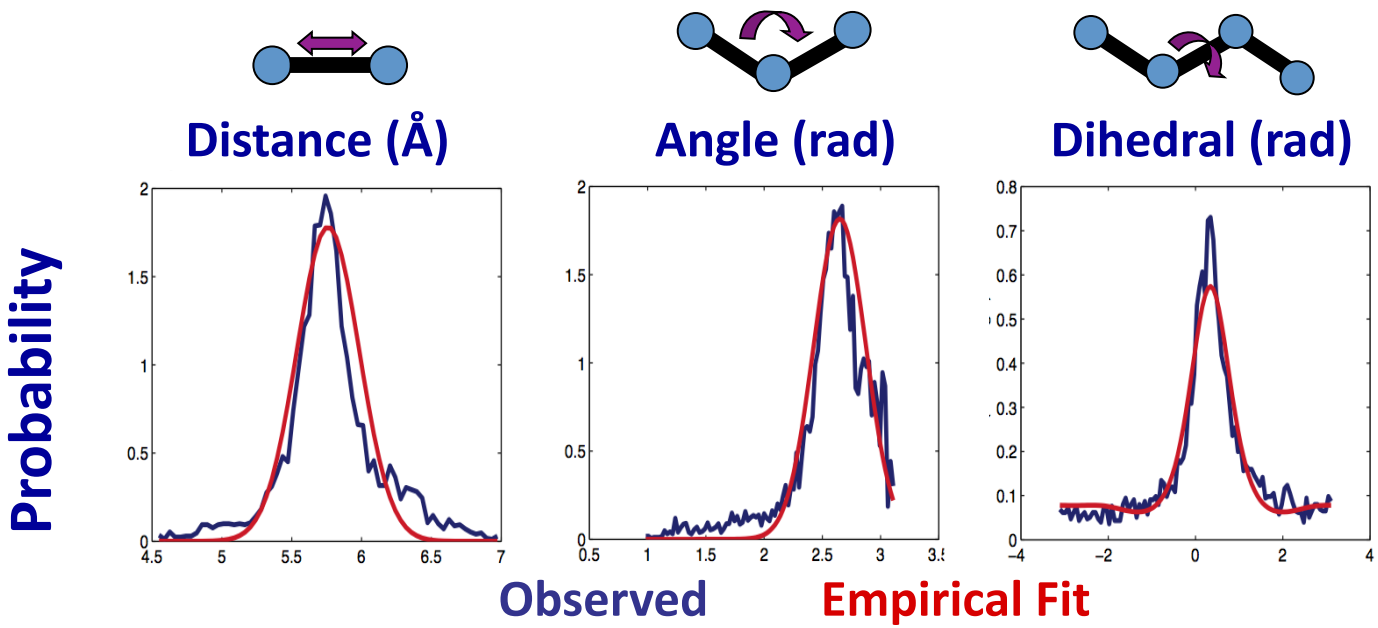
Non-helical regions



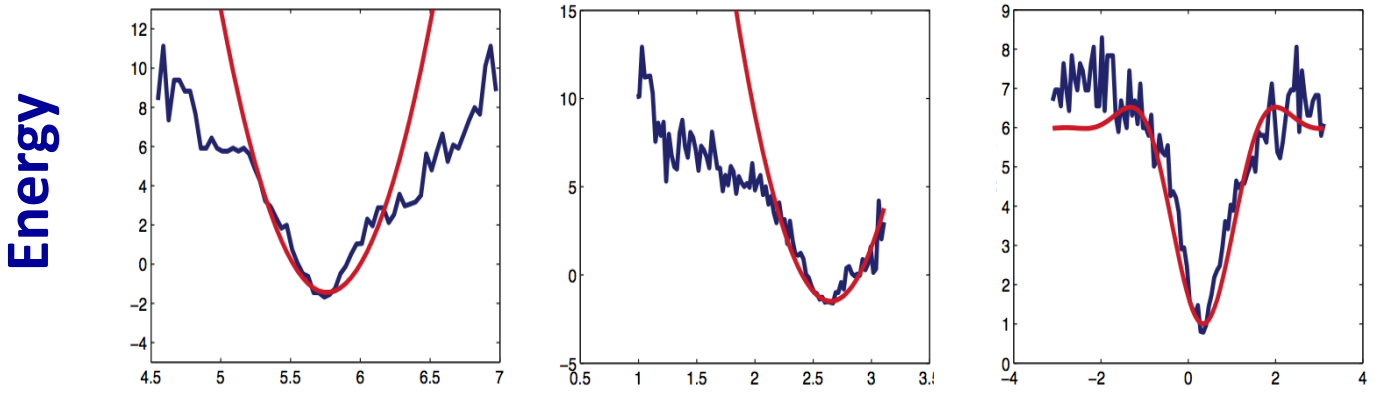
Non-helical geometry



Non-helical geometry

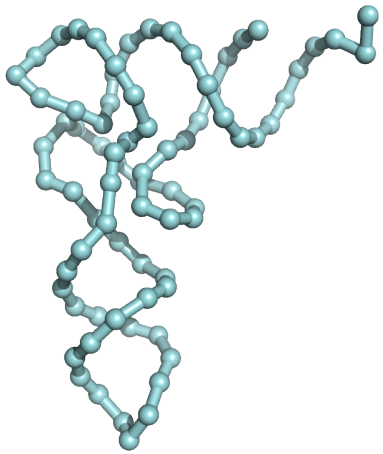


$$E(x) = -RT \cdot \ln[P(x)]$$



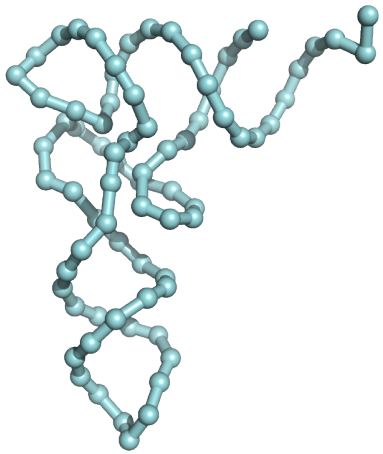
NAST knowledge-based energy function

$$
\begin{aligned}
\text{Energy} = \quad & k_b (R-R_0)^2 \\
+ \quad & k_a (\Theta-\Theta_0)^2 \\
+ \quad & k_1 \cos(\Theta-\Theta_0)+k_3 \cos[3(\Theta-\Theta_0)] \\
+ \quad & k_H (R-R_0)^2 \\
+ \quad & \varepsilon \; 5(r_0/R)^{12}
\end{aligned}
$$

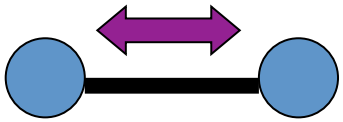


NAST knowledge-based energy function

$$\text{Energy} = \mathbf{k_b (R-R_0)^2} + k_a (\Theta-\Theta_0)^2 + k_1 \cos(\Theta-\Theta_0)+k_3 \cos[3(\Theta-\Theta_0)] + k_H (R-R_0)^2 + \varepsilon\ 5(r_0/R)^{12}$$

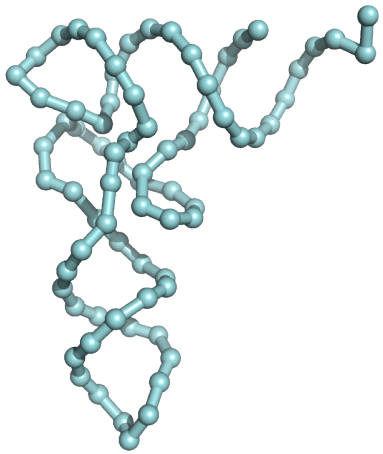


Bonds

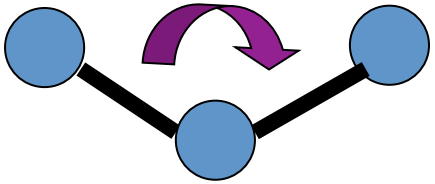


NAST knowledge-based energy function

$$\text{Energy} = k_b (R-R_0)^2 + \mathbf{k_a (\Theta-\Theta_0)^2} + k_1 \cos(\Theta-\Theta_0)+k_3 \cos[3(\Theta-\Theta_0)] + k_H (R-R_0)^2 + \varepsilon\ 5(r_0/R)^{12}$$

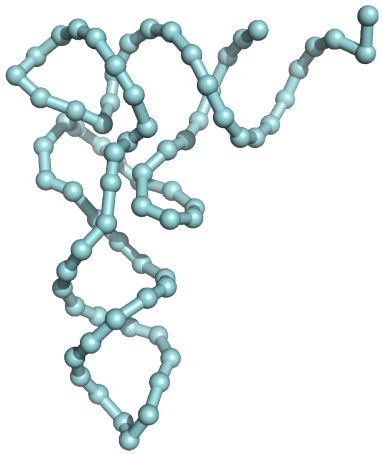


Angles

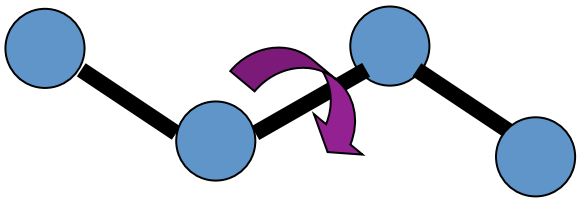


NAST knowledge-based energy function

$$\text{Energy} = k_b (R-R_0)^2 + k_a (\Theta-\Theta_0)^2 + \mathbf{k_1 \cos(\Theta-\Theta_0)+k_3 \cos[3(\Theta-\Theta_0)]} + k_H (R-R_0)^2 + \varepsilon\ 5(r_0/R)^{12}$$

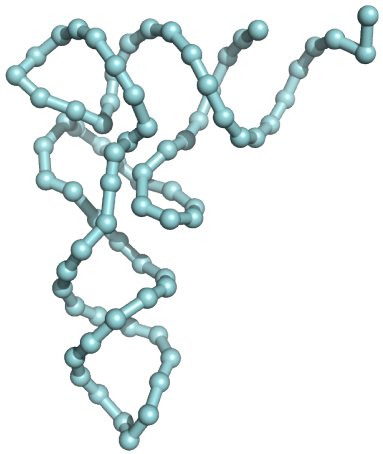


Dihedrals

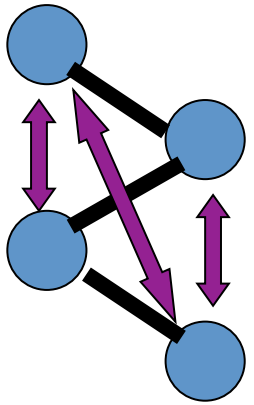


NAST knowledge-based energy function

$$\text{Energy} = k_b (R-R_0)^2 + k_a (\Theta-\Theta_0)^2 + k_1 \cos(\Theta-\Theta_0)+k_3 \cos[3(\Theta-\Theta_0)] + \mathbf{k_H (R-R_0)^2} + \varepsilon\ 5(r_0/R)^{12}$$

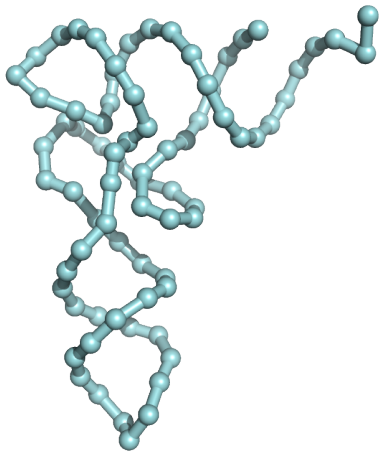


Harmonics

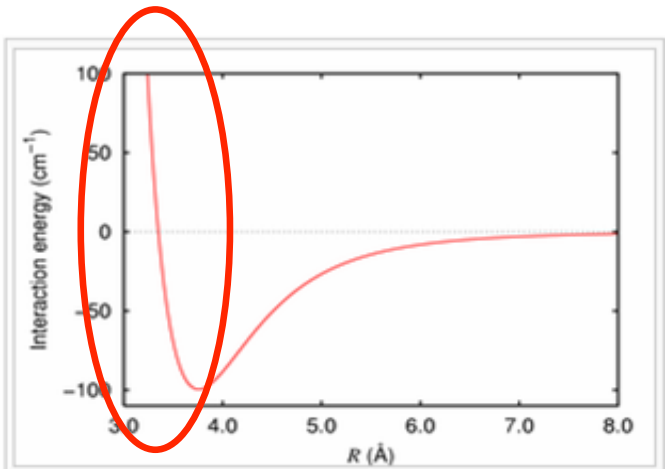


NAST knowledge-based energy function

$$\text{Energy} = k_b (R-R_0)^2 + k_a (\Theta-\Theta_0)^2 + k_1 \cos(\Theta-\Theta_0)+k_3 \cos[3(\Theta-\Theta_0)] + k_H (R-R_0)^2 + \varepsilon\ 5(r_0/R)^{12}$$



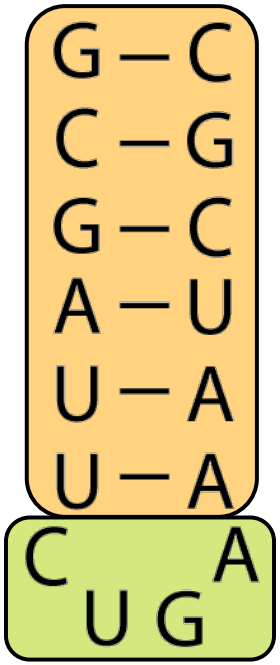
Repulsive van der Waals

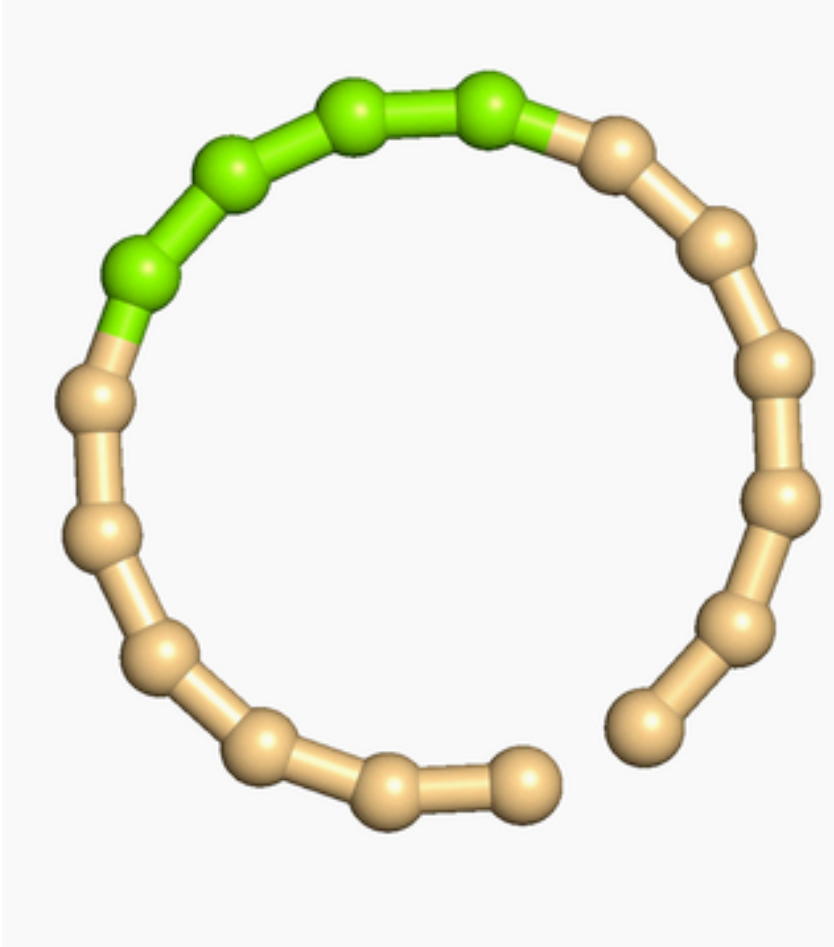


Simple helix

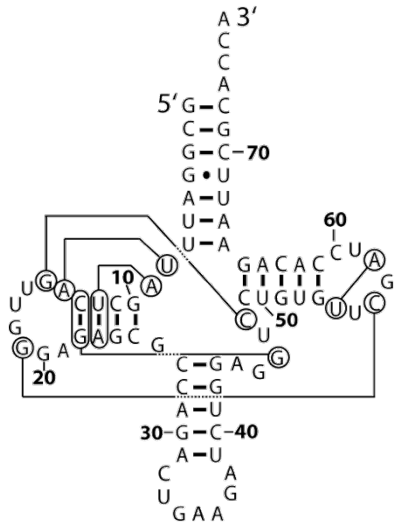
Helical regions

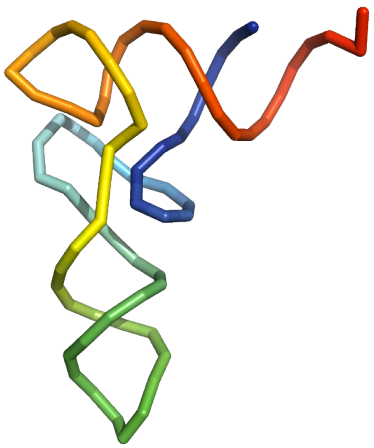
Non-helical regions

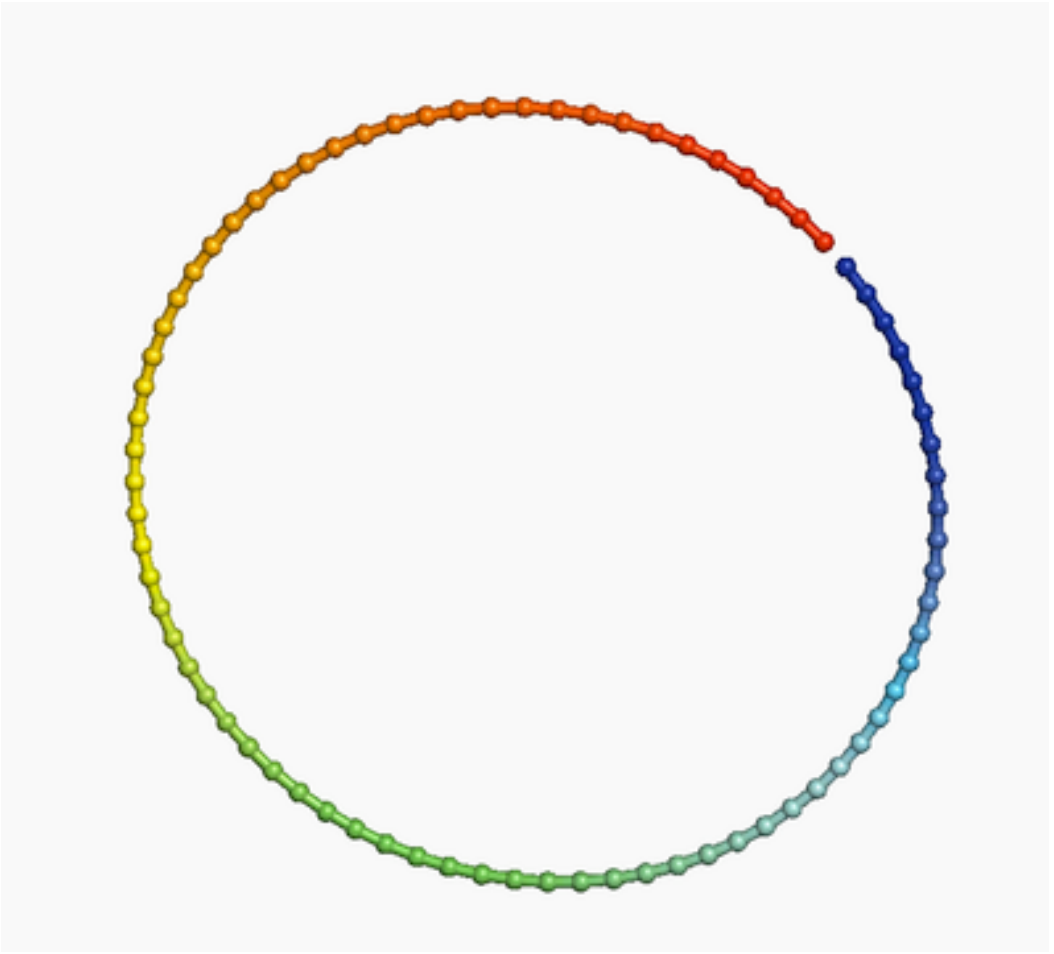




Modeling tRNA from primary sequence







Questions about theory before moving to tutorial?

Prerequisites

- NAST downloaded from www.simtk.org/home/nast
- NAST installed and test example run successfully
- Python 2.5 or 2.6

1. Open a command prompt or terminal window

(Windows) Start -> All Programs -> Accessories -> Command Prompt

(Mac OS) Macintosh HD -> Applications -> Utilities -> Terminal

2. Navigate to the nast examples directory

(Windows) `cd "c:\Documents and Settings\<user_name>\Desktop \nast-0.5.examples\nast"`

(Mac OS) `cd /Users/<user_name>/Desktop/NAST/nast-0.5/examples/nast`

Running the test example

3. Navigate to the 6TNA_MD example folder

(Windows) `cd 6TNA_MD`

(Mac OS) `cd 6TNA_MD`

4. Run the NAST test example

(Windows) `\Python26\python runTest.py`

(Mac OS) `python runTest.py`

What happens?

Let's visualize the output trajectory in VMD

Visualize the test run

5. Open VMD

6. Load the test trace file '6TNA_nast.pdb' in VMD

VMD Main -> File -> New Molecule

Molecule File Browser -> Browse

Chose a molecule file -> 6TNA_nast.pdb -> Open

Molecule File Browser -> Load

7. Connect the residues with a .psf file

Molecule File Browser -> Load files for: 0: 6TNA_nast.pdb -> Browse

Chose a molecule file -> 6TNA_nast.psf -> Open

Molecule File Browser -> Load

Simulate tRNA from unfolded

7. Return to your terminal

8. Copy runNast.py to myRunNast.py

(Windows) `copy runNast.py myRunNast.py`

(Mac OS) `cp runNast.py myRunNast.py`

9. Edit myRunNast.py

Do *not* use Microsoft Word

(Windows) **Start -> All Programs -> Accessories -> WordPad**

(Mac OS) **emacs, vi , TextEdit**

xterm

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Example script to run a Nast simulation for tRNA."""

pdbInFilename           = "6TNA_C3.pdb"
pdbOutFilename          = "6TNA_nast.pdb"
helixFilename           = "6TNA_helix.txt"
contactsFilename        = "6TNA_contacts.txt"
numSteps                = 100000
stepsPerReport          = 1000
defaultTemperature      = 300.0
randomSeed              = 1
useGpu                  = False
verbose                 = False

import simtk.nast.molecule as molecule
molecule.run(pdbInFilename=pdbInFilename,
             pdbOutFilename=pdbOutFilename,
             helixFilename=helixFilename,
             contactsFilename=contactsFilename,
             numSteps=numSteps,
             stepsPerReport=stepsPerReport,
             defaultTemperature=defaultTemperature,
             randomSeed=randomSeed,
             useGpu=useGpu,
             verbose=verbose)
```

Make some changes to the file

	OLD runNast.py	NEW myRunNast.py
pdbInFilename	= "6TNA_C3.pdb"	= "6TNA_C3.seq"
pdbOutFilename	= "6TNA_nast.pdb"	= "T1.pdb"
helixFilename	= "6TNA_helix.txt"	= "6TNA_helix.txt"
contactsFilename	= "6TNA_contacts.txt"	= "6TNA_contacts.txt"
numSteps	= 100000	= 100000
stepsPerReport	= 1000	= 1000
defaultTemperature	= 300.0	= 300.0
randomSeed	= 1	= None
useGpu	= False	= False
verbose	= False	= True

Run the new simulation

10. Run the new script

(Windows) `\Python26\python myRunNast.py`

(Mac OS) `python myRunNast.py`

11. Visualize trace T1.pdb in VMD

(Load T1.psf onto T1.pdb to connect the residues)

Things to notice:

-Started from unfolded circle.

-Secondary structure becomes constrained over the course of the simulation.

Run a longer simultion

12. Modify myRunNast.py with the following changes:

pdbOutFilename = T2.pdb

numSteps = 200000

13. Run the new myRunNast.py

14. Visualize the new trace T2.pdb

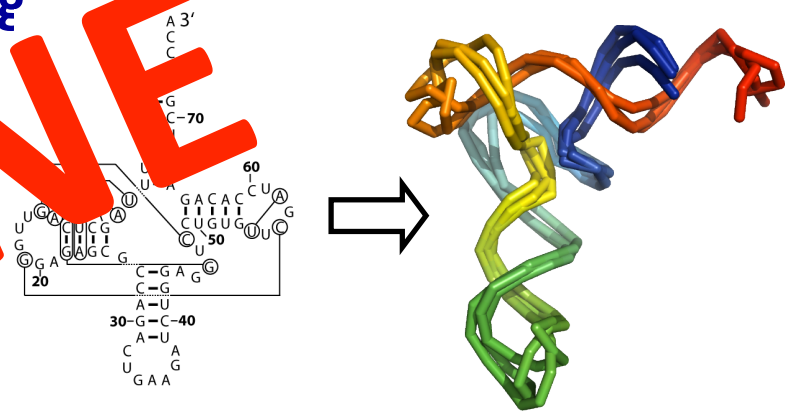
15. Save the last (or a favorite) frame as T2-last.pdb

We will use this structure later to add full atomic detail.

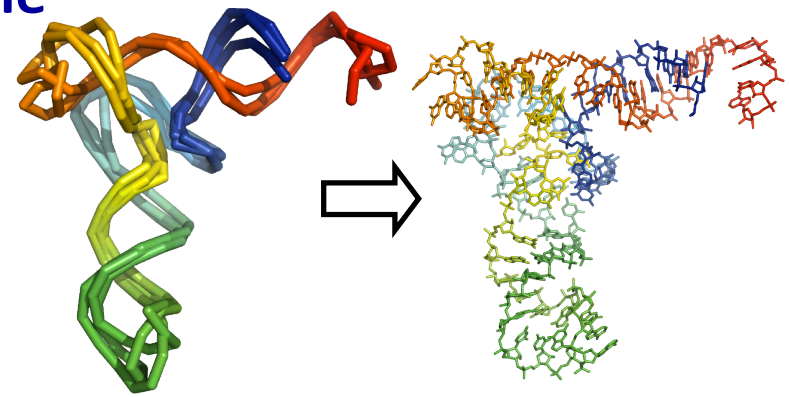
Outline

1. **Building coarse grain models using the Nucleic Acid Simulation Tool (NAST)**
   a) Theory
   b) Tutorial

DONE



2. **Adding full atomic detail using the Coarse to Atomic (C2A) tool**
   a) Theory
   b) Tutorial

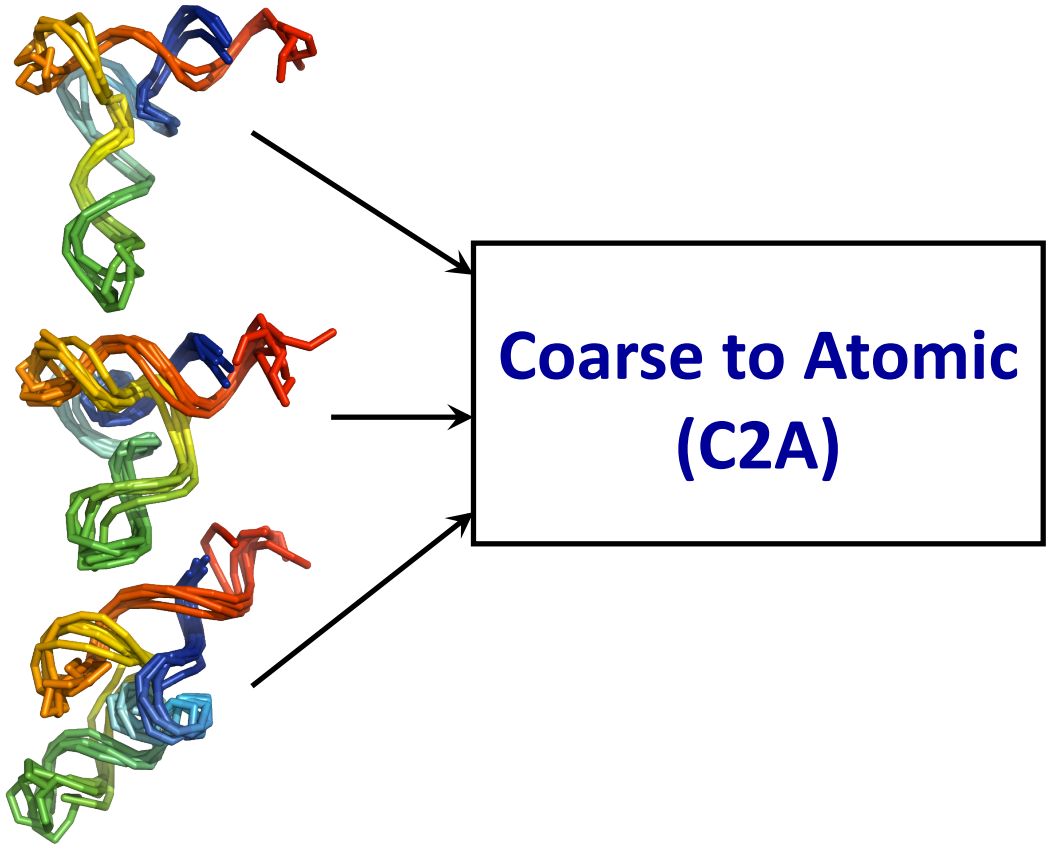


3. **Bringing it together: from primary sequence to full atomic structure**

Adding full atomic detail to coarse-grain models
Overview of C2A

Input

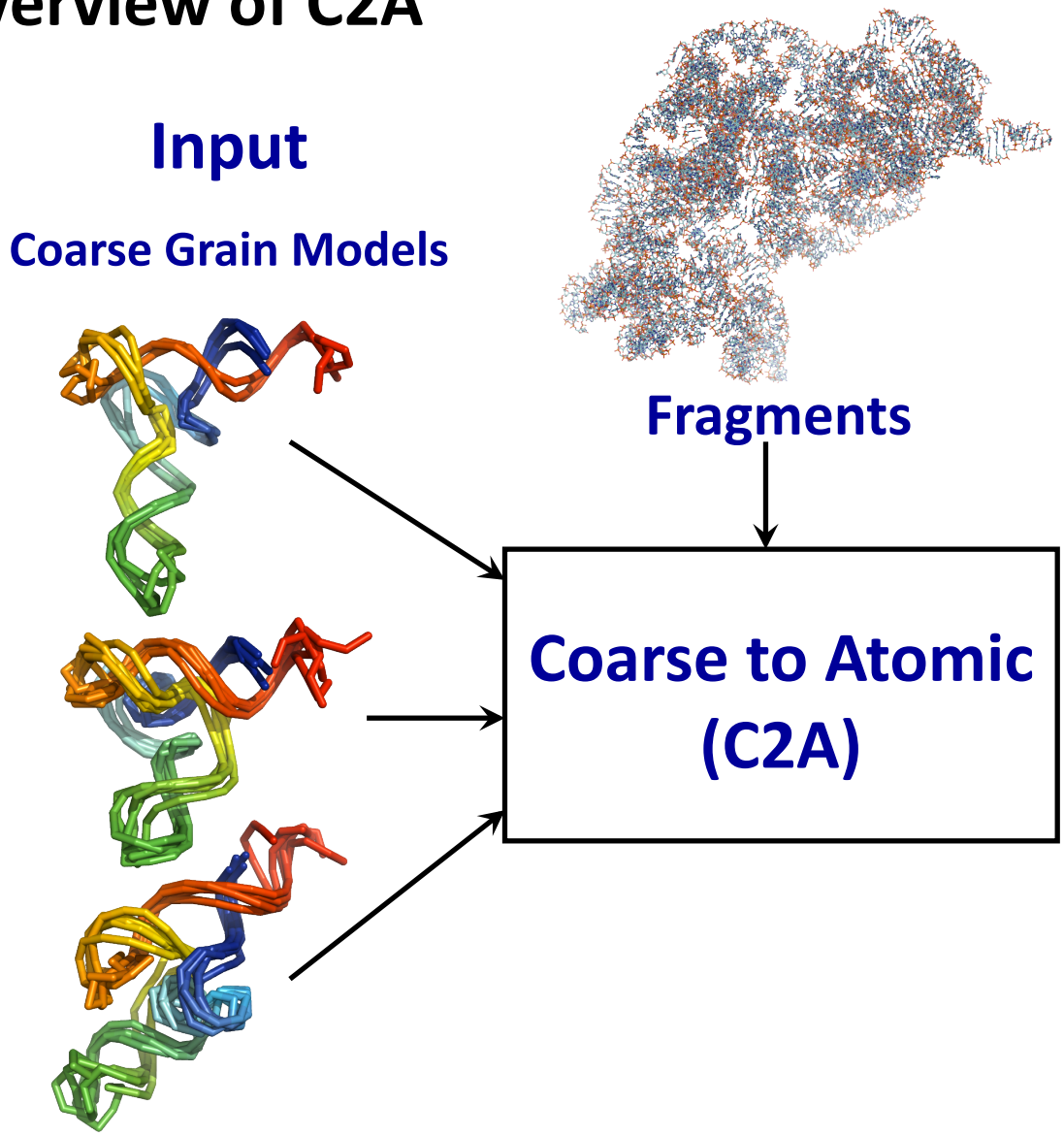
Coarse Grain Models



Jonikas et al., Bioinformatics 2009

Adding full atomic detail to coarse-grain models

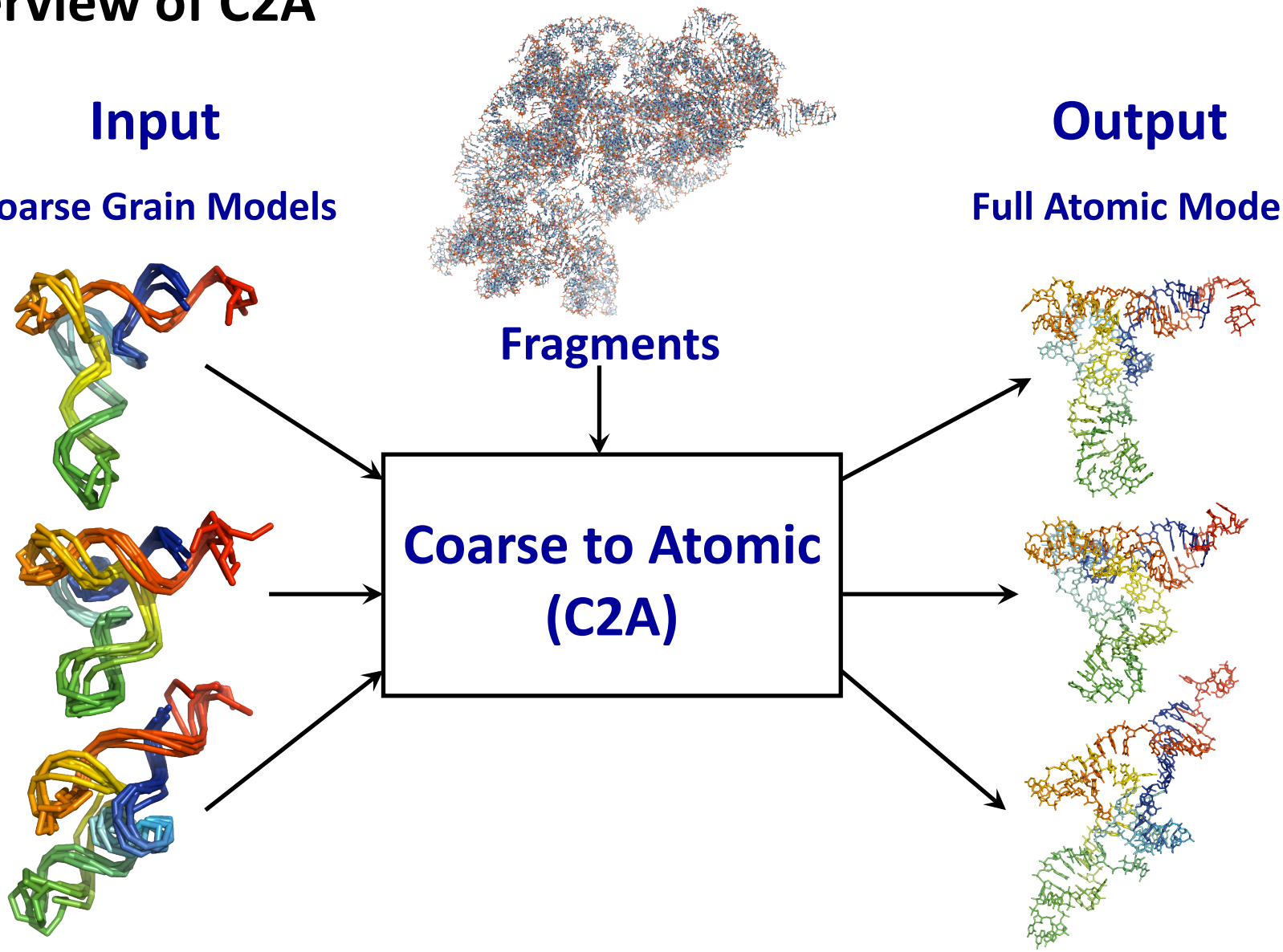
Overview of C2A



Jonikas et al., Bioinformatics 2009

Adding full atomic detail to coarse-grain models

Overview of C2A



Jonikas et al., Bioinformatics 2009

C2A is a two part process

Part 1: Searching a reference molecule for matching fragments and creating a working library of full atomic fragments.

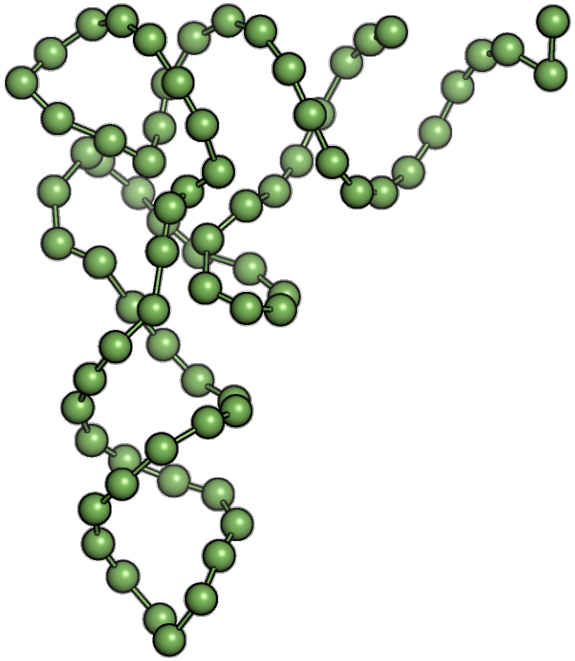
This is done once per template structure.

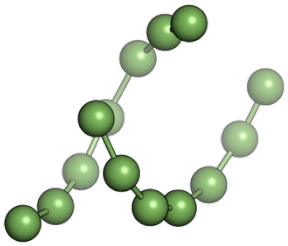
Part 2: Assembling fragments into a full atomic model.

Using the library of fragment matches generated in Part 1.

Part 2 should be run several times per template structure.

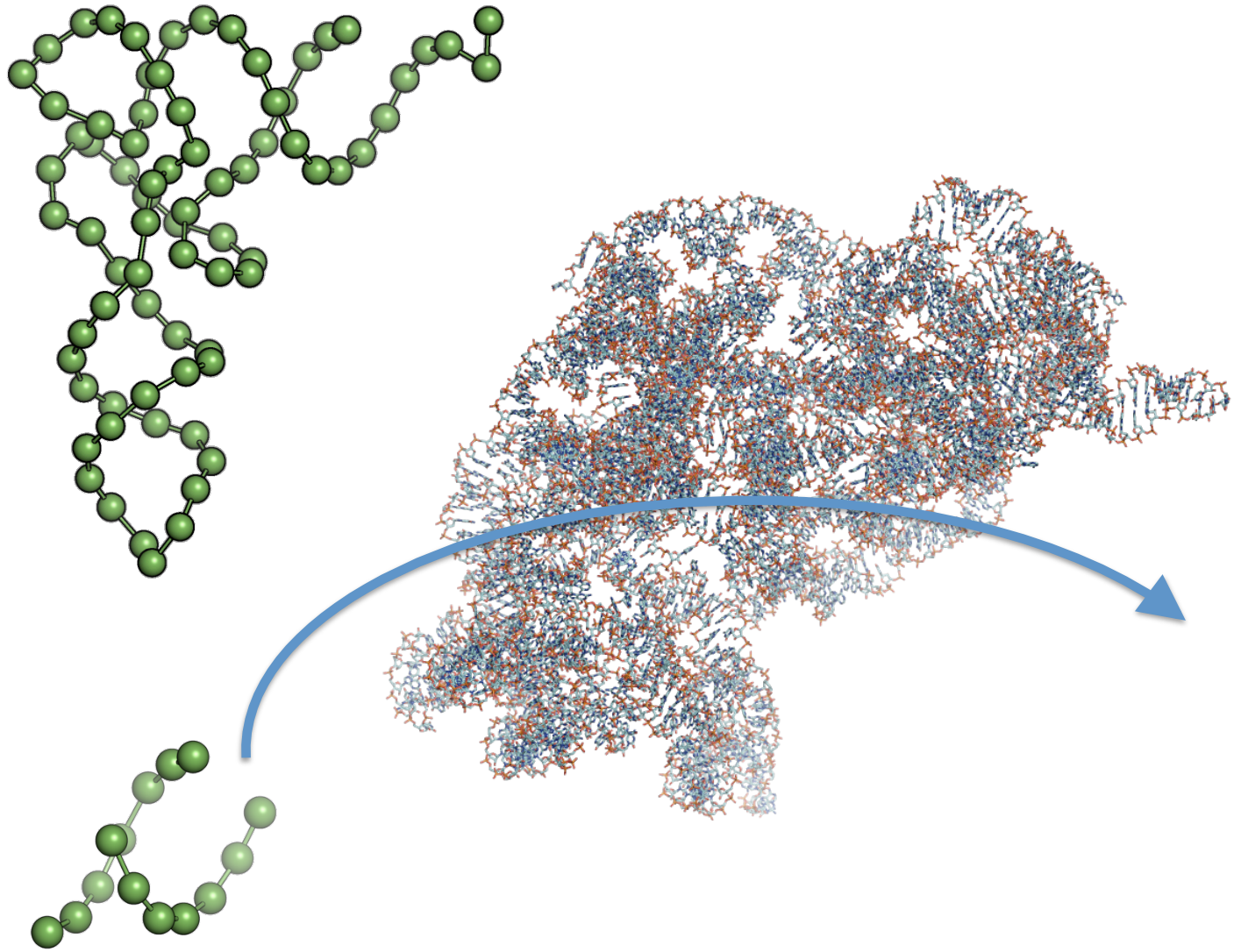
Part 1: generating a library of full atomic fragment matches





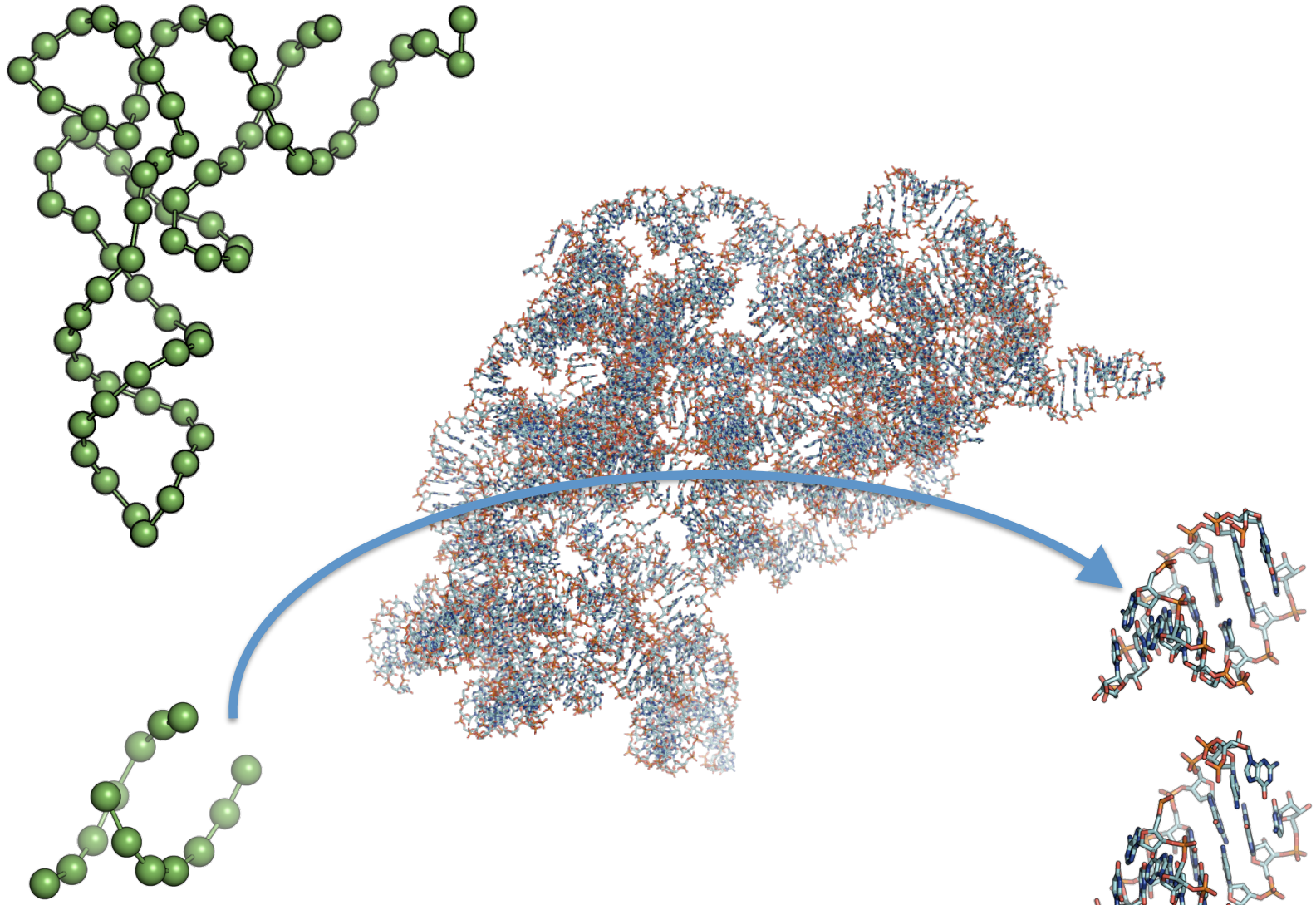
“fragment”

Part 1: generating a library of full atomic fragment matches



"fragment"

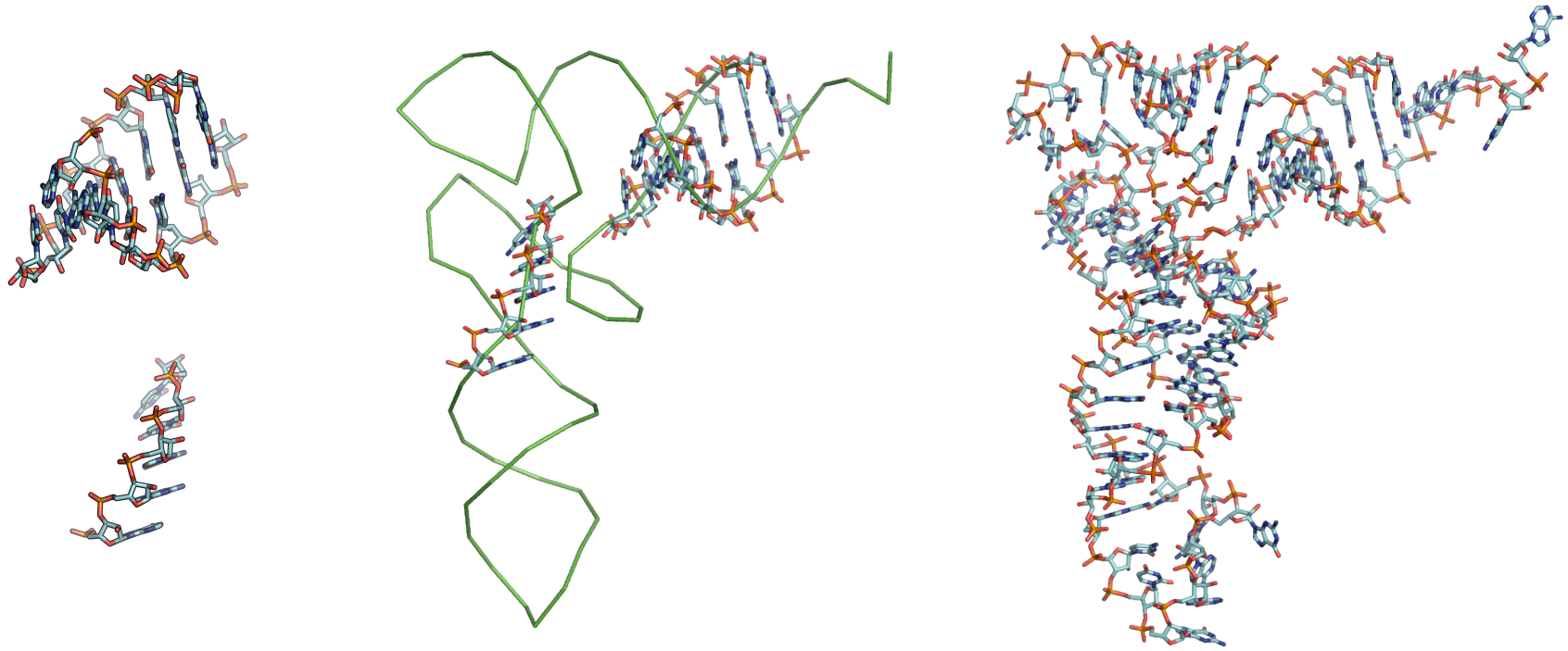
Part 1: generating a library of full atomic fragment matches



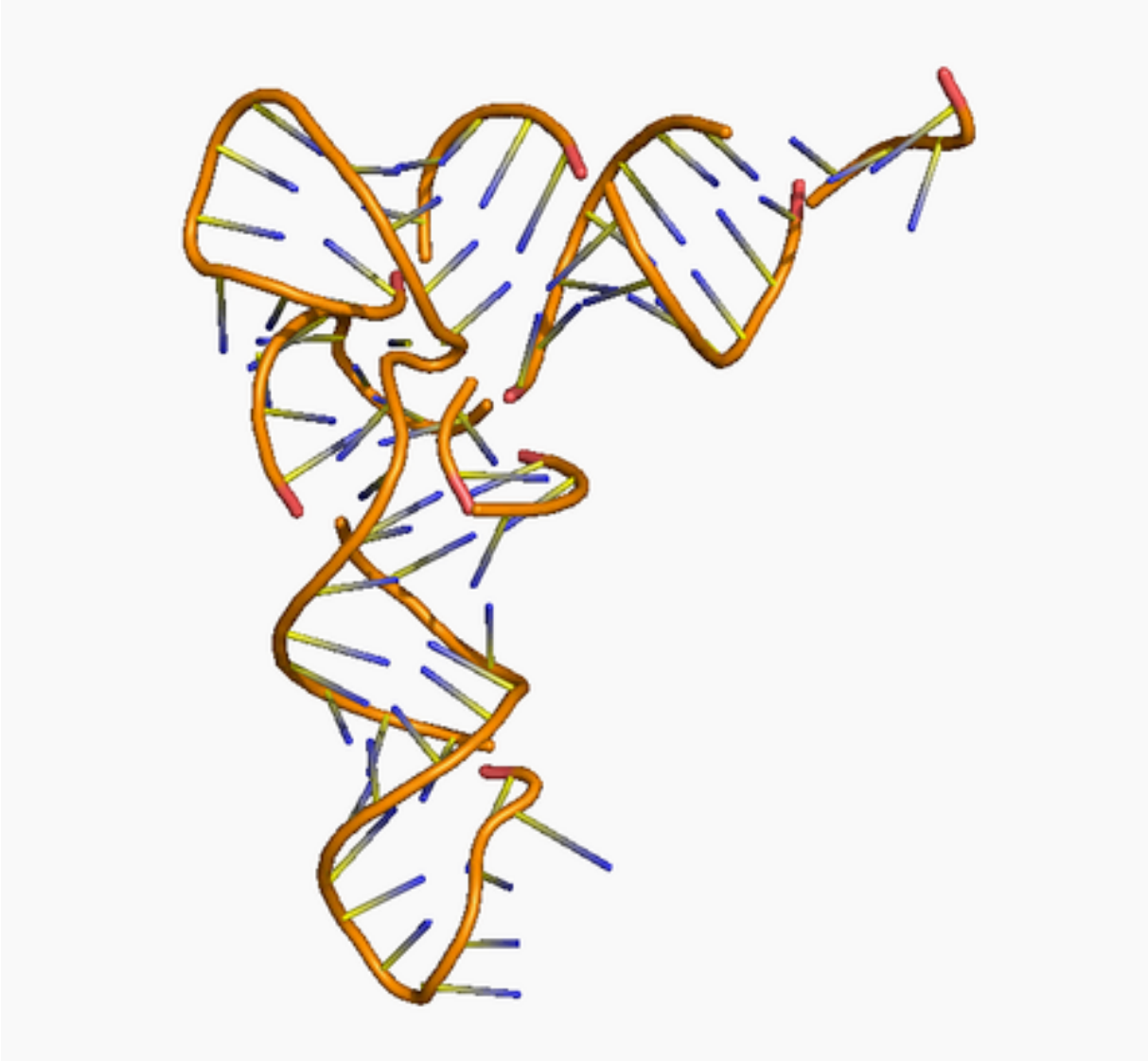
"fragment"

"matches"

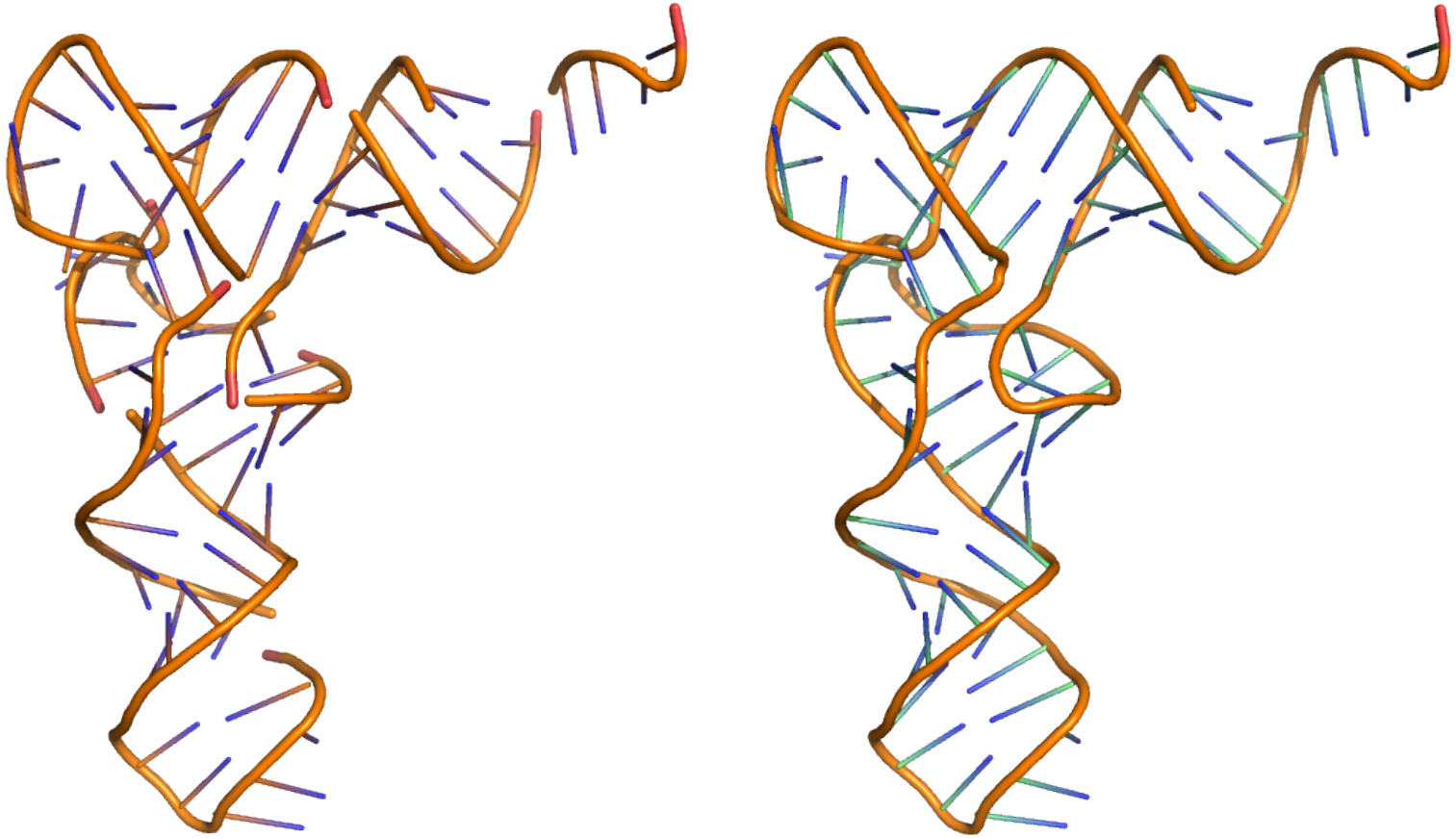
Part 2: Assembling a full atomic structure



Movie of structure assembly search



Reducing gaps with Zephyr



Questions about theory before moving to tutorial?

Part 1: Creating a library of full atomic fragments

1. Return to command prompt or terminal window

2. Navigate to the C2A example folder

(Windows) `cd ..\6TNA_c2a`

(Mac OS) `cd ../6TNA_c2a`

3. Let's look at the contents of file ex1.py

ex1.py

magda@magda:~/Projects/nast/examples/6TNA_c2a

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run first part of c2a calcs (find matches and build datab
ase)."""


import sys, time
import simtk.nast.c2a_findMatches as fm

t0 = time.time()
fm.findMatches(templateTrace = '6TNA-C3.pdb',
                   fragmentFile = '6TNA-FD.txt',
                   conversionData = open('baseConvert.pkl'),
                   searchStructureFile = '1N32-subA.pdb',
                   nhradius = 5.0,
                   hradius = 1.5
                  )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex1.py

magda@magda:~/Projects/nast/examples/6TNA_c2a

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run first part of c2a calcs (find matches and build datab
ase)."""


import sys, time
import simtk.nast.c2a_findMatches as fm

t0 = time.time()
fm.findMatches(templateTrace = '6TNA-C3.pdb',
                   fragmentFile = '6TNA-FD.txt',
                   conversionData = open('baseConvert.pkl'),
                   searchStructureFile = '1N32-subA.pdb',
                   nhradius = 5.0,
                   hradius = 1.5
                  )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex1.py

magda@magda:~/Projects/nast/examples/6TNA_c2a

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run first part of c2a calcs (find matches and build datab
ase)."""


import sys, time
import simtk.nast.c2a_findMatches as fm

t0 = time.time()
fm.findMatches(templateTrace = '6TNA-C3.pdb',
               fragmentFile = '6TNA-FD.txt',
               conversionData = open('baseConvert.pkl'),
               searchStructureFile = '1N32-subA.pdb',
               nhradius = 5.0,
               hradius = 1.5
              )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

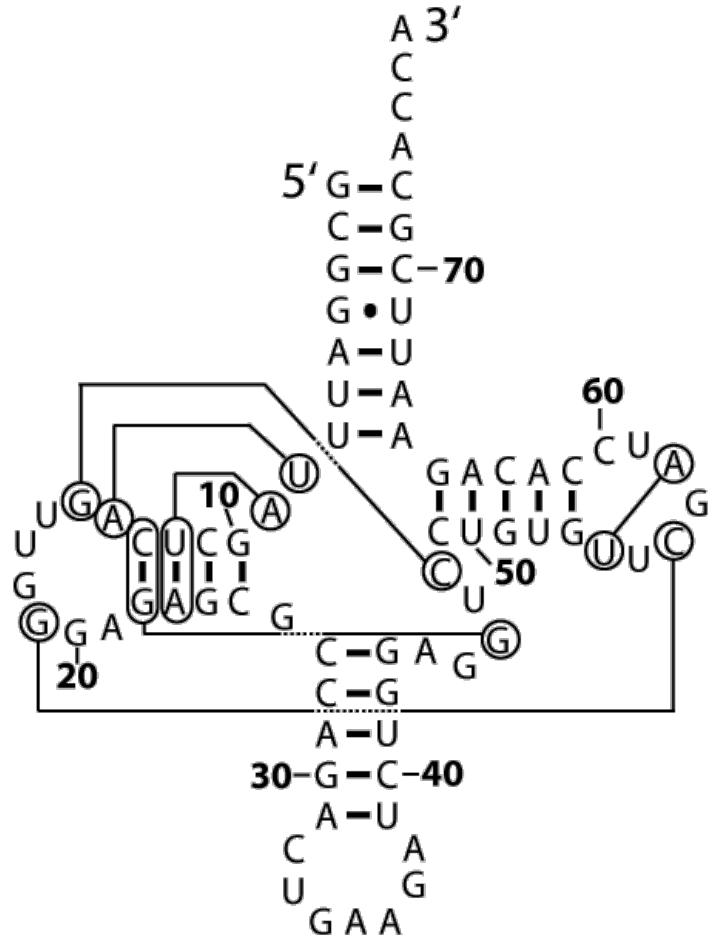
ex1.py

magda@magda:~/Projects/nast/examples/6TNA_c2a

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run first part of c2a calcs (find matches and build datab
ase)."""


import sys, time
import simtk.nast.c2a_findMatches as fm

t0 = time.time()
fm.findMatches(templateTrace = '6TNA-C3.pdb',
                   fragmentFile = '6TNA-FD.txt',
                   conversionData = open('baseConvert.pkl'),
                   searchStructureFile = '1N32-subA.pdb',
                   nhradius = 5.0,
                   hradius = 1.5
                  )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

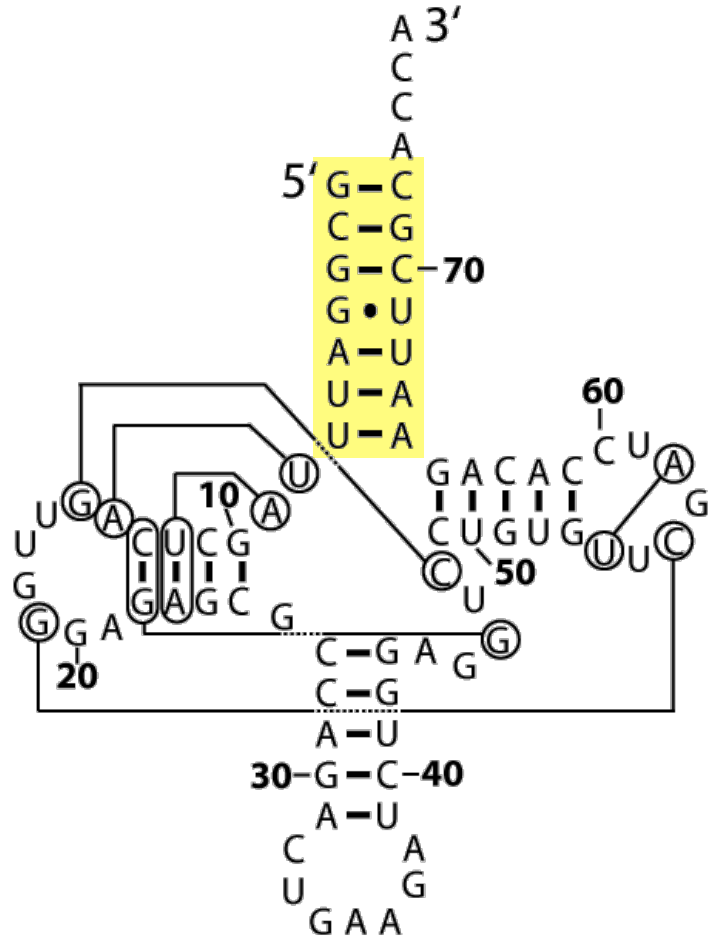
A quick look at fragment definitions

```
H1 1:7,66:72 10
H2 10:13,22:25 10
H3 27:31,39:43 10
H4 49:53,61:65 10
L1 H2 10
L2 H3 10
L3 H4 10
J1 H1:5,H2:5 10
J2 H2:3,H3:5 10
J3 H3:3,H4:5 10
E1 H1:3+4 10
```



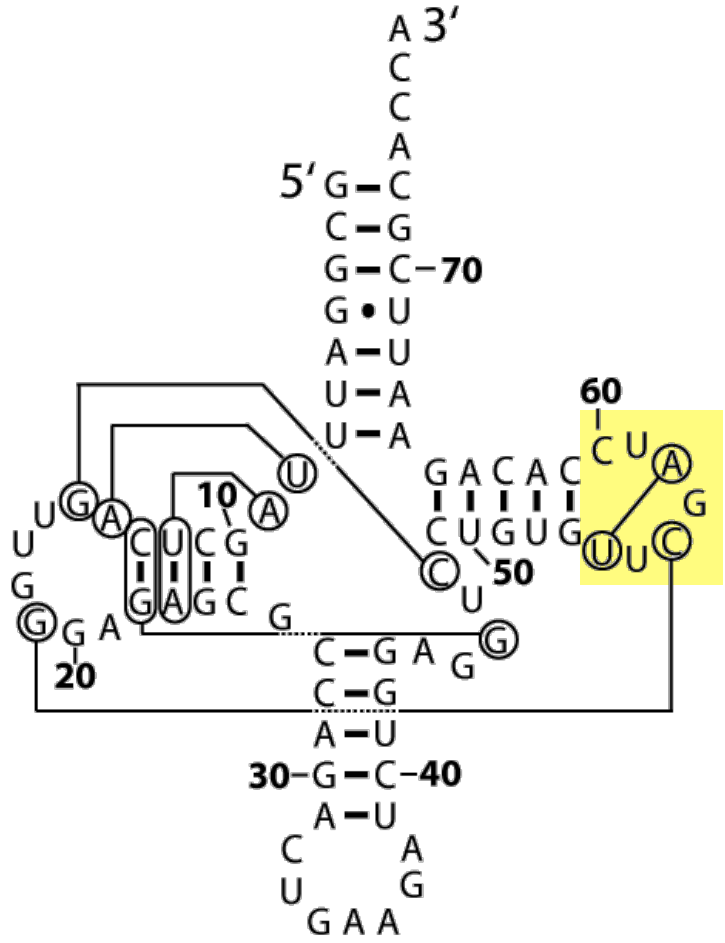
A quick look at fragment definitions

```
H1 1:7,66:72 10
H2 10:13,22:25 10
H3 27:31,39:43 10
H4 49:53,61:65 10
L1 H2 10
L2 H3 10
L3 H4 10
J1 H1:5,H2:5 10
J2 H2:3,H3:5 10
J3 H3:3,H4:5 10
E1 H1:3+4 10
```



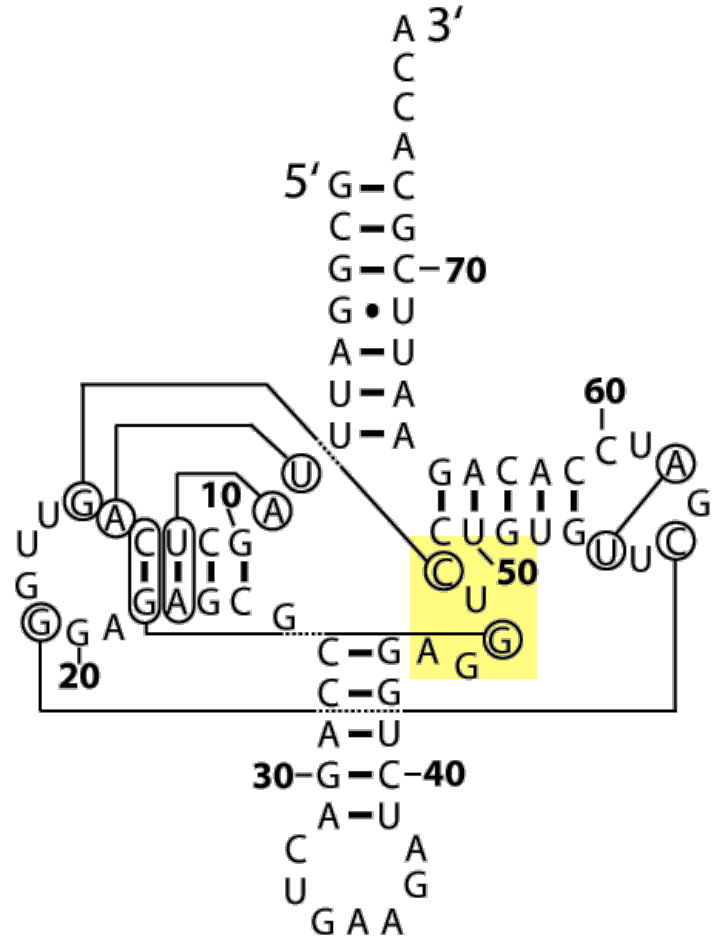
A quick look at fragment definitions

```
H1 1:7,66:72 10
H2 10:13,22:25 10
H3 27:31,39:43 10
H4 49:53,61:65 10
L1 H2 10
L2 H3 10
L3 H4 10
J1 H1:5,H2:5 10
J2 H2:3,H3:5 10
J3 H3:3,H4:5 10
E1 H1:3+4 10
```



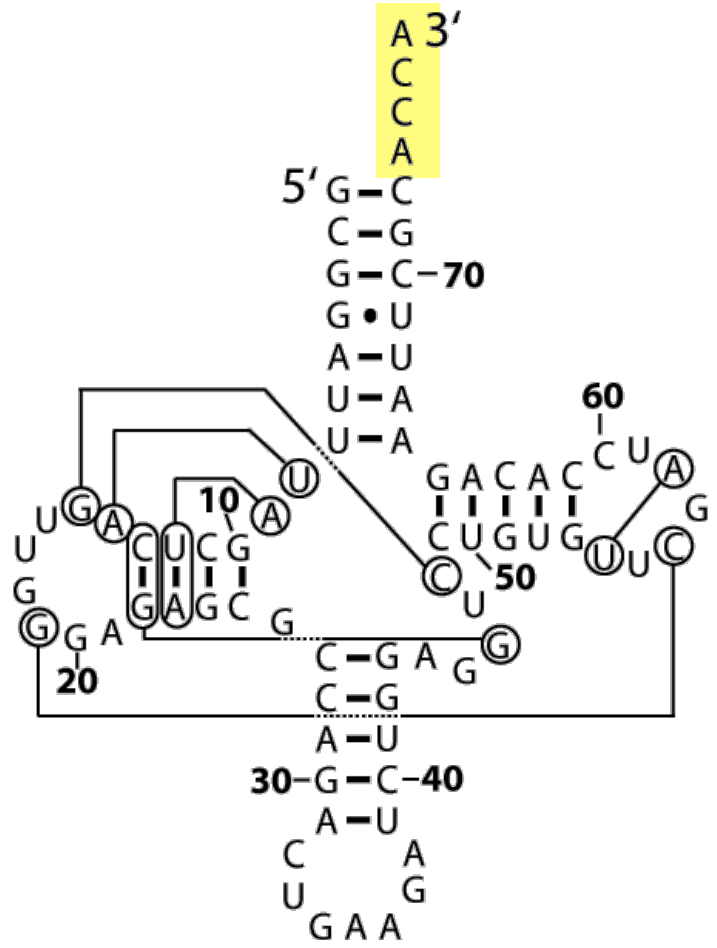
A quick look at fragment definitions

```
H1 1:7,66:72 10
H2 10:13,22:25 10
H3 27:31,39:43 10
H4 49:53,61:65 10
L1 H2 10
L2 H3 10
L3 H4 10
J1 H1:5,H2:5 10
J2 H2:3,H3:5 10
J3 H3:3,H4:5 10
E1 H1:3+4 10
```



A quick look at fragment definitions

```
H1 1:7,66:72 10
H2 10:13,22:25 10
H3 27:31,39:43 10
H4 49:53,61:65 10
L1 H2 10
L2 H3 10
L3 H4 10
J1 H1:5,H2:5 10
J2 H2:3,H3:5 10
J3 H3:3,H4:5 10
E1 H1:3+4 10
```



4. Run ex1.py

(Windows) `\Python26\python ex1.py`

(Mac OS) `python ex1.py`

5. Check the size of the output files

(Windows) `dir *-stats.dat`

(Mac OS) `ls -l *-stats.dat`

- **Make sure none of the *-stats.dat files have size 0. This would indicate that no matches were found for a particular fragment and prevent us from moving on to Part 2 of C2A.**

- **The library of full atomic fragment matches is saved in the file:**

6TNA-C3-1N32-subA-0-lib.pkl

Using naming format:

***<model name>*-*<reference name>*-*<model frame id>*-lib.pkl**

Part 2: Assembling a full atomic model – ex2.py

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run second part of c2a calcs (assemble molecules)."""


import sys, time
import simtk.nast.c2a_assembleOptions as ao

t0 = time.time()
ao.assemble(fragIn = '6TNA-FD.txt',
            coarseIn = '6TNA-C3.pdb',
            pieceLib = open('6TNA-C3-1N32-subA-0-lib.pkl'),
            outName = 'test',
            n = 1,
            cutoff = 0.5
           )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex2.py

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run second part of c2a calcs (assemble molecules)."""


import sys, time
import simtk.nast.c2a_assembleOptions as ao

t0 = time.time()
ao.assemble(fragIn = '6TNA-FD.txt',
            coarseIn = '6TNA-C3.pdb',
            pieceLib = open('6TNA-C3-1N32-subA-0-lib.pkl'),
            outName = 'test',
            n = 1,
            cutoff = 0.5
           )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex2.py

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run second part of c2a calcs (assemble molecules)."""


import sys, time
import simtk.nast.c2a_assembleOptions as ao

t0 = time.time()
ao.assemble(fragIn = '6TNA-FD.txt',
            coarseIn = '6TNA-C3.pdb',
            pieceLib = open('6TNA-C3-1N32-subA-0-lib.pkl'),
            outName = 'test',
            n = 1,
            cutoff = 0.5
           )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex2.py

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run second part of c2a calcs (assemble molecules)."""


import sys, time
import simtk.nast.c2a_assembleOptions as ao

t0 = time.time()
ao.assemble(fragIn = '6TNA-FD.txt',
            coarseIn = '6TNA-C3.pdb',
            pieceLib = open('6TNA-C3-1N32-subA-0-lib.pkl'),
            outName = 'test',
            n = 1,
            cutoff = 0.5
           )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex2.py

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run second part of c2a calcs (assemble molecules)."""


import sys, time
import simtk.nast.c2a_assembleOptions as ao

t0 = time.time()
ao.assemble(fragIn = '6TNA-FD.txt',
            coarseIn = '6TNA-C3.pdb',
            pieceLib = open('6TNA-C3-1N32-subA-0-lib.pkl'),
            outName = 'test',
            n = 1,
            cutoff = 0.5
           )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

ex2.py

```
#!/usr/bin/env python

__author__ = "Randall J. Radmer"
__version__ = "1.0"
__doc__ = """Script to run second part of c2a calcs (assemble molecules)."""


import sys, time
import simtk.nast.c2a_assembleOptions as ao

t0 = time.time()
ao.assemble(fragIn = '6TNA-FD.txt',
            coarseIn = '6TNA-C3.pdb',
            pieceLib = open('6TNA-C3-1N32-subA-0-lib.pkl'),
            outName = 'test',
            n = 1,
            cutoff = 0.5
           )

sys.stdout.write("Total run time: %.1f Sec\n" % (time.time()-t0))
```

7. Run ex2.py

(Windows) `\Python26\python ex2.py`

(Mac OS) `python ex2.py`

8. Visualize the output full atomic structure test-0.pdb in VMD

Run C2A on our NAST tRNA model

9. Copy T2-last.pdb from the 6TNA_MD directory

(Windows) `copy ..\6TNA_MD\T2-last.pdb .`

(Mac OS) `cp ../6TNA_MD/T2-last.pdb .`

10. Fix pdb syntax for c2a

(Windows) `\Python26\python fixvmdforc2a.py T2-last.pdb T2-M1.pdb`

(Mac OS) `python fixvmdforc2a.py T2-last.pdb T2-M1.pdb`

11. Copy ex1.py to myEx1.py and make the following change:

templateTrace = T2-M1.pdb

12. Run myEx1.py

Check for warnings that 0 options were found.

Check that the file T2-M1-1N32-subA-0-lib.pkl was created.

13. Copy ex2.py to myEx2.py and make the following change:

coarseIn = T2-M1.pdb

pieceLib = T2-M1-1N32-subA-0-lib.pkl

outName = T2-M1

14. Run myEx2.py:

(Windows) `\Python26\python myEx2.py`

(Mac OS) `python myEx2.py`

15. Visualize the full atomic output file T2-M1-0.pdb in VMD

The quality of the full atomic reconstruction depends very highly on the quality of the coarse-grain model.

C2A part 2 is a stochastic process -> repeat many times to get different results.

One last step

16. Modify the two output files for later use with Zephyr

(Windows) `\Python26\python fixpdbforgromacs.py test-0.pdb 6TNA-0-fixed.pdb`

(Mac OS) `python fixpdbforgromacs.py test-0.pdb 6TNA-0-fixed.pdb`

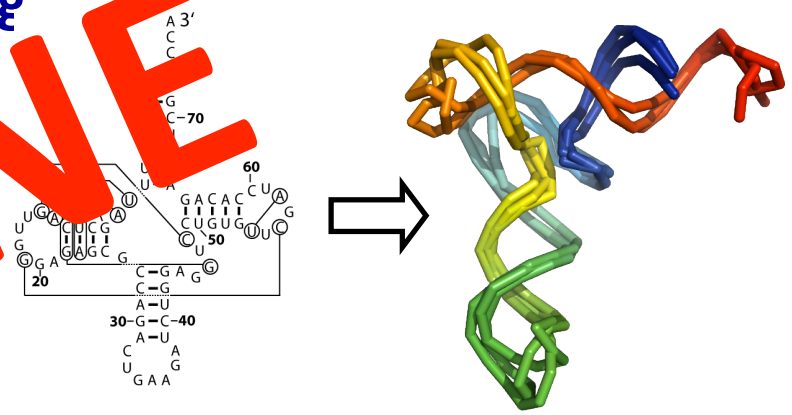
(Windows) `\Python26\python fixpdbforgromacs.py T2-M1-0.pdb T2-M1-0-fixed.pdb`

(Mac OS) `python fixpdbforgromacs.py T2-M1-0.pdb T2-M1-0-fixed.pdb`

Outline

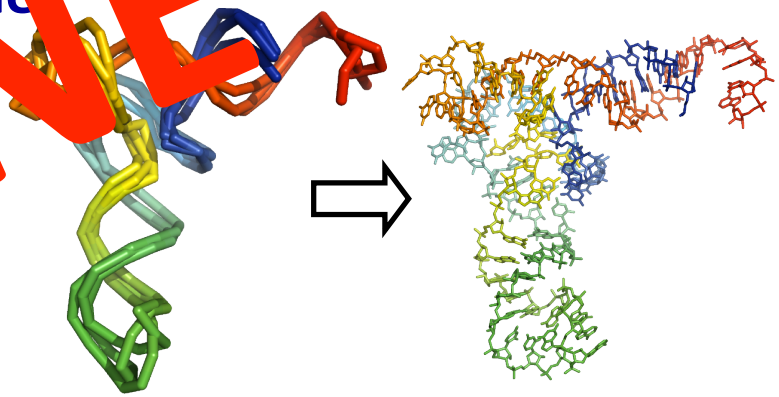
1. **Building coarse grain models using the Nucleic Acid Simulation Tool (NAST)**
   a) Theory
   b) Tutorial

DONE



2. **Adding full atomic detail using the Coarse to Atomic (C2A) tool**
   a) Theory
   b) Tutorial

DONE



3. **Bringing it together: from primary sequence to full atomic structure**

Bringing it all together: from secondary structure to full atomic 3D model.

Last exercise:

Download a secondary structure from the RNA STRAND database and build a full atomic 3D model.

The RNA STRAND database contains known and predicted secondary structures form 1000's of RNA molecules:

1. Open a web browser and navigate to:

www.rnasoft.ca/strand

www.rnasoft.ca/strand

2. Search for PDB_00005

***RNA STRAND v2.0* - The RNA secondary STRucture and statistical ANalysis Database**

[Home | Search | Analyse | Submit structures | News | Help]

RNA STRAND contains ***known RNA secondary structures*** of any type and organism. The ultimate goal of this database is to incorporate a comprehensive collection of known RNA secondary structures, and to provide the scientific community with simple yet powerful ways of **analysing, searching and updating** the proposed database.

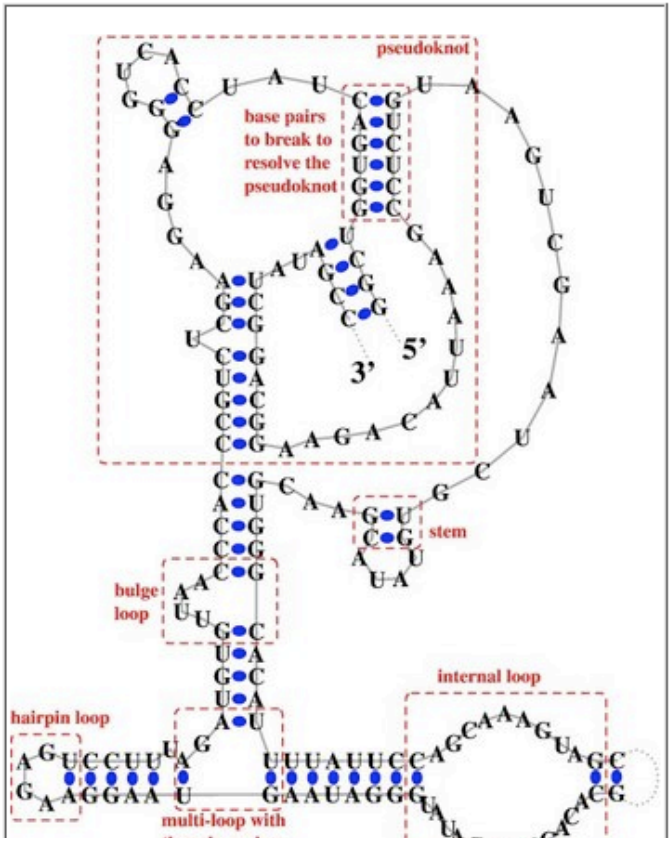
Current holdings: 4666 secondary structures in total.

PDB_00005 | Search RNA STRAND ID

Search	Search for RNA STRAND entries, supports multiple search criteria
Analyse	Analyse one or a group of RNA secondary structures
Submit	Submit new RNA secondary structures to RNA STRAND
News	News and updates on new releases of the database
Help	Brief explanations of RNA STRAND input and output fields, also accessible via the '[?]' links on any RNA STRAND page

Provenance of RNA STRAND structures

#RNAs	Source and link to source
1059	RCSB Protein Data Bank
1056	Gutell Lab CRW Site
726	tmRNA Database
622	Sprinzl tRNA Database
454	RNase P Database
383	SRP Database
313	Rfam Database
53	Nucleic Acid Database



Structural feature occurences in RNA STRAND

#RNAs	#Occurences	Structural motif
2333	6746	Pseudoknots
3582	17537	Multibranched loops
2992	35650	Internal loops
2898	31392	Bulge loops
4575	43442	Hairpin loops
2296	48730	Non-canonical base pairs

Most common RNA types in RNA STRAND

# RNAs	RNA type
726	Transfer Messenger RNA
723	16S Ribosomal RNA
707	Transfer RNA
470	Ribonuclease P RNA
450	Synthetic RNA
394	Signal Recognition Particle RNA
205	23S Ribosomal RNA
161	5S Ribosomal RNA
152	Group I Intron
146	Hammerhead Ribozyme
64	Other Ribosomal RNA
53	Other Ribozyme
42	Group II Intron

3. Select the Bpseq secondary structure format for PDB_00005

RNA STRAND v2.0 - The RNA secondary STRucture and statistical ANalysis Database

[Home | Search | Analyse | Submit structures | News | Help]

General features for molecule PDB_00005
(click to expand/contract all tables)

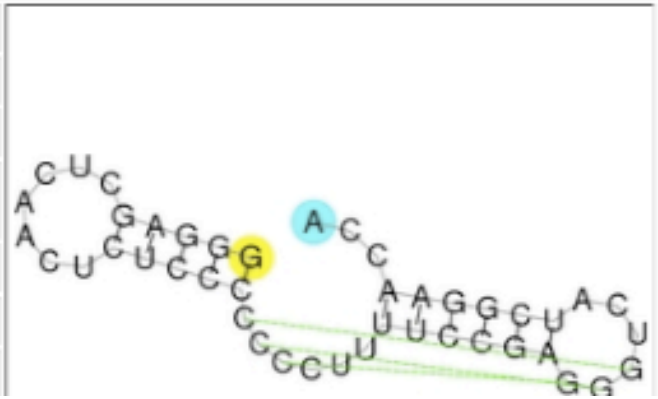
Format: CT

- CT
- RNAML
- Bpseq
- Dot-parentheses
- FASTA

View the RNA sequence and secondary structure for molecule PDB_00005

the RNA Secondary Structure Analyser for molecule PDB_00005

Molecule ID [?]:	PDB_00005
Molecule name [?]:	NMR STRUCTURE OF A CLASSICAL PSEUDOKNOT: INTERPLAY OF SINGLE-AND DOUBLE-STRANDED RNA, 24 STRUCTURES
Source [?]:	RCSB Protein Data Bank
Source ID [?]:	1A60
Reference [?]:	M.H.KOLK,M.VAN DER GRAAF,S.S.WIJMENGA,C.W.PLEIJ, H.A.HEUS,C.W.HILBERS. NMR STRUCTURE OF A CLASSICAL PSEUDOKNOT: INTERPLAY OF SINGLE- AND DOUBLE-STRANDED RNA.. SCIENCE V. 280 434 1998 ASTM SCIEAS US ISSN 0036-8075
Type [?]:	Other RNA
Organism [?]:	N/A
Validated by NMR or X-Ray [?]:	Yes
Method for secondary structure determination [?]:	NMR; ran through RNAview
Number of molecules [?]:	1
Length [?]:	44
Fragments used [?]:	Yes



4. Copy the data into a file named PDB_00005.bpseq

RNA STRAND v2.0 - The RNA secondary STRucture and statistical ANalysis Database

[Home | Search | Analyse | Submit structures | News | Help]

The *Bpseq* file for molecule PDB_00005.

```
# File PDB_00005.ct
# RNA SSTRAND database
# External source: RCSB Protein Data Bank 1A60, number of molecules: 1
# The secondary structure annotation was obtained with RNAview
1 G 17
2 G 16
3 G 15
4 A 14
5 G 13
6 C 0
7 U 0
8 C 0
9 A 0
10 A 0
11 C 0
12 U 0
13 C 5
14 U 4
15 C 3
16 C 2
17 C 1
18 C 32
19 C 31
20 C 30
21 C 0
22 U 0
```

5. Create a new directory for this exercise in the examples directory

```
cd ..
mkdir PDB_00005
cd PDB_00005
```

6. Place your new file PDB_00005.bpseq into this directory

7. Generate necessary input file for NAST:

(Windows) `\Python26\python ../parseBPseq.py PDB_00005`

(Mac OS) `python ../parseBPseq.py PDB_00005`

8. The following files should have been created:

PDB_00005-helix.txt

PDB_00005.seq

PDB_00005-FD.txt

9. Copy the following files into your PDB_00005 directory:

../6TNA_MD/myRunNast.py	(modify all four filenames to run NAST)
../6TNA_c2a/1N32-subA.pdb	(needed for c2a part 1)
../6TNA_c2a/baseConvert.pkl	(needed for c2a part 1)
../6TNA_c2a/fixvmdforc2a.py	(needed to fix pdb files saves in vmd)
../6TNA_c2a/fixpdbforgraomacs.py	(needed for final syntax fix)
../6TNA_c2a/myEx1.py	(modify templateTrace and fragmentFile)
../6TNA_c2a/myEx2.py	(modify fragIn, coarseIn, pieceLiv and outName)

10. Modify myRunNast.py and run NAST on PDB_00005

Note: set contactsFilename = ""

11. Use VMD to save a coarse-grain template

12. Fix the pdb syntax for C2A using fixvmdforc2a.py

13. Modify ex1.py and ex2.py and run C2A parts 1 & 2

14. Use fixpdbforgromacs.py to fix the pdb syntax for later Zephyr minimization